Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
*Escuela Politécnica Superior de Jaén*

Trabajo Fin de Grado

# DESARROLLO DE UN PAQUETE PARA EL ANÁLISIS Y LA PREDICCIÓN DE SERIES TEMPORALES

**Alumno: Alberto Vico Moreno**

Tutor: Prof. D. Antonio Jesús Rivera Rivas
Prof. Dª. Mª Dolores Pérez Godoy
Dpto: Informática

**Septiembre, 2017**

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Antonio Jesús Rivera Rivas y Doña Mª Dolores Pérez Godoy , tutores del Proyecto Fin de Carrera titulado: Desarrollo de un Paquete para el Análisis y la Predicción de Series Temporales, que presenta Alberto Vico Moreno, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Septiembre de 2017

El alumno: Los tutores:

RIVERA RIVAS ANTONIO JESUS - 26015863B
Firmado digitalmente por RIVERA RIVAS ANTONIO JESUS - 26015863B
Fecha: 2017.09.05 11:47:52 +02'00'

PEREZ GODOY MARIA DOLORES - 75093149N
Firmado digitalmente por PEREZ GODOY MARIA DOLORES - 75093149N
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-75093149N, givenName=MARIA DOLORES, sn=PEREZ GODOY, cn=PEREZ GODOY MARIA DOLORES - 75093149N
Fecha: 2017.09.05 11:59:51 +02'00'

Fdo: Antonio Jesús Rivera Rivas

Fdo: Alberto Vico Moreno Fdo: Mª Dolores Pérez Godoy

# Índice

# Índice de ilustraciones

# Índice de tablas

# 1. Introducción y motivación

En este primer apartado se explicará a grandes rasgos el contenido y la motivación de este T.F.G. con el objetivo de ir haciendo una idea de lo que se verá después.

El trabajo consiste en el diseño y la implementación de un paquete en R para el análisis, procesamiento y predicción de series temporales.

La elaboración de este proyecto ha requerido por mi parte una investigación previa sobre las series temporales, muy ligadas a otras materias como la estadística, y un aprendizaje profundo de un lenguaje que no conocía como es R. Por ello la fase previa de documentación e investigación ha sido especialmente larga y complicada para mí. Aun así me encuentro satisfecho por todo lo aprendido con este trabajo y puedo decir que ha valido la pena el esfuerzo.

## 1.1. Introducción

Hoy en día es enorme la cantidad de datos que maneja cualquier entidad ya sea una empresa, una administración o un grupo de investigación. Por no hablar de la cantidad de bases de datos públicas en internet. Los últimos años, el crecimiento de los datos ha sido enorme, y no hay signos de que este crecimiento vaya a frenarse. Por esto son necesarias y son demandadas cada vez más nuevas formas de darles un mayor y más eficiente rendimiento a todos esos datos.

Se trata del llamado Business Intelligence (BI) o inteligencia empresarial. Dentro de este campo se engloban todas las estrategias y tecnologías que utilizan las organizaciones para el análisis de datos, sea enfocado a una posterior toma de decisiones o para adquirir conocimiento (1).

**Ilustración 1.1 El BI es clave para las empresas**

En la actualidad hay multitud de empresas que se dedican en parte o exclusivamente a proveer BI a sus clientes. Las grandes entidades, en cambio, suelen tener sus propios departamentos de inteligencia empresarial.

Algunos ejemplos de las aplicaciones del BI:

- Creación de informes: recopilación y visualización de datos.
- Análisis de datos: Minería de Datos, modelos predictivos, análisis estadístico, etc.
- Gestión del conocimiento: aprendizaje y adopción de medidas.

Este T.F.G. se centra en la parte de análisis de datos, concretamente en la creación de modelos predictivos a partir de series temporales. Pero, ¿qué es una serie temporal? Se explicará detalladamente más adelante.

## 1.2. Motivación

Como hemos visto, el Business Intelligence es muy necesario hoy en día para cualquier empresa, y lo seguirá siendo en un futuro.

Uno de los campos más importantes del Business Intelligence es la predicción de datos futuros a partir de datos pasados. Estos modelos predictivos consisten en encontrar y

cuantificar patrones en los datos utilizando modelos matemáticos complejos que se pueden usar para predecir futuros resultados. Estas predicciones no siempre serán exactas, pero ayudarán a una empresa o entidad a optimizar sus recursos y a tomar decisiones para el futuro.

Podemos hacer predicciones a partir de una serie temporal, que es, a grandes rasgos, una secuencia discreta de medidas o datos tomados a intervalos regulares y ordenados en el tiempo. Estos datos pueden tener una o varias variables (2).

Algunos casos en que las predicciones sobre serie temporales serían útiles:

- Una serie temporal de alumnos matriculados en una universidad puede ayudar a prever futuros presupuestos que se necesitarán.
- Una predicción sobre el crecimiento de la población en una nación puede ayudar a un gobierno a planear una utilización adecuada de los recursos nacionales para canalizarlos en determinados sectores o zonas.
- El crecimiento económico de un banco es otra serie que puede ser útil; es controlada por varios factores incluyendo ratio de interés, préstamos sancionados en el año actual, población de nuevos depositantes y muchos más. Predecir el crecimiento puede ayudar al banco a modificar sus políticas de reasignación de activos para optimizar el beneficio.

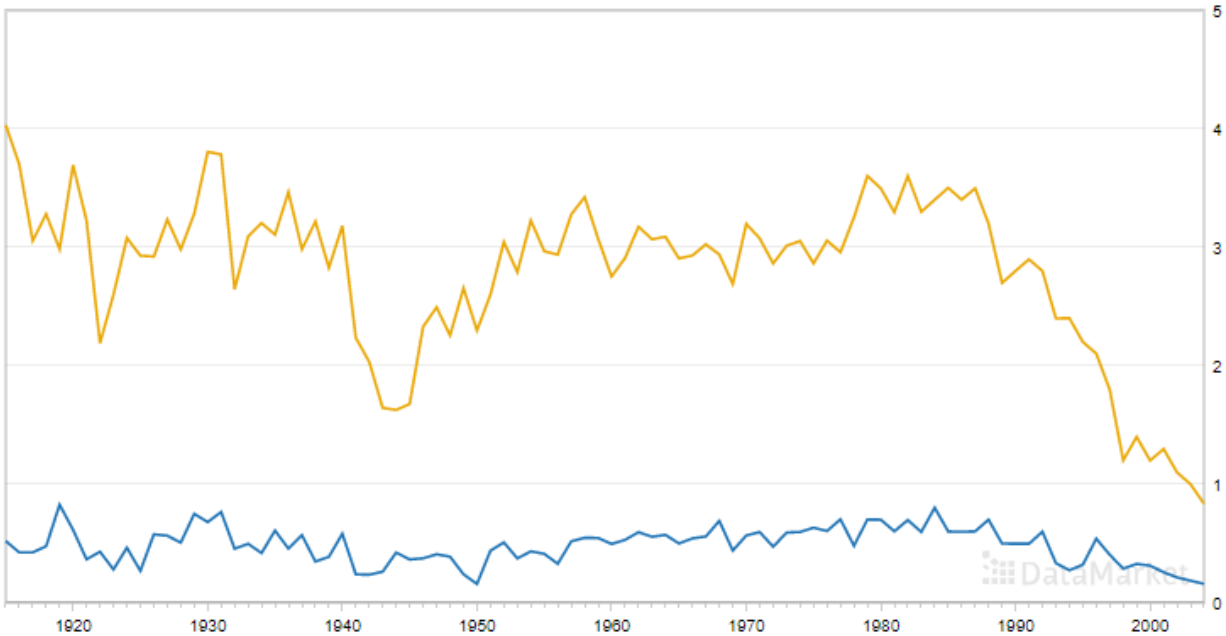



**Ilustración 1.2 Serie temporal: Suicidios(en amarillo) y homicidios(en azul) ocurridos en Australia por cada 100.000 habitantes. 1915-2004**

En la actualidad no existen muchas herramientas que faciliten la tarea de predicción de series temporales. Para crear un modelo predictivo, es necesario estudiar a fondo previamente el proceso que se ha de llevar a cabo, que puede llegar a ser muy complejo para un usuario no experto en la materia.

En R, el lenguaje de programación para análisis estadístico por excelencia, no existe ninguna biblioteca que agrupe las distintas fases del proceso de creación de un modelo ni los diferentes tipos de métodos para la predicción.

En este trabajo implementaremos una biblioteca para R que integre tanto las fases del proceso como varios tipos de modelos predictivos con el objetivo de distribuirla en el repositorio público de paquetes de R. Nos proponemos que el trabajo sobre la biblioteca sea fácil y que se automatice el proceso en la medida de lo posible para evitar que un usuario no experto tenga que estudiar a fondo el significado de todos los parámetros. A su vez, la biblioteca debe ser flexible y permitir la modificación y la experimentación sobre todas las funciones y sus parámetros, para hacerla atractiva también a un usuario experto.

Se estudiarán distintos métodos para cada fase del proceso, y se implementarán algunos de ellos. Se hará especial hincapié y se investigará sobre la predicción con algoritmos de regresión de Minería de Datos, cuyo uso no está muy extendido en el campo de las series temporales; no existe ninguna biblioteca que lleve a cabo esta tarea.

Dada la naturaleza de código abierto de R y sus bibliotecas, el objetivo final de este trabajo es sentar una base sobre la que seguir construyendo: los usuarios podrán ir añadiendo al paquete nuevos métodos que no estén implementados, lo que a la larga daría como resultado una herramienta de predicción sobre series temporales muy potente y completa.

# 2. Planificación y objetivos

En este apartado esbozaremos los objetivos a conseguir, la metodología que se siguió en la elaboración del trabajo, la estructura de esta memoria y la planificación previa realizada.

## 2.1. Objetivos

Nos planteamos los siguientes objetivos:

- Estudio a fondo de las series temporales y sus modelos predictivos más usados, haciendo especial hincapié en el modelo ARIMA.
- Instalación de todo el software necesario para desarrollar en R. Aprender el lenguaje de programación y cómo tratar las series temporales.
- Aprendizaje en el manejo de la biblioteca caret, que provee de herramientas para crear modelos predictivos de Minería de Datos y que cuenta con un gran número de algoritmos implementados.
- Diseño del paquete y de los módulos, clases y funciones indispensables.
- Desarrollo de los módulos necesarios. Las funciones deben ser lo más sencillas y automáticas posible. Debe haber opciones de visualización que faciliten la comprensión de lo que el usuario está haciendo.
- Documentación del código con roxygen2 y creación del paquete.
- Distribución del paquete en CRAN.
- Pruebas y experimentación utilizando el paquete.

## 2.2. Metodología

El desarrollo del trabajo se ha llevado a cabo de la siguiente manera:

- Estudiar las series temporales y sus características, así como los modelos predictivos más usuales.
- Instalar el software necesario para desarrollar y crear paquetes en R.
- Adquirir las nociones básicas de R necesarias para empezar a desarrollar.
- Aprender sobre el pre-procesamiento de series temporales y cómo influye sobre las predicciones.
- Aprender cómo funciona el modelo predictivo ARIMA y cómo trasladarlo a R.
- Aprender a utilizar el paquete caret.

- Diseñar el paquete, teniendo en cuenta el flujo de trabajo más probable de un posible usuario.
- Crear un módulo de pre-procesamiento.
- Crear un módulo de modelado.
- Crear un módulo de predicción.
- Crear un módulo de post-procesamiento.
- Conseguir que el proceso sea lo más gráfico posible, creando herramientas para la visualización de la serie en distintas etapas y haciendo una función para comparar distintos modelos.
- Documentar exhaustivamente el código con roxygen2.
- Construir y checkear el paquete.
- Distribuir el paquete en CRAN.
- Llevar a cabo una experimentación sobre algunas series temporales, variando parámetros y opciones.
- Redactar la memoria del T.F.G.

## 2.3. Estructura de la memoria

Decidimos estructurar la memoria de la siguiente forma:

El capítulo 1 sirve para poner al lector en antecedentes sobre la utilidad del Business Inteligence y la predicción de datos, así como la importancia que tienen hoy en día el tratamiento de las cantidades masivas de datos de las que disponemos. Relacionamos este hecho con el objetivo de este trabajo: proveer de una nueva herramienta en forma de paquete en R para la predicción de series temporales.

El capítulo 2 trata sobre la fase previa a la elaboración del proyecto, en la que nos planteamos los objetivos a cumplir y las fases que pasaremos a lo largo del tiempo. También se describe la metodología seguida en el desarrollo del trabajo para dar una idea de las tareas que he ido realizando.

En el capítulo 3 se estudiará la teoría sobre series temporales, sus características y distintos métodos para el análisis y la predicción. Nos detendremos especialmente en los modelos ARIMA y en Minería de Datos, que son los implementados en el paquete.

El capítulo 4 nos trasladará más de cerca a la elaboración del paquete. Hablaremos de las primeras fases de diseño, veremos algunos diagramas de Ingeniería del Software y concretaremos un poco más las diferentes herramientas que nos ofrece este paquete. También hablaremos aquí de la fase de documentación, construcción y distribución del paquete.

El capítulo 5, el manual de usuario, enseñará al lector el funcionamiento exacto de cada una de las funciones y herramientas de las que dispone el paquete. Lo haremos siguiendo el flujo de trabajo lógico esperado por el usuario.

En el capítulo 6 llevaremos a cabo una serie de pruebas y experimentación sobre varias series temporales, alterando los parámetros y jugando con las distintas opciones que nos ofrece el paquete.

Por último el capítulo 7 contendrá las conclusiones a las que se llegan tras todo el proceso de elaboración de este trabajo.

## 2.4. Planificación

En la realización de este T.F.G. se han empleado aproximadamente unos 6 meses, desde febrero hasta agosto de 2017.

Es importante hacer una planificación previa del tiempo que vamos a emplear. En un caso de un proyecto real, nos ayudaría a estimar el presupuesto y los recursos que se van a necesitar.

Para visualizar esta planificación utilizaremos un diagrama de Gantt. Tomaremos como tareas los puntos especificados en la metodología un poco más sintetizados.

| Tarea | Inicio | Duración | Fin |
|---|---|---|---|
| **Estudio series temporales** | 15-feb | 15 | 02-mar |
| **Instalación software** | 02-mar | 1 | 03-mar |
| **Aprendizaje R** | 03-mar | 20 | 23-mar |
| **Aprendizaje métodos y modelos** | 23-mar | 15 | 07-abr |
| **Diseño del paquete** | 07-abr | 5 | 12-abr |
| **Módulo pre-procesamiento** | 12-abr | 20 | 02-may |
| **Módulo modelado** | 02-may | 25 | 27-may |
| **Módulo predicción** | 27-may | 15 | 11-jun |

| | | | |
|---|---|---|---|
| **Módulo post-procesamiento** | 11-jun | 15 | 26-jun |
| **Documentación roxygen2** | 26-jun | 5 | 01-jul |
| **Creación y distribución del paquete** | 01-jul | 2 | 03-jul |
| **Experimentación** | 03-jul | 4 | 07-jul |
| **Redacción de la memoria** | 07-jul | 40 | 16-ago |

**Tabla 2.1 Tareas del diagrama de Gantt**

A continuación el diagrama:

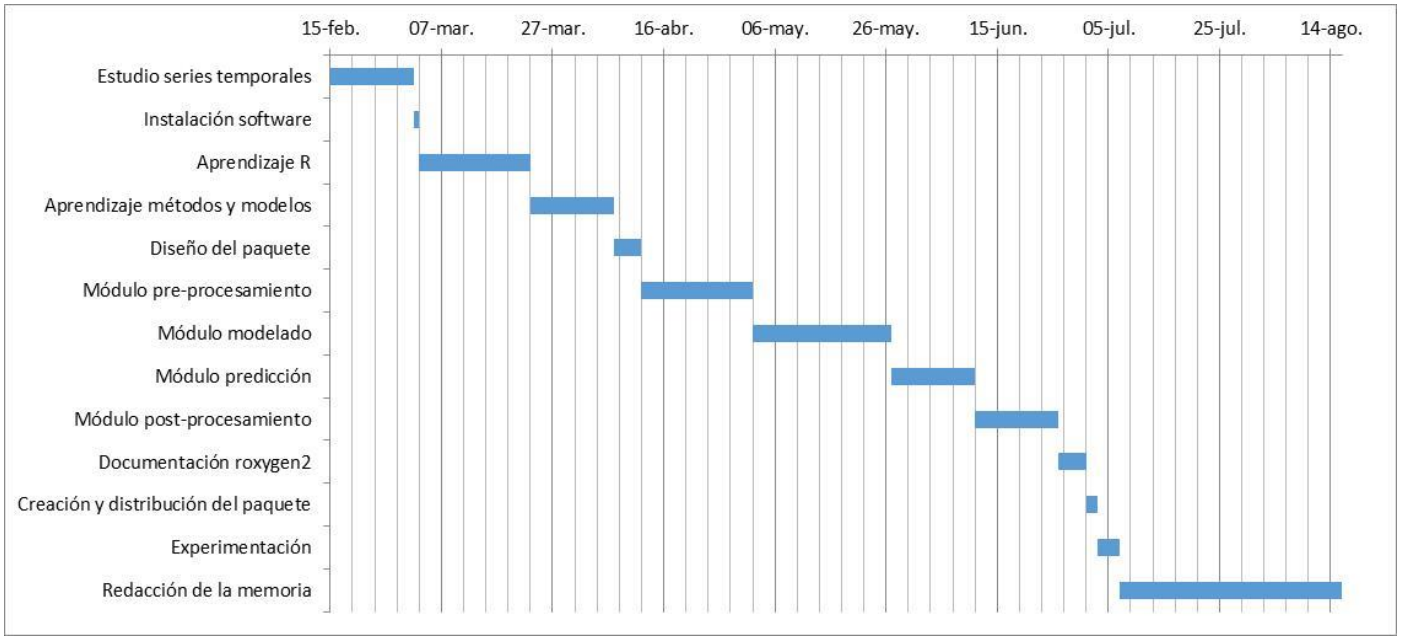



**Ilustración 2.1 Diagrama de Gantt**

# 3. Análisis de series temporales

## 3.1. Definición

Llamamos serie temporal a un conjunto de datos que representa una colección de valores obtenidos a partir de mediciones secuenciales a los largo del tiempo. Ejemplos de series temporales son tanto la medida de altura de las mareas oceánicas tomada mensualmente como el último valor diario de las acciones de una empresa en bolsa. (3)

Dada su naturaleza simplista resulta muy fácil interpretar una serie temporal dibujada en un gráfico, por lo que son muy comunes.

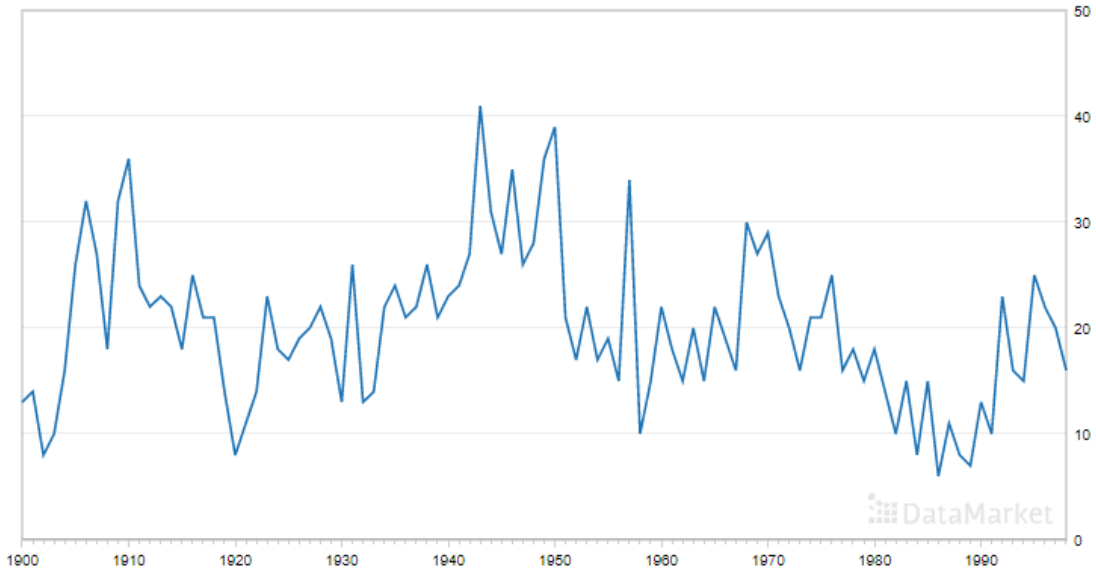



**Ilustración 3.1 Serie temporal: Número de terremotos anuales. 1900-1998**


**Ilustración 3.2 Serie temporal: Ratio de cambio del dólar australiano. $A por 1 US $. Media mensual. 1969-1995**

Las series temporales son usadas en estadística, reconocimiento de patrones, econometría, matemáticas financieras, previsión meteorológica, predicción de terremotos, astronomía, ingeniería de comunicaciones, y básicamente cualquier campo de ciencia aplicada o ingeniería que involucre mediciones temporales.

En una serie temporal, llamamos frecuencia al número de mediciones anuales.

Dependiendo del número de variables una serie temporal puede ser:

- Monovariable: para cada medición se recoge una sóla muestra o variable. Es lo más común y sobre lo que nos vamos a centrar en este trabajo. Las ilustraciones 3.1 y 3.2 son serie temporables monovariables.
- Multivariable: cada medición recoge datos de dos o más fuentes. No son muy comunes, pero es interesante conocerlas. La siguiente figura representa datos cuatrimestrales de una compañía: “Sales” contiene las ventas, “AdBudget” el presupuesto para publicidad y “GDP” el producto interno bruto.

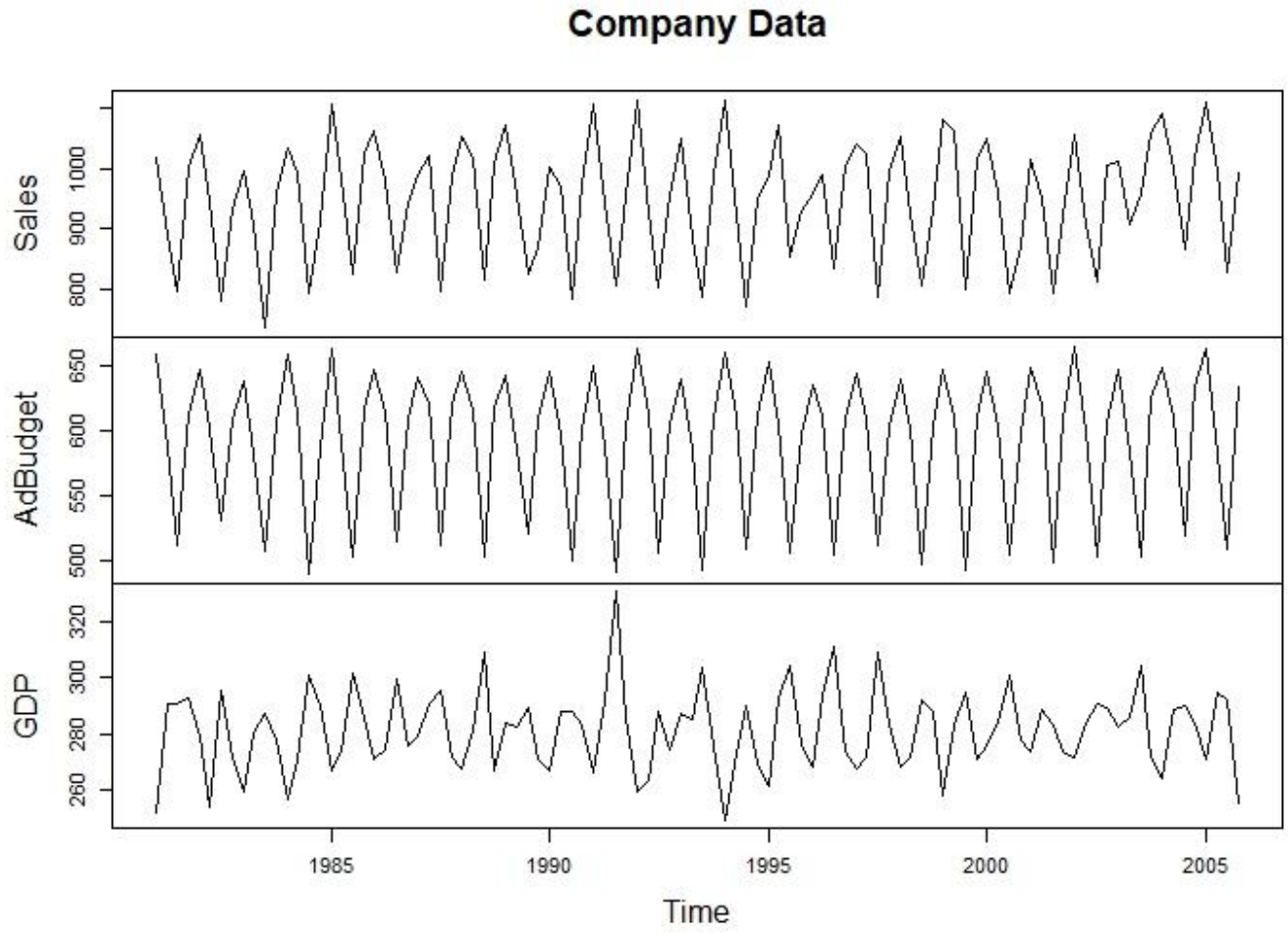



**Ilustración 3.3 Serie temporal multivariable: Datos cuatrimestrales de una compañía. 1981-2005**

El análisis de series temporales comprende métodos para analizar los datos de una serie temporal con el objetivo de extraer estadísticas significativas y otras características de los datos. La predicción de series temporales es el uso de un modelo para predecir valores futuros basándose en valores observados anteriormente.

## 3.2. Componentes

El análisis más clásico de las series temporales se basa en que los valores que toma la variable de observación es la consecuencia de cuatro componentes, cuya actuación conjunta da como resultado los valores medidos. (4) Estas componentes son:

- Tendencia: refleja el movimiento general de la variable en el periodo de observación, es decir, el cambio a largo plazo del nivel de la serie.
- Variación estacional o estacionalidad: fluctuaciones de la variable que se producen y repiten en periodos de tiempo cortos, normalmente no superiores a

un año. Una serie es estacional cuando podemos observar en ella un patrón sistemático.

- Variación cíclica: oscilaciones periódicas de amplitud superior a un año. Tiene un periodo y amplitud variables.
- Variación aleatoria o ruido: también denominada residuo, no muestran ninguna regularidad; suelen ocurrir debido a fenómenos de carácter ocasional.

En la siguiente figura se ilustra estas componentes en cuatro series distintas:

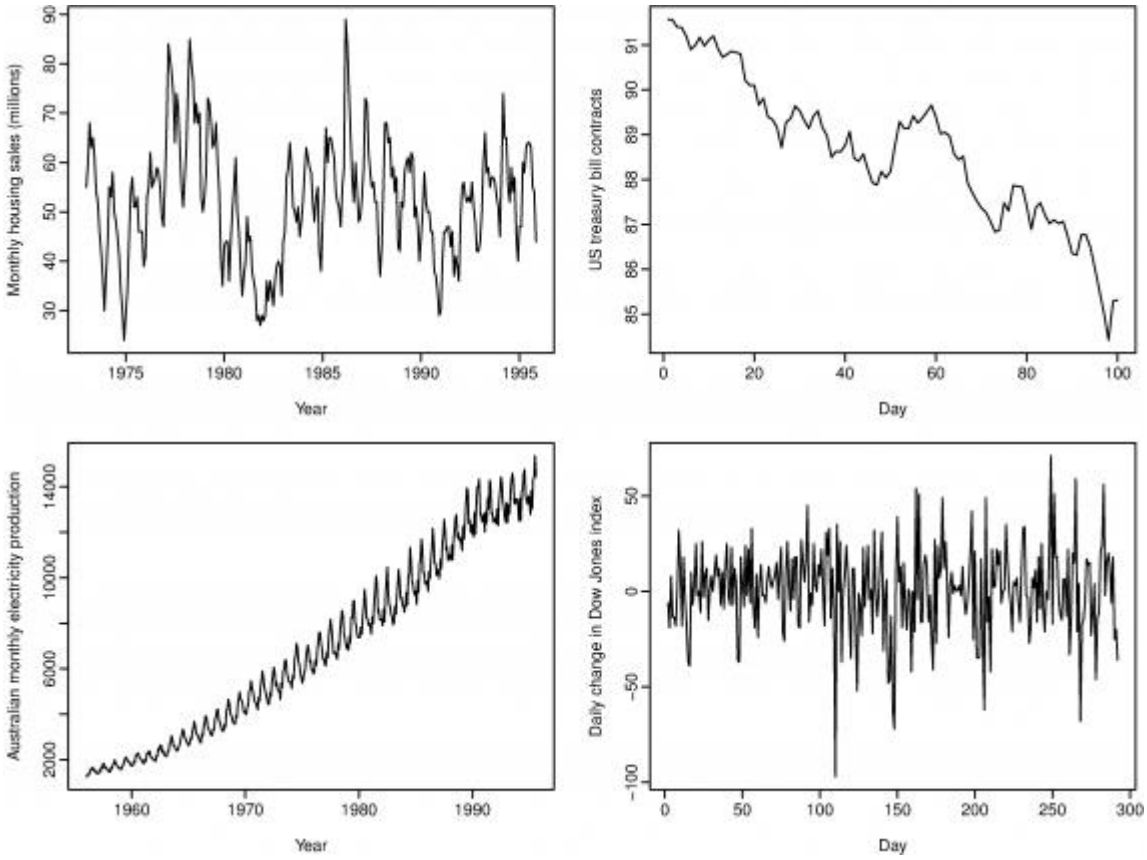



**Ilustración 3.4 Cuatro series temporales mostrando distintos tipos de patrones de comportamiento**

Arriba a la izquierda: venta de casas (mensual), muestra comportamiento estacional y cíclico. Arriba a la derecha: resultados del mercado de Chicago durante cien días consecutivos, muestra una clara tendencia a la baja. Abajo a la izquierda: producción mensual de electricidad en Australia, obvia componente estacional y de tendencia. Abajo a la derecha: el cambio diario del estado de Dow Jones no sigue ningún patrón reconocible. (5)

## 3.3. Pre-procesamiento

Un concepto importante es la estacionariedad. Una serie temporal es estacionaria si las variables aleatorias que lo componen poseen la misma función de distribución de probabilidad, con independencia del momento del tiempo considerado. (5) (6)

En otras palabras, una serie temporal es estacionaria cuando no tiene tendencia ni estacionalidad y la varianza permanece constante a lo largo del tiempo. En una serie estacionaria ni la media, ni la varianza ni las autocorrelaciones dependen del tiempo.

Esto significa que si conseguimos transformar una serie en estacionaria podremos estudiar la presencia de patrones en la serie (que no dependerán del tiempo) para identificar un posible modelo matemático.

En eso consiste el pre-procesamiento de una serie temporal: en eliminar, si las hubiera, la tendencia y la estacionalidad, y además conseguir una varianza lo más homogénea posible. Pero, ¿cómo hacemos esto?

A continuación describimos los métodos que se han utilizado en este trabajo para alcanzar este propósito. Representaremos gráficamente los efectos de uno de cada tipo sobre la siguiente serie temporal.

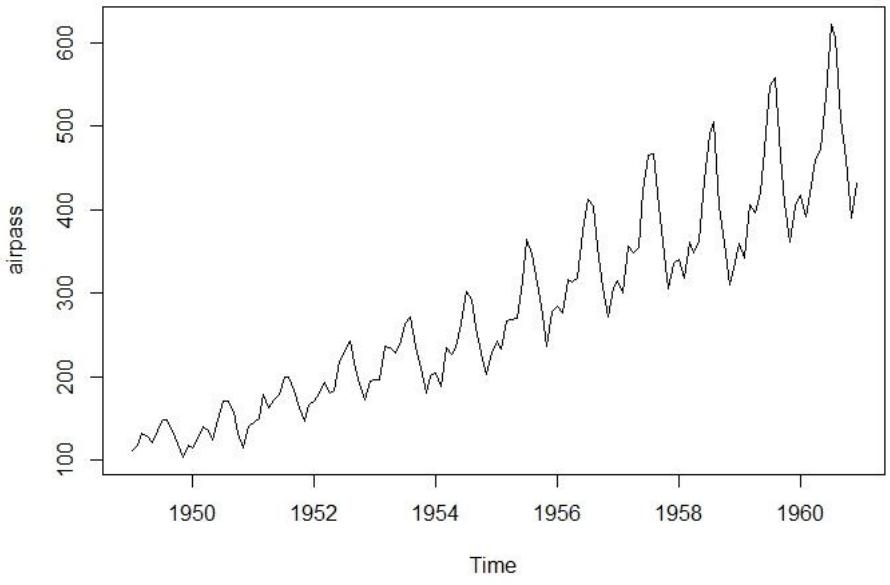



**Ilustración 3.5 Serie temporal: Total mensual de pasajeros internacionales de una aerolínea. 1949-1960**

Podemos observar claramente como la varianza aumenta con el tiempo, existe una tendencia al alza y posee una estacionalidad muy marcada.

### 3.3.1. Transformación logarítmica para homogeneizar la varianza

Aplicar el logaritmo a una serie temporal reduce la distancia entre valores extremos y no extremos.

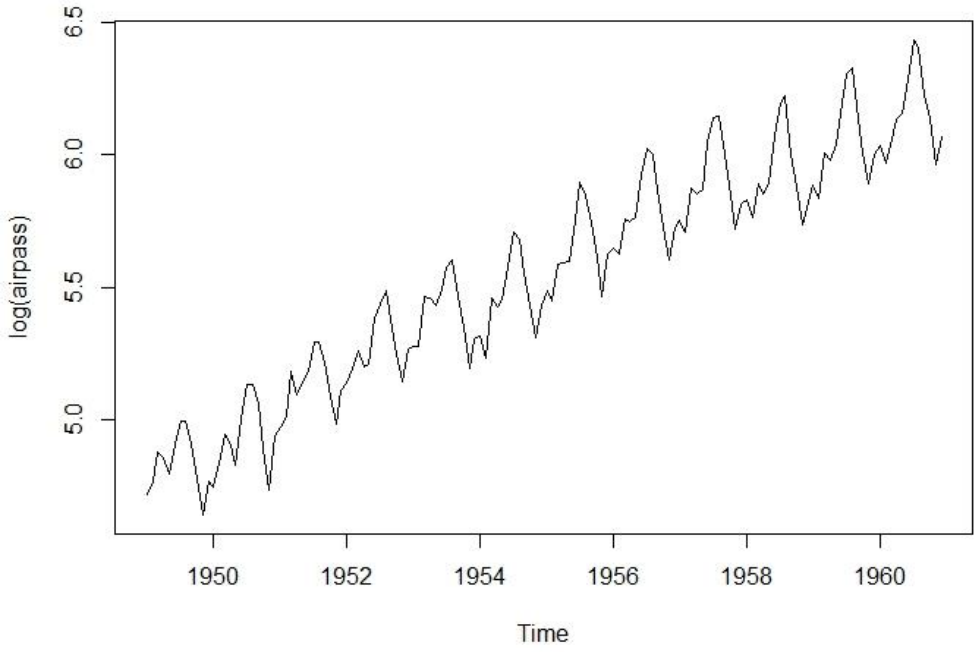



**Ilustración 3.6 Transformación logarítmica**

Se puede ver como la varianza se ha ajustado y ya no aumenta con el tiempo.

### 3.3.2. Transformación Box-Cox para homogeneizar la varianza

Se trata de un método clásico para corregir sesgos en la distribución de errores y para corregir varianzas desiguales a partir de un modelo matemático que depende de un parámetro lambda. (7)

### 3.3.3.Diferencias regulares para eliminar la tendencia

Consiste en aplicar diferenciación con tantas diferencias como sea necesario hasta que la tendencia sea anulada. (8) Las diferencias regulares consisten en aplicar a cada valor de la serie la diferencia del valor inmediatamente anterior de la siguiente forma:

$$Xi = Xi - Xi\text{-}1$$

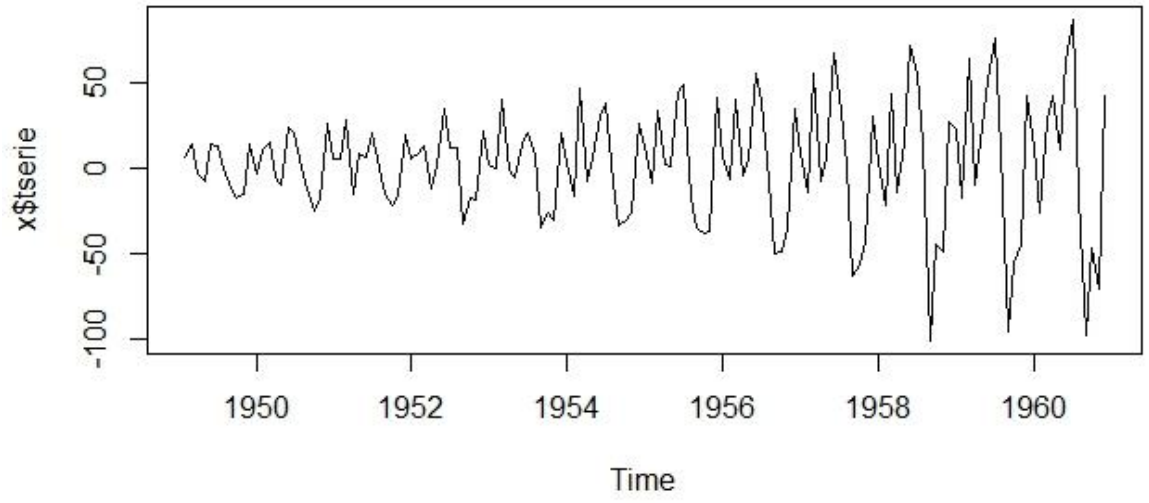



**Ilustración 3.7 Diferenciación regular de orden 1**

Vemos como ha desaparecido la tendencia al alza.

### 3.3.4.Extracción de medias estacionales para eliminar la tendencia

Este método fue creado como una manera de corregir la tendencia de forma automática. Fue propuesto por Weizhong Yan, y consiste en dividir la serie en segmentos del mismo tamaño que la frecuencia de la serie. Se extrae la media de las observaciones históricas en cada uno de estos segmentos. Sean {M1, M2, … ,Mk} las medias de los k segmentos de la serie y sea l la frecuencia. (9) La serie sin tendencia DTi se relaciona con la serie original Xi como sigue:

$$DTi = Xi - Mj=cell(i/j),\ i = 1, 2, \dots, n$$

Donde n es el tamaño de la serie.

### 3.3.5. Diferencias estacionales para eliminar la estacionalidad

Consiste en aplicar diferenciación con tantas diferencias estacionales como sea necesario hasta que la estacionalidad quede eliminada. Son como las regulares pero, en lugar de restar el valor inmediatamente anterior, restamos el valor correspondiente a la misma fecha del año anterior. (8) Lo conseguimos utilizando la frecuencia de la serie de la siguiente forma:

$$Xi = Xi - Xi\text{-frecuencia(ts)}$$

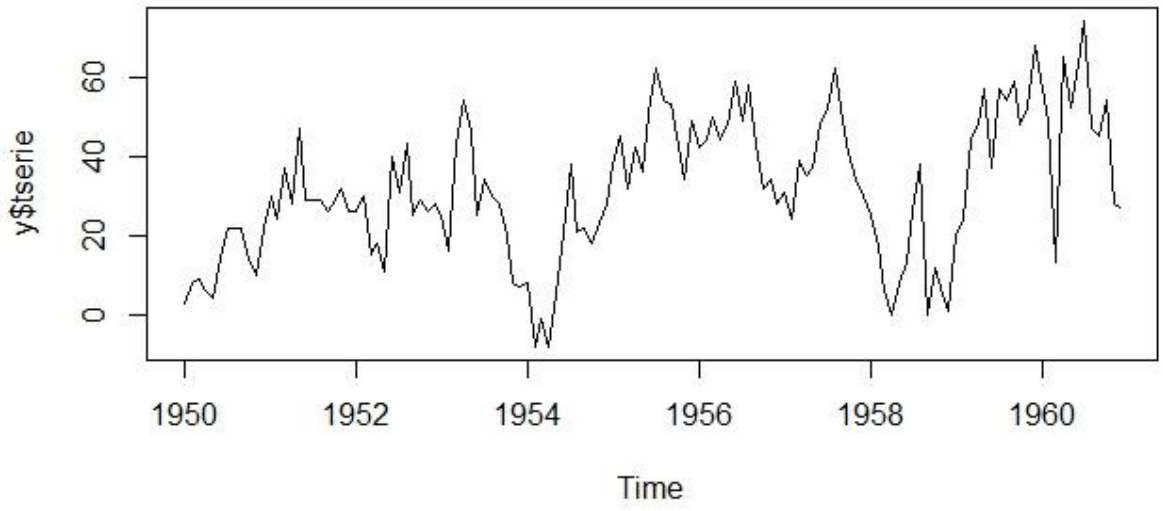



**Ilustración 3.8 Diferenciación estacional de orden 1**

Si prestamos atención a este último gráfico, vemos que en efecto se ha eliminado la estacionalidad. Pero aparentemente también nos hemos deshecho de la tendencia sin haber aplicado diferencias regulares.

Esto puede ocurrir, y tendremos que tenerlo en cuenta. Siempre debemos revisar el estado de la serie antes de efectuar transformaciones sobre ella.

### 3.3.6. ¿Cómo nos aseguramos de que una serie es estacionaria?

No existe un método infalible. Tenemos varias opciones: (10) (5)

- Visualización de las gráficas: podemos deducir si una serie es estacionaria si prestamos atención a su gráfica. Sin embargo, no siempre resulta evidente.

- Tests de raíz unitaria: se trata de tests de hipótesis estadísticos de estacionariedad que indican si es necesario diferenciar. Hay varios tests disponibles, y están basados en diferentes suposiciones y pueden llevar a respuestas conflictivas. Uno de los más populares es el Augmented Dickey-Fuller (ADF). Para este test se estima el siguiente modelo de regresión:

$$Y't=\phi Yt-1 + \beta 1Y't-1 + \beta 2Y't-2 + \cdots + \beta kY't-k'$$

  Donde Y't denota la primera serie diferenciada, Y't = Yt - Yt-1 y k es el número de lags a incluir en la regresión. Si la serie original Yt necesita diferenciación el coeficiente $\phi$ debería ser aproximadamente cero. Si Yt ya es estacionaria, $\phi < 0$.

  Estos métodos tampoco son infalibles, pero ayudan en la comprobación.

- Correelograma o diagrama de autocorrelación (ACF): una herramienta muy importante de diagnóstico. Es una sucesión de valores que miden el grado de dependencia entre observaciones próximas, y que habitualmente se representan con barras. ¿Cómo lo interpretamos? En el correelograma de una serie estacionaria, la dependencia temporal debe acercarse a cero rápidamente. Un decrecimiento lento de las autocorrelaciones indica que la serie no es estacionaria.

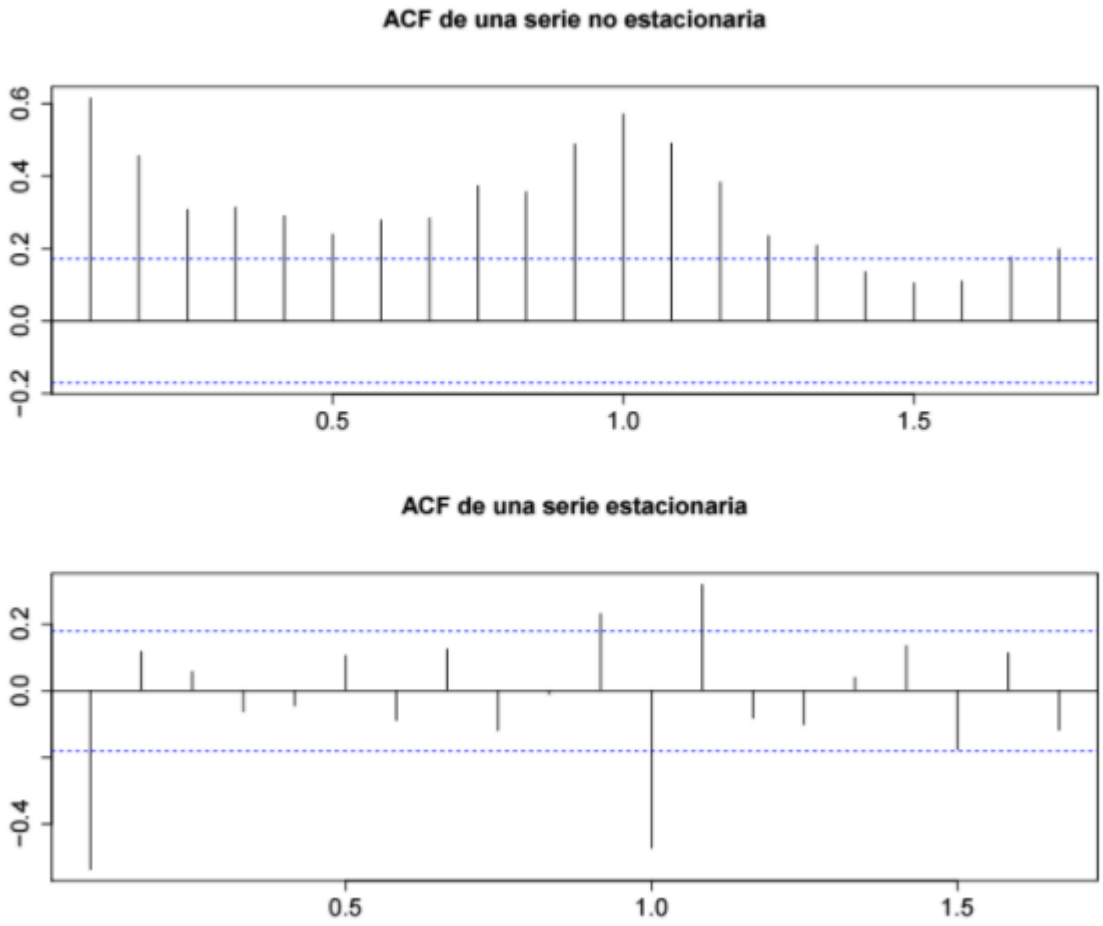



**Ilustración 3.9 Comparativa de ACF de una serie no estacionaria y otra estacionaria**

Una vez nos hemos asegurado de la estacionariedad de una serie, podemos empezar a aplicar modelos predictivos. Si la serie no es estacionaria cuando construyamos el modelo, los resultados serán muy malos.

Por último debemos tener en cuenta todas las transformaciones que hemos realizado a la serie. Una vez hayamos encontrado las predicciones, querremos revertir los cambios para observar los valores y las gráficas reales.

## 3.4. Algunos modelos sencillos

En este apartado veremos una primera aproximación a los métodos de predicción mediante la explicación de algunos de los más sencillos (10) (5).

- Método de la media: las predicciones de valores futuros serán iguales a la media de los datos históricos. Podemos describir este modelo como:

$$Y\hat{}\ t+h|t = y\hat{} = (Y1 + ... + Yt)/t$$

- Método naïve (ingenuo en inglés): todas las predicciones son determinadas simplemente por el valor de la última observación. Las predicciones de todos los valores futures serán Yt, donde Yt es el último valor observado.
- Método drift (deriva en inglés): una variación del método naïve que permite a las predicciones incrementarse o decrementarse durante el tiempo, donde la cantidad de cambio a lo largo del tiempo(deriva) será el cambio medio visto en los datos históricos. Es el equivalente a dibujar una línea entre la primera y la última observación y extrapolarla al futuro.

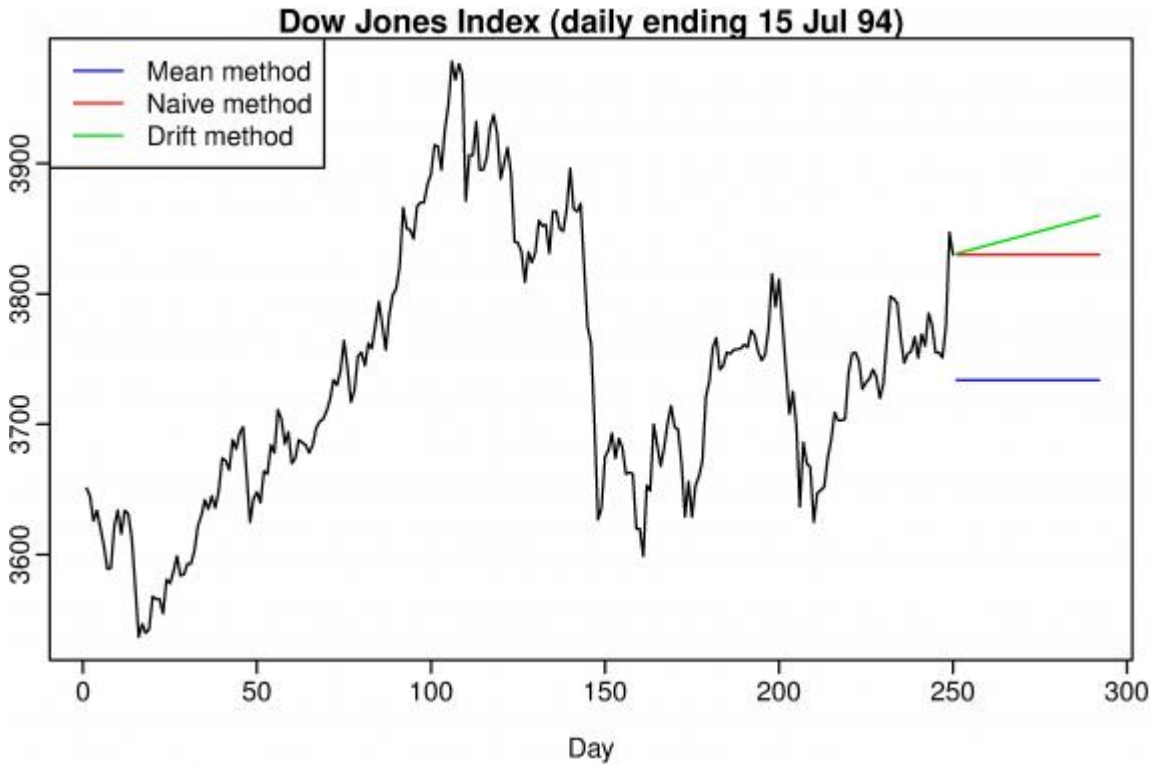



**Ilustración 3.10 Serie temporal: Dow Jones diario. Visualización de las predicciones resultado de aplicar los métodos de la media, naive y drift**

- Alisado exponencial simple: se basa en modelos paramétricos deterministas que se ajustan a la evolución de la serie, a diferencia de los anteriores, que no dan más peso a los valores más recientes. Se utiliza sólo sobre series estacionarias. Depende de un parámetro alpha que modula la importancia que tiene sobre el

presente las observaciones pasadas. Su valor oscila entre 0 y 1. Si alpha toma un valor próximo a 0 las predicciones a lo largo de la serie son muy similares entre sí y se modifican poco con la nueva información. Si alpha toma un valor próximo a 1 la predicción se va adaptando al último valor observado.

En la siguiente figura podemos ver el resultado de aplicar un alisado exponencial simple a una serie estacionaria. Si nos fijamos en la línea roja podemos entender por qué se le llama "alisado". Las fluctuaciones del modelo son mucho menos bruscas que las de la serie original.

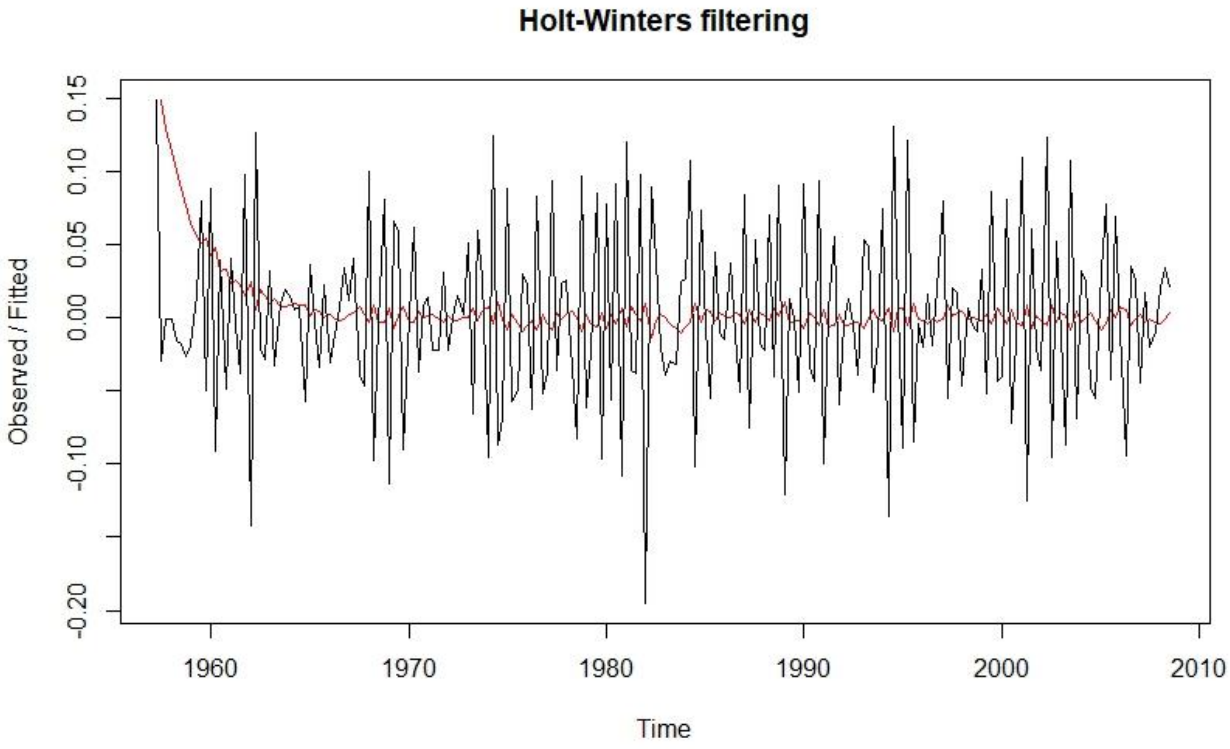



**Ilustración 3.11 Serie temporal transformada en estacionaria: Producción total cuatrimestral de cerveza en Australia. 1956-2008. Alisado exponencial simple**

## 3.5. Modelos ARIMA

Los modelos ARIMA(Autoregressive Integrated Moving Average) incluyen los modelos AR(Autoregressive), MA(Moving Averages) y ARMA(Autoregressive Moving Average).

Estos modelos proveen una nueva aproximación al problema de predicción de series temporales. El alisado exponencial y los modelos ARIMA son las dos aproximaciones más usadas a nivel global. (10) (5) (11)

La notación de los modelos ARIMA para series no estacionales es ARIMA(p,d,q), donde :

- p es el orden de los modelos autorregresivos AR(p).
- d es el número de diferencias regulares que convierten la serie en estacionaria. Será 0 si la serie ya es estacionaria.
- q es el orden de los modelos de medias móviles MA(p).

En caso de que la serie sea estacional debemos añadir nuevos parámetros de la siguiente forma: ARIMA(p,d,q)(P,D,Q). D es el número de diferencias estacionales necesarias para hacer la serie estacionaria. No nos vamos a detener en esto, ya que en nuestro trabajo procuraremos siempre pasar series estacionarias a los modelos.

En las etapas tempranas del modelado tenemos que tomar las siguientes decisiones:

- Tomar o no logaritmos para homogeneizar la varianza.
- Identificar el número de diferencias regulares que convierten la serie en estacionaria. Podemos hacerlo con los métodos que vimos anteriormente. Si la serie que le pasamos ya es estacionaria, este paso no es necesario.
- El orden de los polinomios autorregresivos y de medias móviles de la estructura regular.

Pero, ¿qué son los modelos autorregresivos y de medias móviles?

### 3.5.1. Modelos autorregresivos AR(p)

En los modelos autorregresivos AR(p) cada observación depende linealmente de las anteriores, y el número de observaciones de las que depende es el orden p.

$$Xt = \varphi 1Xt-1 + \varphi 2Xt-2 + \ldots + \varphi pXt-p + At$$

Siendo Xt la observación en tiempo t. Φp es un coeficiente de orden p que relaciona la observación en tiempo t con observaciones pasadas Xt-p. At es el ruido blanco en tiempo t.

### 3.5.2.Modelos de medias móviles MA(q)

En los modelos de medias móviles MA(q) cada observación depende linealmente de las perturbaciones anteriores, o dicho de otra manera, el nivel de cambio o innovación de una observación con respecto a las anteriores. El número de perturbaciones de las que depende es el orden q.

$$Xt = At - \theta 1At{-}1 - \theta 2At{-}2 - \ldots - \theta qAt{-}q$$

Xt es la observación en tiempo t. At es el ruido blanco en tiempo t. θq es un coeficiente de orden q que relaciona la observación en tiempo t con perturbaciones pasadas At-q.

### 3.5.3.Modelos ARMA(p,q)

Un modelo ARMA(p,q) es aquel en que cada observación depende linealmente de las p anteriores y las q últimas innovaciones. Es una generalización de los procesos AR(p) y MA(q).

$$Xt = \varphi 1Xt{-}1 + \varphi 2Xt{-}2 + \ldots + \varphi pXt{-}p - \theta 1At{-}1 - \theta 2At{-}2 - \ldots - \theta qAt{-}q + At$$

### 3.5.4.Modelos ARIMA(p,d,q)

Si combinamos diferenciación con autorregresión y medias móviles obtenemos un modelo ARIMA(p,d,q). El modelo puede ser escrito así:

$$Y't = C + \phi 1Y't{-}1 + ... + \phi pY't{-}p + \theta 1Et{-}1 + ... + \theta qEt{-}q + Et$$

Donde Y't es la series diferenciada. La parte de la derecha incluye los valores lageados de Yt y los errores lageados. Igual que antes, p es el orden de AR, q es el orden de MA, y d es el grado de diferenciación (0 si la serie es estacionaria).

La estimación de los parámetros es compleja. No suele ser posible deducir los valores de p y q que son más apropiados para los datos mirando sólo las gráficas. Para hacerlo manualmente tendríamos que visualizar las gráficas de autocorrelación(ACF) y autocorrelación parcial(PACF), pero este método sólo es fiable si p o q es 0. En la práctica se utilizan algoritmos de búsqueda para encontrar la solución óptima.

Si una serie está bien identificada, cuando se ajusta el modelo los residuos deben parecer ruido blanco. Un ruido blanco es una serie estacionaria, en la que ninguna observación depende de las otras.

Una vez tenemos el modelo, podemos extraer predicciones ejecutando métodos matemáticos sobre la fórmula de ARIMA(p,d,q), aislando las observaciones futuras.

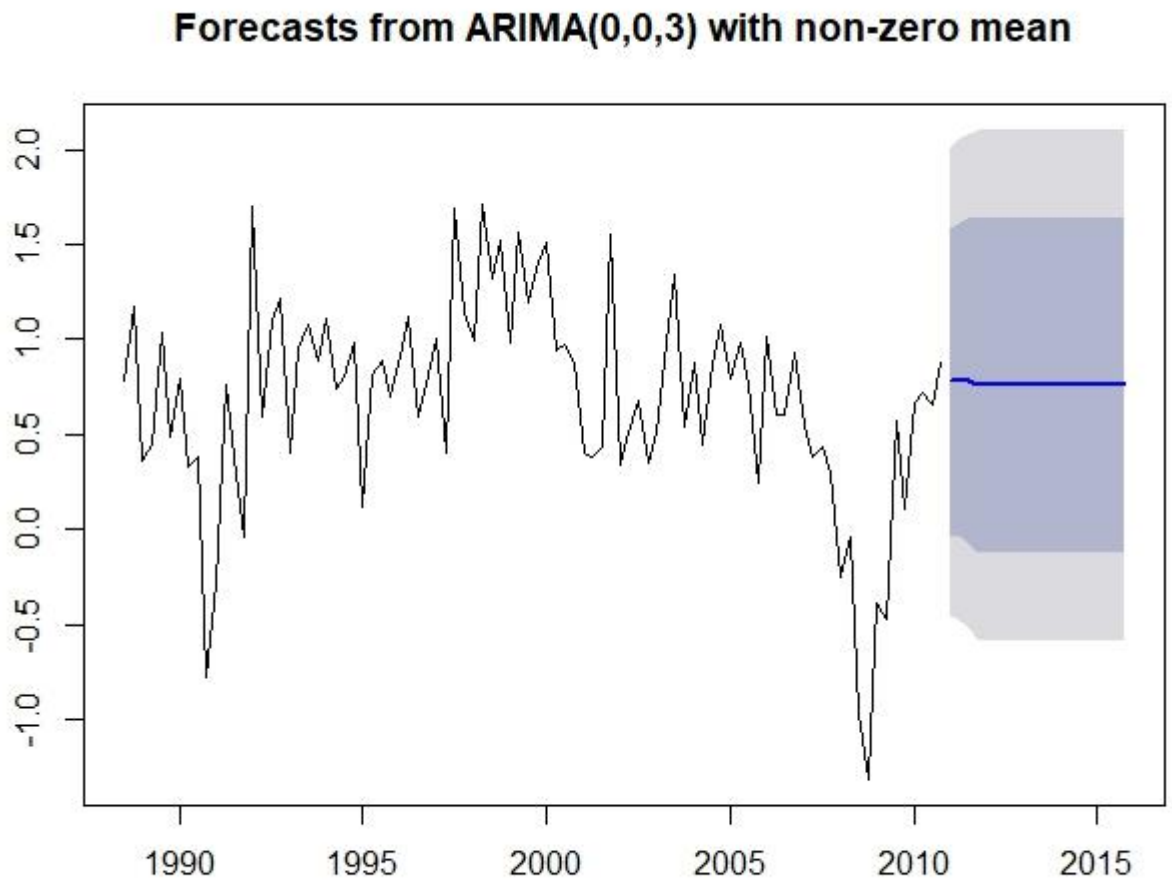



**Ilustración 3.12 Serie temporal: Ratio de crecimiento de consumo personal e ingresos personales. 1970-2010. Predicciones en azul con un modelo ARIMA(0,0,3)**

## 3.6. Minería de Datos

### 3.6.1. Conceptos generales

Llamamos Minería de Datos al proceso computacional de descubrir patrones en datasets muy grandes con métodos en la intersección de la inteligencia artifical, la estadística y los sistemas de bases de datos. Es un campo interdisciplinar de la informática. (12)

Estos patrones pueden verse como un resumen de los datos de entrada y pueden ser utilizados para tareas como aprendizaje automático o análisis predictivos.

A grandes rasgos podemos identificar dos grandes grupos de modelos de Minería de Datos: (13)

- Modelo predictivo: tiene como objetivo la predicción de nuevos valores a partir de datos ya conocidos. Incluye tareas como clasificación, regresión, análisis de series temporales y predicción.
- Modelo descriptivo: tiene como objetivo identificar patrones que expliquen o resuman los datos y sus relaciones. Así, es posible descubrir propiedades presentes en los datos actuales. Incluye técnicas como el agrupamiento, la extracción de reglas de asociación o el descubrimiento de correlaciones y secuencias.

Podemos resumir el proceso de Minería de Datos en los siguientes puntos: (14)

- Pre-procesamiento: la selección, transformación y limpieza de los datos de entrada suele ser la tarea más larga y tediosa del proceso. El dataset debe ser lo suficientemente grande como para hacer posible encontrar patrones mientras debe ser lo bastante conciso para poder ejecutar el algoritmo en un tiempo de ejecución razonable.

- Minería de Datos: contiene varios tipos de tareas. Las más comunes son: (4)

    - Detección de anomalías(outliers): identificación de recogidas de datos altamente inusuales para posterior análisis. Por ejemplo, pueden tratarse de errores de medición.
    - Reglas de asociación: búqueda de relaciones entre las distintas variables. Por ejemplo, la regla {cebollas,patatas}=>hamburguesa encontrada entre los datos de ventas de un supermercado indicaría que si un cliente compra cebollas y patatas es probable que también compren carne de hamburguesa.
    - Agrupamiento(clustering): trata de descubrir grupos y estructuras(clusters) que son de una forma u otra más similares entre sí que con el resto de los datos. Puede utilizarse, por ejemplo, en medicina para detectar un grupo de

pacientes con los mismos síntomas y por tanto probablemente con la misma afección.

- Clasificación: consiste en identificar a qué categoría, de entre un conjunto de categorías, pertenece una nueva observación. Por ejemplo, la asignación a un e-mail de la categoría de "spam" o "no-spam".
- Regresión: búsqueda de una función que modele la relación entre una variable dependiente y una o más independientes(predictoras). Predecir un futuro valor conociendo los anteriores en una serie temporal es una tarea de regresión.

- Validación de resultados: se comprueban que las conclusiones que arroja el modelo de Minería de Datos son válidas y lo bastante satisfactorias. Si se han realizado varios modelos, se buscará el que mejor se ajuste al problema. Si ninguno de los modelos alcanza los resultados esperados, debe alterarse alguno de los pasos anteriores para generar nuevos modelos. Esta comprobación se hace dividiendo el dataset de la siguiente manera: (15)
  - Conjunto de entrenamiento: subconjunto del dataset sobre los que se ejecutará el algoritmo de minería y en el que se buscan los posibles patrones.
  - Conjunto de test: subconjunto del dataset sobre el que no se ha aplicado minería y que sirve para evaluar el modelo entrenado. Se intentan buscar en este conjunto los mismos patrones que se encontraron en el de entrenamiento.

  Se mide la precisión del modelo con una o varias medidas de error que se obtienen al aplicar el modelo entrenado sobre el conjunto de test.

  El cómo dividir el dataset es una de las decisiones que habrá que enfrentar. Un método ampliamente utilizado que da buenos resultados es la validación cruzada.

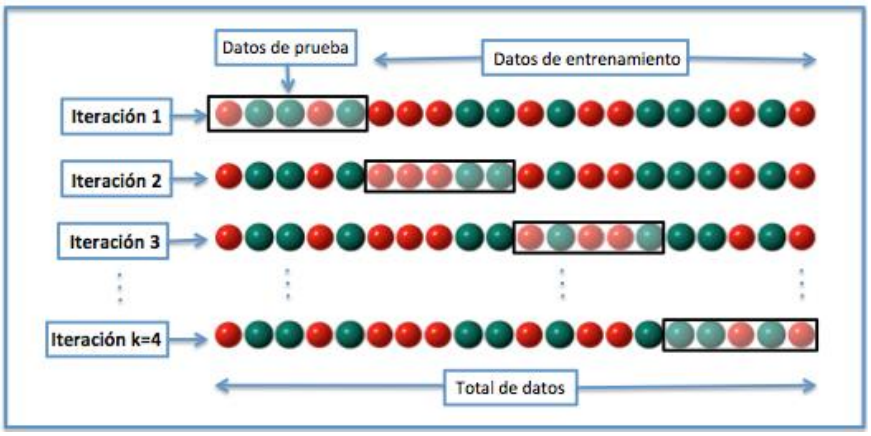



**Ilustración 3.13 Esquema explicativo de la validación cruzada clásica**

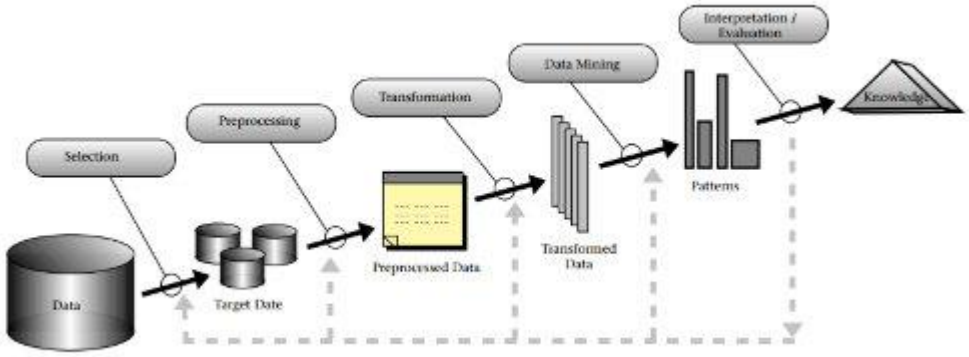



**Ilustración 3.14 Fases del proceso de Minería de Datos**

### 3.6.2.Minería de Datos aplicada a series temporales

La Minería de Datos para la predicción de series temporales consiste en construir un modelo predictivo utilizando alguna técnica de regresión.

Los métodos de regresión de Minería de Datos han demostrado tener una buena capacidad para realizar predicciones. Algoritmos como redes neuronales o sistemas basados en reglas difusas son aceptados como métodos de predicción aceptables para series temporales. (3)

Pero, ¿cómo adaptamos una serie temporal para hacer ejecutar sobre ella un algoritmo de Minería de Datos?

No resulta muy complicado. Para empezar, obviamente debemos adoptar los datos de la serie temporal a un formato que el algoritmo pueda leer. Pero tenemos que tener siempre en cuenta que lo que hace especial a las series temporales es que todos sus datos son consecutivos en el tiempo, lo que significa que una variable en un determinado instante depende de todas las variables que le preceden. Por tanto deben evitarse algunas técnicas típicas en minería como la validación cruzada que no tienen en cuenta la dependencia consecutiva de los datos. (5)

Ya que la validación es una parte fundamental del proceso de Minería de Datos, tenemos que encontrar un método alternativo para dividir los datos en entrenamiento y test.

### 3.6.3.Rolling forecast origin

Vamos a describir el método conocido como "rolling forecast origin", que podemos traducir como "predicción con origen móvil". Si llamamos k al mínimo número de observaciones para producir una predicción fiable, este proceso funciona de la siguiente manera: (5) (16)

- Seleccionar la observación en tiempo k + i para el conjunto de test, y usar las observaciones en 1, 2, … ,k + i – 1 para producir el modelo. Computar el error en la predicción para todos los datos con tiempo k + i.
- Repetir el paso anterior para i = 1, 2, … ,T – k donde T es el número total de observaciones.
- Computar las medidas de precisión de la predicción basadas en los errores obtenidos.

Básicamente siempre utilizamos como entrenamiento datos anteriores a los del test, y el test siempre será consecutivo al conjunto de entrenamiento. Además, los conjuntos van desplazándose a lo largo del tiempo hasta que el conjunto de test llega al final de la serie.

Como opción, el tamaño del conjunto de entrenamiento puede variar o no a lo largo del tiempo.

En la siguiente figura viene representada una variación de este proceso; en rojo el conjunto de entrenamiento, y en azul el de test. "Horizon" indica el tamaño del conjunto de

test, y “fixedWindow” indica si el tamaño del entrenamiento será siempre el mismo o se hará más grande a lo largo del tiempo(con el origen en el principio de la serie).

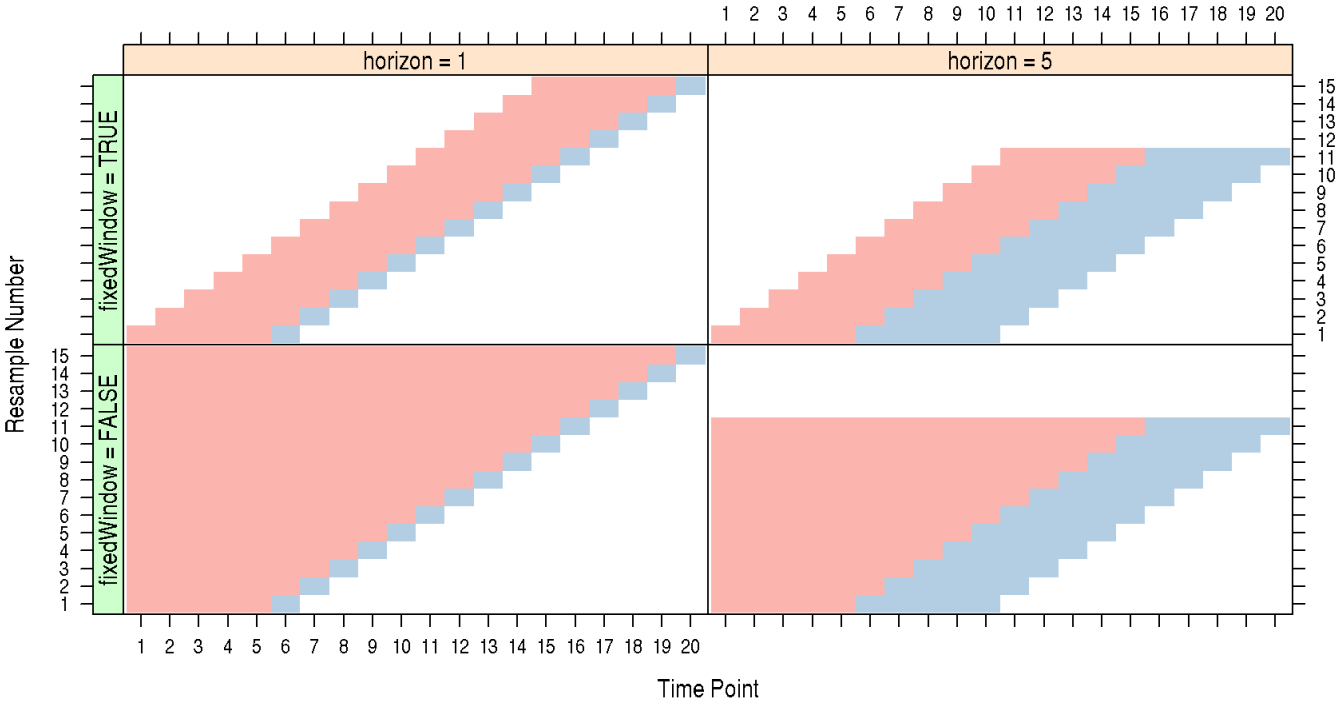



**Ilustración 3.15 Rolling forecast origin**

# 4. Diseño, análisis e implementación del paquete

En esta sección vamos a profundizar en el desarrollo del paquete.

En la fase de diseño hablaremos sobre técnicas de Ingeniería del Software que se han llevado a cabo, y se mostrarán los módulos y las funciones del paquete.

En la fase de implementación hablaremos sobre las herramientas utilizadas para el desarrollo del paquete, así como los requisitos necesarios para la construcción y la distribución del mismo.

## 4.1. Análisis de requisitos

Cada sistema basado en software tiene un propósito, usualmente expresado con algo que el sistema debe hacer. Un requerimiento es una característica del sistema o una descripción de algo que el sistema es capaz de hacer con el objeto de satisfacer el propósito del sistema. (17)

En otras palabras, los requisitos son lo que los clientes/usuarios esperan que haga el sistema.

El análisis consiste en listar una serie de requisitos que no dejen lugar a dudas sobre lo que debe hacer el sistema. Es algo que debemos hacer antes de empezar a desarrollar, ya que hay que tener claro todas y cada una de las funcionalidades que se esperan del software.

Vamos a llevar a cabo el análisis de dos tipos de requisitos.

### 4.1.1. Análisis de requisitos funcionales

Los requisitos funcionales describen la funcionalidad o los servicios que se espera que el sistema proveerá. Imponemos los siguientes:

- Proveer de herramientas en forma para el procesamiento y la predicción de series temporales.
- Incluir, en módulos separados y distinguidos, las distintas fases del proceso: pre-procesamiento, modelado, predicción y post-procesamiento.
- Incluir herramientas para la visualización en los módulos que lo permitan.
- Permitir la inspección de los resultados como salida de texto.

- Permitir crear modelos predictivos con los métodos de ARIMA y de Minería de Datos.
- Mostrar mensajes de error si se introducen de forma errónea los datos en todas las funciones.
- Incluir alguna herramienta para comparar de forma visual distintos modelos predictivos sobre la misma serie temporal.
- Facilitar la inclusión de nuevos algoritmos.

#### 4.1.2. Análisis de requisitos no funcionales

Los requisitos no funcionales no se refieren directamente a las funciones específicas del sistema, sino a las propiedades emergentes de este, como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento. Se nos presentan los siguientes:

- Permitir la posterior edición e inclusión de nuevos métodos en el sistema. El código debe ser claro y bien documentado para facilitar el trabajo de terceras personas sobre el paquete. La documentación y el código deben estar orientados al inglés, para hacer el paquete lo más global posible.
- El proceso de trabajo sobre el paquete debe ser claro y no dar lugar a confusión. Los nombres de las funciones y objetos deben describir a primera vista la funcionalidad de las mismas.
- El proceso debe ser lo más automático posible, de forma que un usuario no experto pueda hacer predicciones sin pararse a estudiar el significado de todos los parámetros de las funciones.
- Los módulos deben ser, cuando sea posible, independientes y flexibles, de forma que se pueda por ejemplo ir al módulo de modelado sin pre-procesar, o que se pueda utilizar el módulo de predicción sobre un modelo que ha procesado el propio usuario sin utilizar el módulo de modelado.
- Las salidas de texto deberán ser claras y no dar lugar a confusión.

### 4.2. Casos de uso

La técnica de los casos de uso nos permite entender y describir los requisitos. Los casos de uso son requisitos funcionales que ponen el acento en el uso del producto. (18)

Describen como debe comportarse el sistema desde el punto de vista del usuario, sin especificar cómo debe hacerlo internamente.

Se trata de un formato simple donde los usuarios y los desarrolladores pueden trabajar juntos, por lo que resulta un buen punto de partida para el diseño de software.

Identificamos las siguientes fases en el proceso:

- Pre-procesar.
- Modelar.
- Predecir.
- Post-procesar.
- CompararModelos.

A continuación vamos a representar un caso de uso para cada fase que hemos identificado en nuestro sistema. Los hemos creado con la siguiente estructura:

- Caso de uso: nombre del caso de uso
- Actor: quien interactuará con el sistema.
- Precondición: condición previa al proceso.
- Postcondición: estado tras el proceso.
- Secuencia normal: serie de pasos que habitualmente seguirá el usuario en el uso del sistema.
- Excepciones: mensajes de error posibles en los pasos de la secuencia normal.

| **Caso de uso** | **Pre-procesar** |
|---|---|
| **Actor** | Usuario |
| **Precondición** | El usuario tiene una serie temporal que quiere convertir en estacionaria. |
| **Postcondición** | El usuario obtiene una serie estacionaria. |
| **Secuencia normal** | 1. El usuario ejecuta la función automática de pre-procesamiento.<br>2. El usuario inspecciona el resultado en su salida de texto.<br>3. El usuario visualiza el resultado en una gráfica.<br>4. El usuario comprueba si la serie es estacionaria ejecutando tests.<br>5. Si la serie resulta no ser estacionaria, modifica los parámetros de la |

| | |
|---|---|
| | función de pre-procesamiento hasta que consigue una serie estacionaria. |
| **Excepciones** | 1. El usuario ha introducido un dato que no es una serie temporal.<br>1. El usuario ha introducido parámetros incorrectos en la función de pre-procesamiento. |

**Tabla 4.1 Caso de uso Pre-procesar**

| **Caso de uso** | **Modelar** |
|---|---|
| **Actor** | Usuario |
| **Precondición** | El usuario tiene una serie estacionaria. |
| **Postcondición** | El usuario obtiene un modelo predictivo. |
| **Secuencia normal** | 1. El usuario ejecuta la función de modelado indicando solamente el método predictivo que desea.<br>2. El usuario inspecciona el resultado en su salida de texto. |
| **Excepciones** | 1. El usuario ha introducido un método no válido. |

**Tabla 4.2 Caso de uso Modelar**

| **Caso de uso** | **Predecir** |
|---|---|
| **Actor** | Usuario |
| **Precondición** | El usuario tiene un modelo predictivo. |
| **Postcondición** | El usuario obtiene una serie de predicciones. |
| **Secuencia normal** | 1. El usuario ejecuta la función de predicción automática indicando el número de predicciones que desea.<br>2. El usuario inspecciona el resultado en su salida de texto.<br>3. El usuario visualiza el resultado en una gráfica, en la que se muestran la serie y las predicciones realizadas. |
| **Excepciones** | 1. El usuario introduce un valor por encima de 100 para el número de predicciones. |

**Tabla 4.3 Caso de uso Predecir**

| **Caso de uso** | **Post-procesar** |
|---|---|
| **Actor** | Usuario |

| | |
|---|---|
| **Precondición** | El usuario tiene una serie de predicciones sobre una serie temporal a la que aplicó un pre-procesamiento previo. |
| **Postcondición** | El usuario obtiene la serie original y las predicciones adaptadas. |
| **Secuencia normal** | 1. El usuario ejecuta la función automática de post-procesamiento introduciendo las predicciones y los datos del pre-procesamiento.<br>2. El usuario inspecciona el resultado en la salida de texto y comprueba que la serie coincide con la original.<br>3. El usuario visualiza el resultado en una gráfica. |
| **Excepciones** | 1. El usuario ha introducido parámetros incorrectos. |

**Tabla 4.4 Caso de uso Post-procesar**

| **Caso de uso** | **CompararModelos** |
|---|---|
| **Actor** | Usuario |
| **Precondición** | El usuario ha obtenido varios modelos con distintas técnicas y quiere compararlos. |
| **Postcondición** | El usuario concluye qué modelo predictivo es el mejor. |
| **Secuencia normal** | 1. El usuario lista por pantalla los errores de todos los modelos.<br>2. El usuario visualiza una comparativa de las predicciones de los modelos en una gráfica. |
| **Excepciones** | 1. El usuario ha introducido menos de dos o más de cinco modelos distintos. |

**Tabla 4.5 Caso de uso CompararModelos**

### 4.2.1.Diagrama de casos de uso

Con estos casos de uso podemos pintar un diagrama en el que visualizamos la interacción del usuario con el sistema.

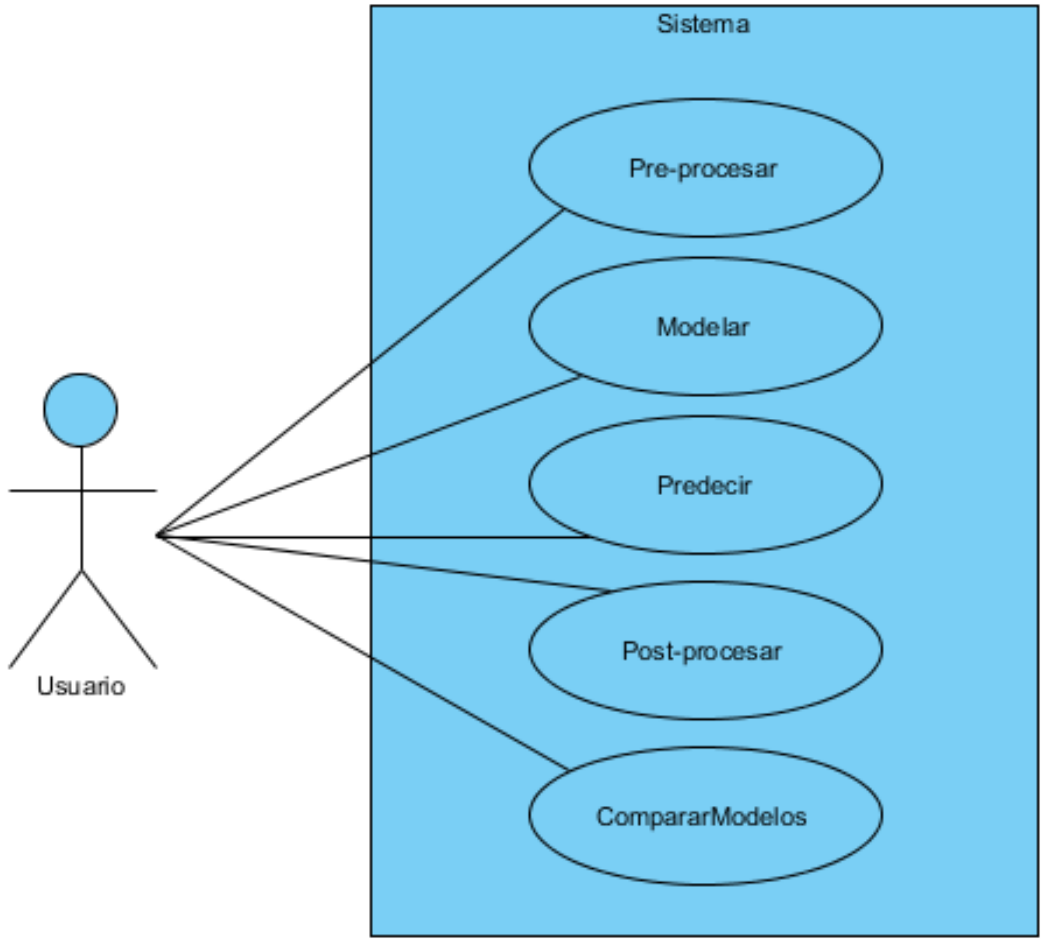



**Ilustración 4.1 Diagrama de casos de uso**

## 4.3. Diseño

En la fase de diseño se decide estructurar la biblioteca en cuatro módulos que se corresponden con cuatro de las cinco tareas identificadas en el apartado anterior:

- Pre-procesamiento.
- Modelado.
- Predicción.
- Post-procesamiento.

En cuanto a la quinta tarea, la visualización de la comparación de modelos quedará implementada en una sola función contenida en el módulo de predicción. Las medidas de error serán accesibles fácilmente en la fase de modelado.

Los módulos serán grupos de funciones y clases, contenidos cada uno en un archivo diferente y con el nombre del módulo. Además, todos los nombres de las funciones contenidas en un mismo módulo empezarán con la misma palabra clave, que corresponderá al nombre del módulo.

El objetivo de esta estructuración es hacer más sencilla la comprensión del proceso, separando las distintas fases de forma que queden bien diferenciadas. Esto también facilitará la edición del código en un futuro.

A continuación se explicará con más detalle la funcionalidad de cada módulo y algunos métodos que proporcionan.

#### 4.3.1.Módulo de pre-procesamiento

En esta fase el objetivo es aplicar transformaciones a una serie temporal para eliminar patrones que perjudiquen el rendimiento de los modelos predictivos. En otras palabras, convertir la serie en estacionaria.

Estos patrones que perjudican a los modelos son la tendencia, la estacionalidad y la heterogeneidad de la varianza. En este módulo se integran algunos métodos para eliminarlos, siempre dejando la puerta abierta a la implementación futura de nuevos métodos:

- Para homogeneizar la varianza, se ha implementado la transformación logarítmica y la transformación Box-Cox.
- Para eliminar la tendencia, se ha implementado la diferenciación regular y la sustracción de medias estacionales.
- Para eliminar la estacionalidad, se ha implementado la diferenciación estacional.

También se han añadido el test Augmented Dickey-Fuller y la visualización del gráfico de autocorrelación para verificar la estacionariedad de la serie.

Con el objetivo de agilizar y automatizar el proceso, además de estas funciones implementamos una clase que recoja toda la información sobre las transformaciones que se han aplicado a la serie. Esto se hace necesario para el posterior post-procesamiento de los datos, ya que después de obtener las predicciones querremos volver a ver la serie temporal en su forma original.

### 4.3.2.Módulo de modelado

En esta fase el objetivo es construir un modelo predictivo a partir de una serie temporal.

Existe una gran variedad de modelos que podemos aplicar a una serie. Algunos de ellos se han explicado anteriormente en esta memoria.

En esta primera versión implementamos el modelo ARIMA, que se trata de uno de los modelos más populares y más eficaces. También integraremos la construcción de modelos con algoritmos de regresión de Minería de Datos.

Se ha implementado una clase que facilita el modelado. Esta clase se ocupará de ejecutar las tareas necesarias para construir el modelo que se le pase como parámetro. También almacenará la información sobre el modelo predictivo, que será tanto ilustrativa como necesaria para la siguiente fase de predicción.

### 4.3.3.Módulo de predicción

Una vez tenemos el modelo predictivo podemos usarlo para calcular predicciones.

En este módulo implementamos el cálculo de predicciones para modelos ARIMA y los modelos de regresión de Minería de Datos.

Se ha implementado además una clase para la predicción, que automatiza el proceso de cálculo y contiene como atributos las predicciones y la serie temporal sobre la que se han hecho.

En este apartado también creamos la función para comparara modelos de forma visual, pasándole como parámetros la serie original y las predicciones de los distintos modelos.

### 4.3.4.Módulo de post-procesamiento

Esta fase es de vital importancia para obtener datos realistas después de predecir sobre una serie convertida en estacionaria.

Utilizando un objeto predicción y un objeto pre-procesamiento (que recordemos que contiene todas las operaciones realizadas en la transformación de la serie), la función de post-procesamiento deshace todo las transformaciones realizadas en la fase de pre-procesamiento tanto a la serie temporal como a la predicciones. Devuelve un objeto de tipo predicción.

Esta función llama a otras funciones que se corresponden en nombre con los métodos implementados en el módulo de pre-procesamiento (cambiando la primera palabra correspondiente al módulo), y que tienen como objetivo revertir la transformación correspondiente.

### 4.3.5. Diagrama de clases

El lenguaje de programación utilizado(R) no es propiamente orientado a objetos, aunque sí que permite la creación de objetos. En nuestro paquete las funciones no pertenecen a una clase si no que están englobadas en módulos. Por tanto el diagrama de clases no ayudará a describir de forma exhaustiva el paquete, pero sí nos permitirá indicar las relaciones entre los módulos que forman parte de él.

En la biblioteca implementamos tres clases: prep, modl y pred. Cada clase pertenece a su módulo homónimo. Además tenemos el módulo postp que no es una clase si no que provee funciones, por lo que en el diagrama se dibujará como una interfaz.

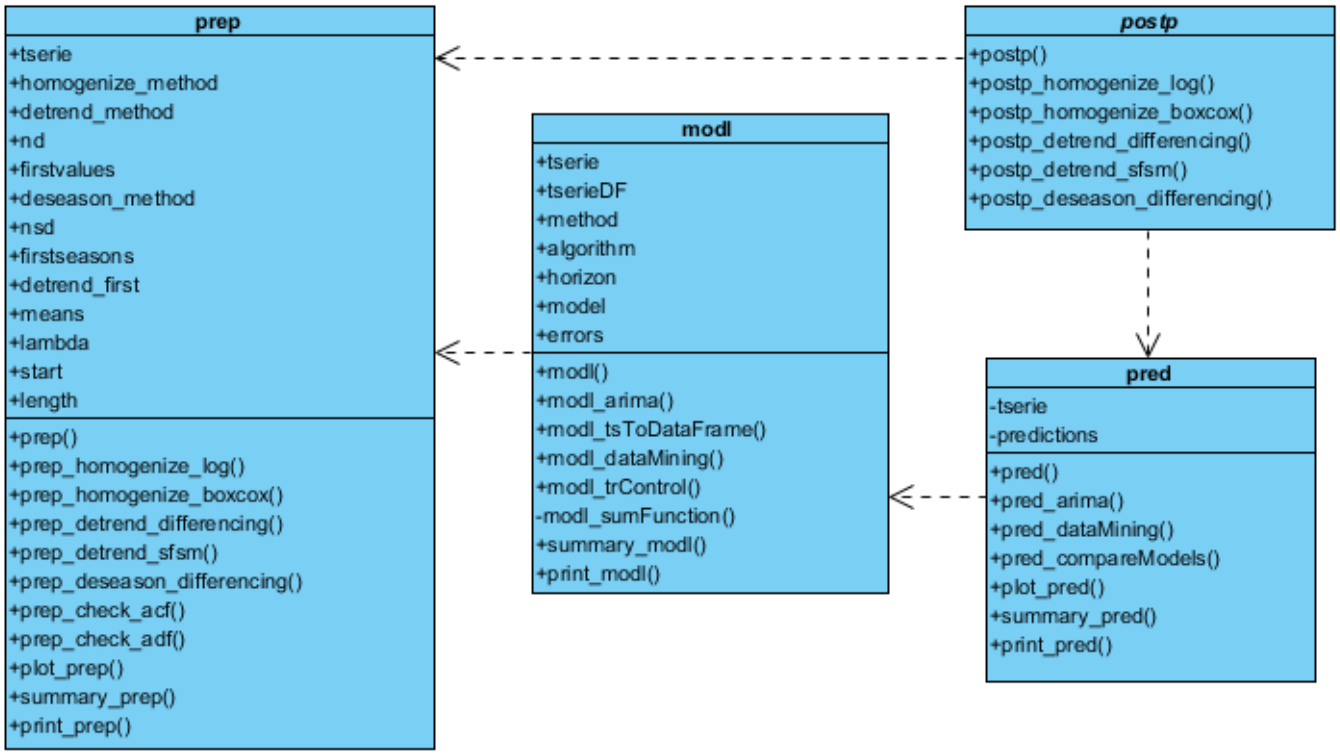



**Ilustración 4.2 Diagrama de clases**

En la práctica el comportamiento de estos módulos es similar al de clases, y por eso me decanté por utilizar este diagrama que describe muy detalladamente la estructura de la biblioteca y las relaciones entre los módulos.

## 4.4. Implementación

En este apartado hablaremos de los retos y las tareas que han ido surgiendo en la elaboración del paquete, del que ya hemos visto el diseño.

Se explicará el lenguaje de programación utilizado, el entorno de desarrollo y el proceso de documentación y distribución.

### 4.4.1. Lenguaje de programación

El lenguaje de programación utilizado para desarrollar este proyecto es R.

R es un lenguaje de código abierto orientado a estadística computacional y gráficos respaldado por la R Foundation for Statistical Computing. Es ampliamente utilizado en campos como la estadística o la Minería de Datos para el desarrollo de software estadístico y análisis de datos. La popularidad de R ha ido en aumento en los últimos años. (19) (20) (21)

**Ilustración 4.3 Logotipo de R**

R proporciona un amplio abanico de herramientas estadísticas y gráficas, incluyendo modelado lineal y no lineal, tests estadísticos clásicos, análisis de series temporales, clasificación, y muchos otros más.

Podemos extender el número de herramientas que nos proporciona R a través de un repositorio público de paquetes elaborados por los usuarios para áreas de estudio concretas. Existe una gran comunidad detrás de este lenguaje, pero hablaremos de eso más tarde.

R es un intérprete, lo que significa que al ejecutar un código obtenemos directamente el resultado sin compilación previa. Es, además, un híbrido entre la programación procedural y la programación orientada a objetos.

En R, de forma similar a Python, todos las variables son tratadas como objetos. Esto significa que no podremos escribir directamente en memoria, como en el caso de C++, lo que hace el lenguaje más sencillo pero menos eficiente. Cuando almacenamos cualquier variable, ya sea numérica, de tipo carácter o lógica R la tratará como a un vector.

```
> test <- 5
> test[1]
[1] 5
> length(test)
[1] 1
```

**Ilustración 4.4 Ejemplo: R trata todas las variables como vectores**

Puede parecer una ventaja a primera vista, pero es necesario conocer y tener en cuenta este hecho mientras se trabaja en R. La escasez de restricciones puede llegar a resultar una mala aliada cuando programamos.

Existen tres métodos distintos para crear una clase: S3, S4 y Reference:

- S3: es primitivo por naturaleza. Carece de definición formal y objetos de esta clase se crean simplemente añadiéndoles un atributo de clase. Esta sencillez explica el hecho de que sea ampliamente utilizada en R; de hecho, la mayoría de las clases son de este tipo.

```
> s <- list(name="Alberto",age="26")
> class(s)
[1] "list"
> class(s) <- "student"
> class(s)
[1] "student"
```

**Ilustración 4.5 Ejemplo de clase S3**

- S4: es una mejora del S3. Tienen una estructura formal definida lo que fuerza a objetos de la misma clase a ser similares. Se definen usando la función setClass(), y se crean nuevos objetos con la función new().

```
> setClass("student", slots=list(name="character", age="numeric"))
> s <- new("student",name="Alberto", age=26)
> s
An object of class "student"
Slot "name":
[1] "Alberto"

Slot "age":
[1] 26
```

**Ilustración 4.6 Ejemplo de clase S4**

- Reference: es más similar a lo que estamos acostumbrados a ver en los lenguajes orientados a objetos más comunes.

```
> student <- setRefClass("student",
+         fields = list(name = "character", age = "numeric"))
> s <- student(name="Alberto", age=26)
> s
Reference class object of class "student"
Field "name":
[1] "Alberto"
Field "age":
[1] 26
```

**Ilustración 4.7 Ejemplo de clase Reference**

Para las clases de nuestro paquete se decidió utilizar el enfoque S3, por ser el más sencillo y ampliamente utilizado. Además, no íbamos a necesitar clases complejas, para cuyo caso quizá hubiera sido preferible utilizar uno de los otros dos. S3 nos aporta más flexibilidad y sencillez y en contrapartida exige un mayor conocimiento del lenguaje y un especial cuidado con su manejo.

R proporciona una interfaz para trabajar con línea de comandos, pero existen buenos entornos de desarrollo disponibles.

### 4.4.2.Entorno de desarrollo

Para desarrollar el paquete utilizamos el entorno de desarrollo para R llamado RStudio.

En cuanto a su interfaz incluye una consola, un editor de textos que facilita la creación de scripts y su ejecución al instante, un entorno global que nos muestra las funciones y variables almacenadas en memoria, un menú de navegación de ficheros y una ventana para la visualización de gráficas, entre otras cosas.

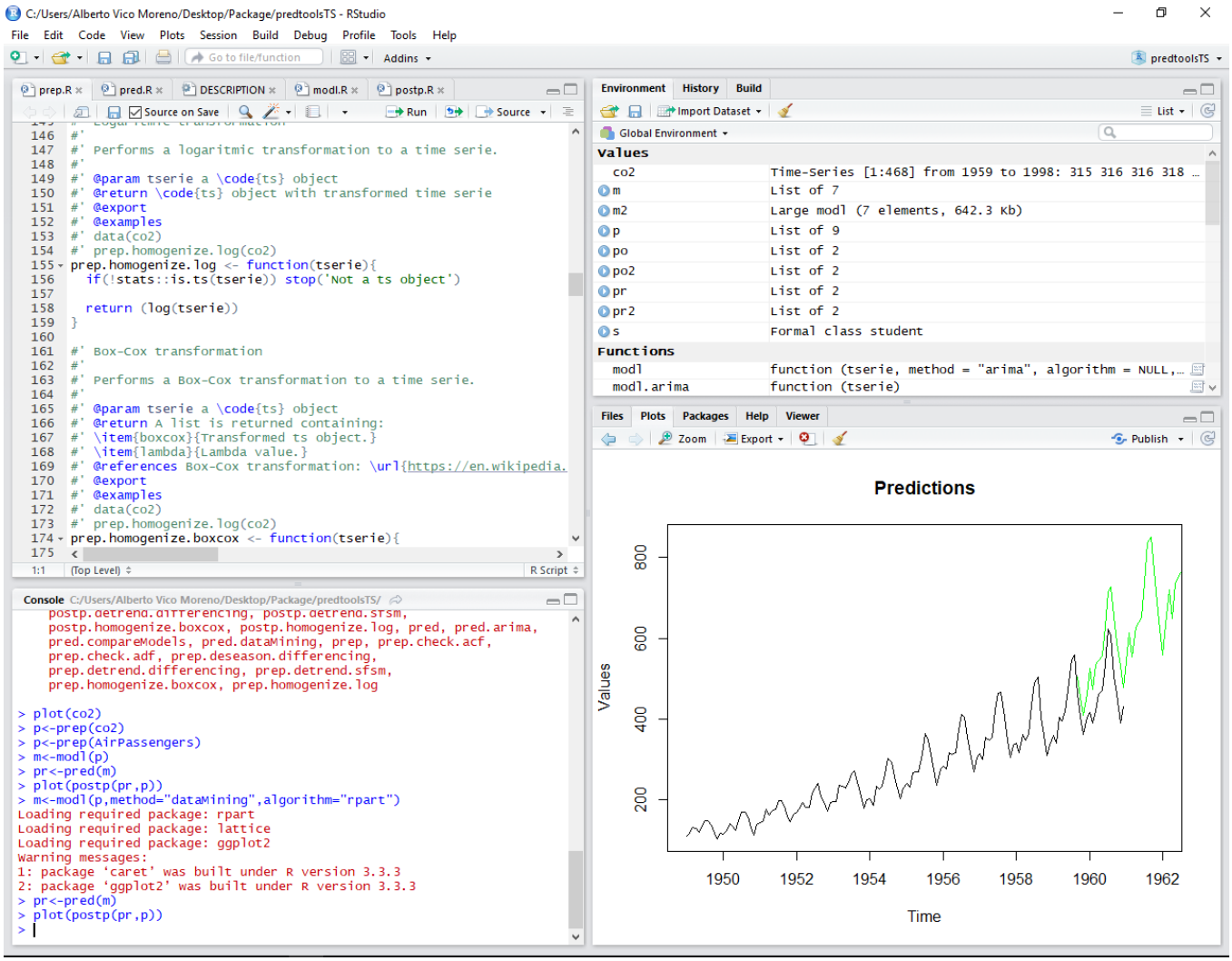



**Ilustración 4.8 Interfaz de usuario de RStudio**

Esta herramienta facilita mucho el flujo de trabajo con R, pero su característica más importante y que me hizo decidirme por ella es la ayuda que proporciona al crear un paquete. Se trata de un proceso realmente tedioso sin las opciones que nos aporta. (22)

### 4.4.3. Bibliotecas utilizadas

Nuestro código hace uso de una buena cantidad de bibliotecas externas proporcionadas por otros usuarios. El CRAN es una red de servidores ftp y web alrededor del mundo que

almacena paquetes creados por los usuarios y evaluados por cierto número de expertos voluntarios. [1]

Las bibliotecas más relevantes que se han utilizado son las siguientes:

- Stats: contiene diversas funciones para cálculos estadísticos. Es la que nos proporciona la estructura de datos "ts" donde almacenamos las series temporales.
- Forecast: métodos y herramientas para procesar y analizar predicciones sobre series temporales univariables. Lo utilizamos para diversas tareas en el pre-procesamiento, y para el modelado de ARIMA. (23)
- Caret: proporciona herramientas para utilizar modelos de clasificación y de regresión de Minería de Datos. Recopila e integra una amplia gama de algoritmos. También se ocupa de la división de los datos en entrenamiento y test. Lo utilizamos en la etapa de modelado y de predicción de los modelos de Minería de Datos. (16)

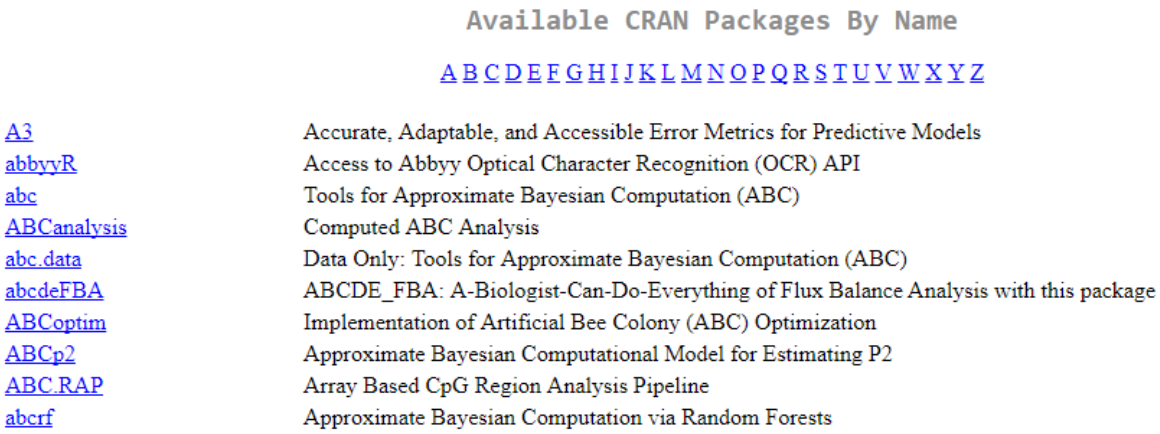
#### 4.4.4.Documentación, construcción y distribución del paquete

El objetivo final del proyecto es crear una biblioteca lo bastante útil y sólida como para distribuirla en CRAN, de forma que otros usuarios puedan utilizarla e incluso mejorarla.

Una vez tenemos el código fuente implementado, es hora de crear el paquete.

El primer paso es encontrar un nombre adecuado. El nombre es importante; idealmente debe ser capaz de definir la finalidad de la biblioteca en un vistazo. Debemos elegir un nombre que no esté en uso, lo cual podemos comprobar en el archivo de paquetes de CRAN. (24)

[1] https://cran.r-project.org/



Available CRAN Packages By Name

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

| | |
|---|---|
| A3 | Accurate, Adaptable, and Accessible Error Metrics for Predictive Models |
| abbyyR | Access to Abbyy Optical Character Recognition (OCR) API |
| abc | Tools for Approximate Bayesian Computation (ABC) |
| ABCanalysis | Computed ABC Analysis |
| abc.data | Data Only: Tools for Approximate Bayesian Computation (ABC) |
| abcdeFBA | ABCDE_FBA: A-Biologist-Can-Do-Everything of Flux Balance Analysis with this package |
| ABCoptim | Implementation of Artificial Bee Colony (ABC) Optimization |
| ABCp2 | Approximate Bayesian Computational Model for Estimating P2 |
| ABCRAP | Array Based CpG Region Analysis Pipeline |
| abcrf | Approximate Bayesian Computation via Random Forests |

**Ilustración 4.9 Repositorio de paquetes de CRAN**

Nos decantamos por el siguiente nombre: predtoolsTS. Significa “herramientas de predicción para series temporales”, lo que engloba bastante bien el contenido y llamaría la atención de los usuarios interesados.

Una parte fundamental en la construcción y distribución del proyecto es la documentación. Ya que vamos a distribuir el paquete de forma global, es importante que tanto el código como la documentación estén en inglés. Utilizaremos el paquete roxygen2, que está inspirado en el Doxygen de C++. (25) (22)

Roxygen2, en combinación con RStudio, nos ayuda a automatizar el proceso de la creación del paquete. Eso sí, la documentación de cada una de las funciones del paquete debe ser exhaustiva y sin errores.

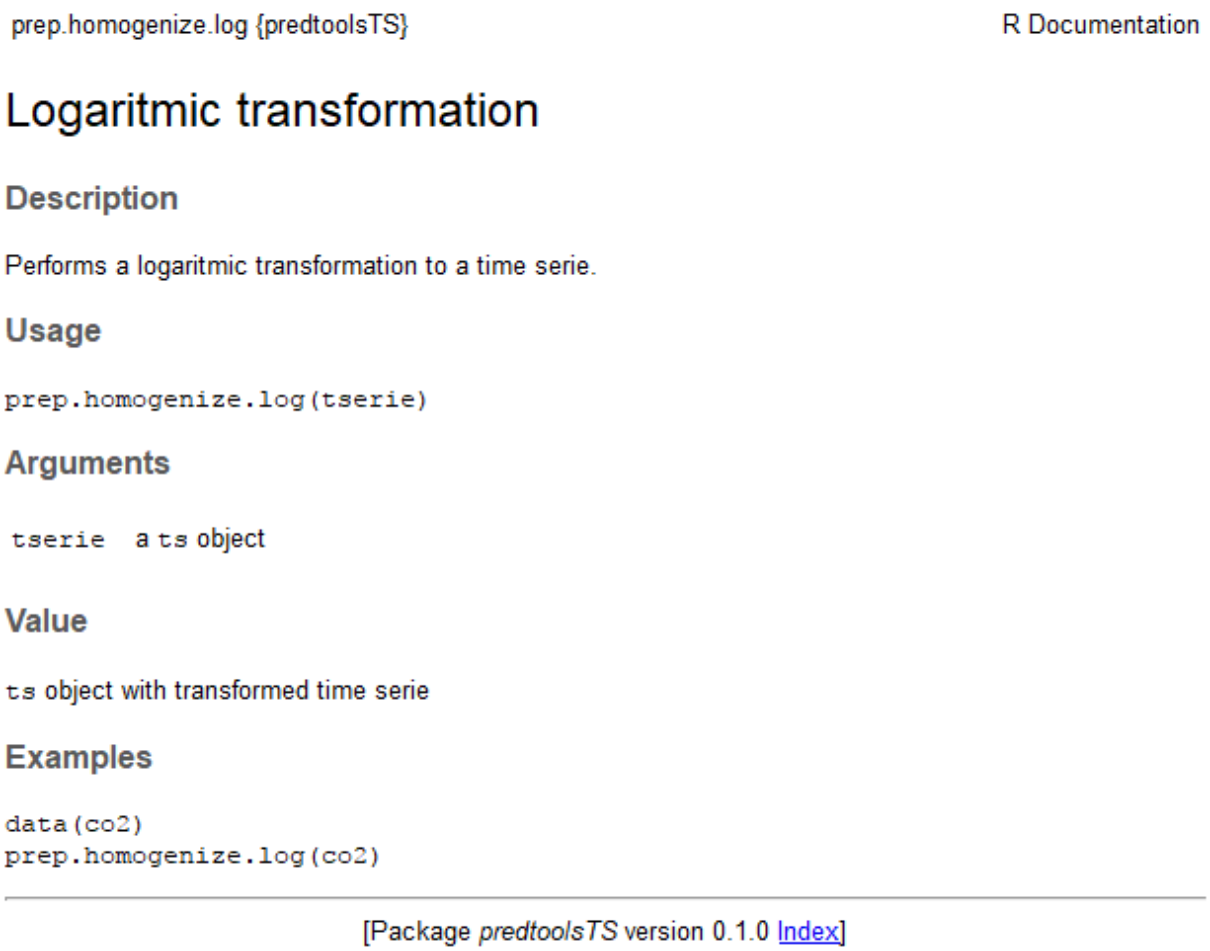
```
#' Logaritmic transformation
#'
#' Performs a logaritmic transformation to a time serie.
#'
#' @param tserie a \code{ts} object
#' @return \code{ts} object with transformed time serie
#' @export
#' @examples
#' data(co2)
#' prep.homogenize.log(co2)
prep.homogenize.log <- function(tserie){
  if(!stats::is.ts(tserie)) stop('Not a ts object')

  return (log(tserie))
}
```

**Ilustración 4.10 Ejemplo de función documentada con roxygen2**

Pone a nuestra disposición una buena cantidad de opciones. Un elemento a destacar es la opción @examples, que nos permite poner ejemplos de la ejecución de la función. Poner ejemplos aporta atractivo a la biblioteca, pero hay que tener cuidado porque tanto RStudio como CRAN comprueban que todos los ejemplos del paquete funcionen.

La documentación de una función se transformará a posteriori en la página de ayuda de dicha función, a la que se accede con el comando "?funcion".



prep.homogenize.log {predtoolsTS} R Documentation

# Logaritmic transformation

## Description

Performs a logaritmic transformation to a time serie.

## Usage

```
prep.homogenize.log(tserie)
```

## Arguments

| | |
|---|---|
| `tserie` | a `ts` object |

## Value

`ts` object with transformed time serie

## Examples

```
data(co2)
prep.homogenize.log(co2)
```

[Package *predtoolsTS* version 0.1.0 Index]

**Ilustración 4.11 Página de ayuda de la función prep.homogenize.log()**

Una vez tenemos el código bien documentado, podemos empezar a crear el paquete con RStudio a partir del código fuente. El entorno de desarrollo creará para nosotros la estructura de ficheros necesaria. (22) (26) (27) (28) (29) (24)

La estructura básica de ficheros es la siguiente:

- Carpeta “R”, donde se almacena el código fuente.
- Carpeta “man”, donde se almacenan los archivos .Rd. Existe uno para cada función de la biblioteca, y están escritos en un formato especial basado en LaTeX. Se generan automáticamente a partir de la documentación de roxygen2. Cuando llamamos a la ayuda sobre una función en R, se lee el archivo .Rd y se interpreta, convirtiéndolo en html para mostrarlo.
- Archivo “DESCRIPTION”, que almacena los metadatos de la biblioteca. CRAN exige que para distribuir un paquete este archivo esté completo. Incluye datos de contacto del autor, una descripción, la versión y una lista de paquetes externos que se utilizan en el nuestro, entre otras cosas. Este archivo sí debemos editarlo y completarlo.
- Archivo “NAMESPACE”, que se encarga del espacio de nombres. En él se especifican las funciones de la biblioteca que se exportan. También controla qué funciones de paquetes externos son utilizadoa en tu código. Roxygen2 se encarga de editar este archivo, así que lo mejor es no tocarlo.

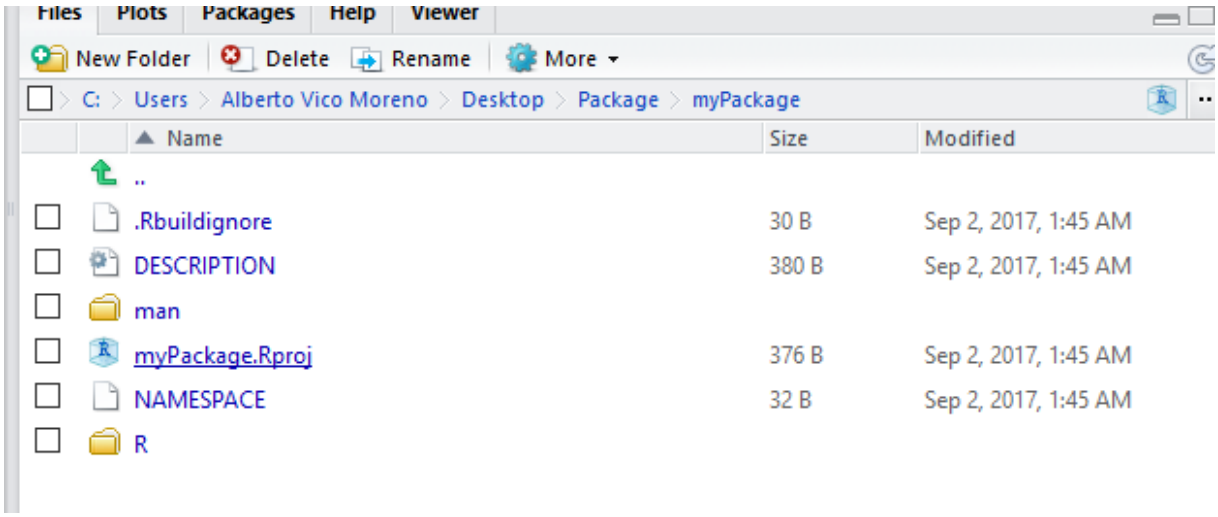



**Ilustración 4.12 Estructura de archivos creada por RStudio**

Por último antes de construir el paquete es recomendable que hagamos uso de la herramienta “Check” de RStudio, que realiza un chequeo completo al correcto funcionamiento de toda la biblioteca, incluyendo la ejecución de los ejemplos. Se genera un log en el que se nos listan errores, alertas y notas. Es recomendable resolver no sólo los

errores si no también las alertas y las notas. CRAN también ejecutará un chequeo sobre el paquete, y automáticamente lo descartará si encuentra algún aviso.

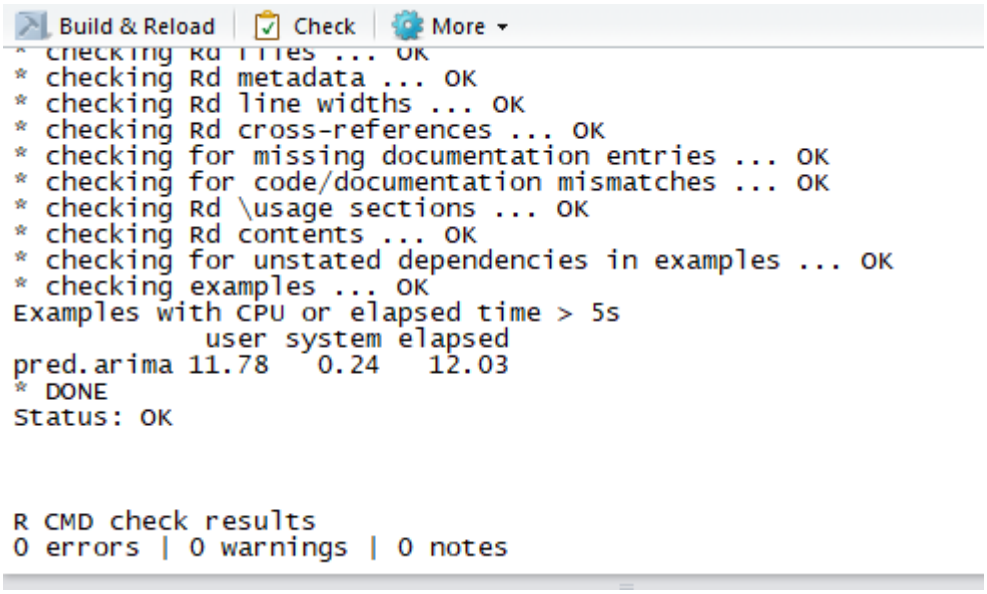


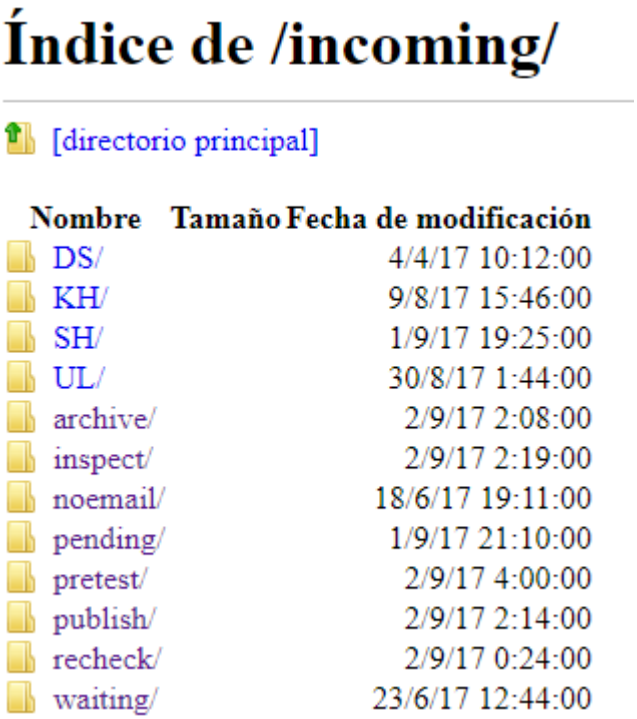

**Ilustración 4.13 Log resultado del chequeo del paquete en RStudio sin errores**

Ya podemos guardar el paquete en forma de código fuente. El siguiente paso es la distribución del paquete.

CRAN tiene una larga serie de políticas y estándares que debemos tener en cuenta durante todo el proceso de elaboración, documentación y creación del paquete. Un paquete que no cumpla todos los estándares tiene menos probabilidades de acabar en el repositorio de paquetes.

Para subir el paquete rellenamos un formulario con nuestro e-mail y una breve descripción del paquete.[2] Una vez lo subimos queda en un directorio ftp con varias carpetas, que indican la fase de evaluación en la que se encuentran los paquetes. Al subirlo por primera vez, se almacena en “pretest/”.

[2] https://cran.r-project.org/submit.html



Índice de /incoming/

[directorio principal]

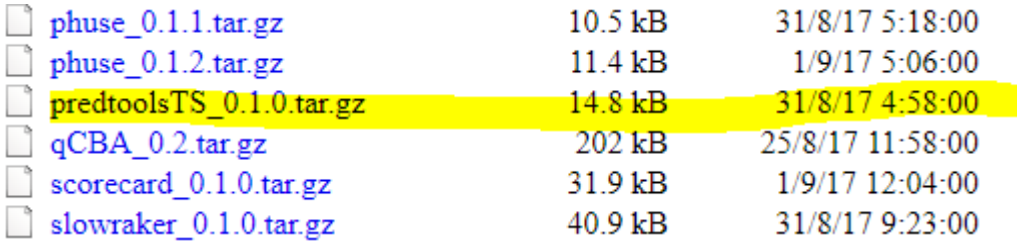
| Nombre | Tamaño | Fecha de modificación |
|---|---|---|
| DS/ | | 4/4/17 10:12:00 |
| KH/ | | 9/8/17 15:46:00 |
| SH/ | | 1/9/17 19:25:00 |
| UL/ | | 30/8/17 1:44:00 |
| archive/ | | 2/9/17 2:08:00 |
| inspect/ | | 2/9/17 2:19:00 |
| noemail/ | | 18/6/17 19:11:00 |
| pending/ | | 1/9/17 21:10:00 |
| pretest/ | | 2/9/17 4:00:00 |
| publish/ | | 2/9/17 2:14:00 |
| recheck/ | | 2/9/17 0:24:00 |
| waiting/ | | 23/6/17 12:44:00 |

**Ilustración 4.14 Directorio de CRAN donde se almacenan los paquetes sometidos a evaluación**

La revisión de los paquetes la llevan a cabo una serie de usuarios voluntarios expertos; ellos comprueban la calidad del paquete, y lo van moviendo entre las carpetas del directorio.

En caso de que el paquete sea rechazado, se pondrían en contacto con el autor via e-mail explicándole los motivos.

Nuestro paquete se encuentra actualmente en la carpeta “archive/”, después de haber pasado los primeros filtros. El tiempo que pueden tardar en evaluar un paquete es impredecible.



| | | |
|---|---|---|
| phuse_0.1.1.tar.gz | 10.5 kB | 31/8/17 5:18:00 |
| phuse_0.1.2.tar.gz | 11.4 kB | 1/9/17 5:06:00 |
| predtoolsTS_0.1.0.tar.gz | 14.8 kB | 31/8/17 4:58:00 |
| qCBA_0.2.tar.gz | 202 kB | 25/8/17 11:58:00 |
| scorecard_0.1.0.tar.gz | 31.9 kB | 1/9/17 12:04:00 |
| slowraker_0.1.0.tar.gz | 40.9 kB | 31/8/17 9:23:00 |

**Ilustración 4.15 Paquete predtoolsTS a la espera de evaluación**

# 5. Manual de usuario

## 5.1. Manual de instalación

Para la instalación de R en un equipo lo mejor es seguir las instrucciones en la web oficial CRAN. [3]

No es necesario para instalar un paquete pero si queremos un entorno de desarrollo, instalar RStudio facilita la instalación de paquetes y cualquier trabajo que hagamos sobre R. Lo conseguimos en su página oficial. Hay que prestar atención a las instrucciones ya que tendremos que modificar variables de entorno. [4]

Para instalar un paquete en una sesión de R utilizaremos la consola de comandos que nos proporciona. Existen dos formas de hacerlo:

- De forma remota: accede al repositorio de paquetes de CRAN a través de Internet. De momento la biblioteca predtoolsTS está en período de evaluación así que aún no se encuentra en este repositorio. Por ejemplo, si queremos instalar el paquete forecast, ejecutamos la orden:
  - install.packages("forecast")
- Desde un archivo local: si tenemos el paquete en nuestro equipo. Por convenio, el código fuente de un paquete de R debe estar comprimido en un archivo .tar.gz. Para instalar nuestro paquete, ejecutamos de igual forma:
  - install.packages("carpeta_del_archivo\\predtoolsTS_0.1.0.tar.gz", type="source)

  Si estamos trabajando en Windows debemos tener cuidado de usar la doble barra \\ al especificar ubicaciones de archivos en R.

En RStudio, podemos instalar paquetes a través del menú Tools << Install packages. Se nos da a elegir si queremos cargar una biblioteca local o remota.

De esta forma queda instalada la biblioteca, pero aún no está cargada en la sesión de trabajo actual. Para ello ejecutamos:

[3] https://cran.r-project.org/
[4] https://www.rstudio.com/

- library(predtoolsTS)

A partir de este momento estarán a nuestra disposición todas las funciones del paquete.

```
> p <- prep(co2)
Error: could not find function "prep"
> install.packages("C:/Users/Alberto Vico Moreno/Desktop/Package/predtoolsTS_0.1.0.ta
r.gz", repos = NULL, type = "source")
Installing package into 'C:/Users/Alberto Vico Moreno/Documents/R/win-library/3.3'
(as 'lib' is unspecified)
* installing *source* package 'predtoolsTS' ...
** R
** preparing package for lazy loading
** help
*** installing help indices
** building package indices
** testing if installed package can be loaded
* DONE (predtoolsTS)
> p <- prep(co2)
Error: could not find function "prep"
> library(predtoolsTS)
> p <- prep(co2)
> summary(p)
Preprocessed time series object
~Homogenizing method: logarithmic transformation
~Detrending method: differencing
 Number of differences:  1
First original values:  5.753905
~No deseason performed
~Original serie start:  1959 1
~Original serie length:  468
~Preprocessed time serie:
 Time-Series [1:467] from 1959 to 1998: 0.002818 0.0006 0.003344 0.001793 -0.000409 .
```

**Ilustración 5.1 Proceso de instalación de un paquete por línea de comandos**

## 5.2. Manual de uso

### 5.2.1. Introducción al manual

A lo largo de este manual exploraremos todas las funcionalidades y el flujo de trabajo más adecuado para trabajar con el paquete predtoolsTS.

Este paquete fue creado con el objetivo de condensar distintos métodos para el análisis de series temporales y su posterior predicción.

Actualmente cuenta con soporte para los modelos ARIMA y para modelos de regresión de Minería de Datos.

El paquete se divide en cuatro módulos, que coinciden con distintas fases del proceso de modelado de series temporales:

- prep: módulo de pre-procesamiento.
- modl: módulo de modelado.
- pred: módulo de predicción.
- postp: módulo de post-procesamiento.

La notación elegida para las funciones, siguiendo los estándares de R, es "nombreMódulo.nombreFunción".

Los tres primeros módulos contienen una clase cada uno con el mismo nombre que el módulo al que pertenecen. Se han implementado funciones genéricas para todas ellas. Se detallarán estas funciones en la descripción de estas clases.

Este paquete se construyó intentando dar un enfoque lo más automático posible al proceso. Por eso muchos de los parámetros en las funciones tienen valores por defecto, de forma que pueden indicarse o no.

A continuación se exploran en detalle las funcionalidades de cada uno de los módulos.

### 5.2.2.Módulo prep

El objetivo de este módulo es transformar una serie previamente al modelado de la misma para eliminar patrones que influyan negativamente en la predicción.

Buscamos que la serie sea estacionaria, es decir, que no tenga tendencia ni estacionalidad y que la varianza sea homogénea.

La clase prep, creada por la función prep(), contiene toda la información sobre el pre-procesamiento realizado a la serie.

Vamos a describir las funciones:

| **Función** | **prep()** |
|---|---|
| **Descripción** | Lleva a cabo una serie de operaciones que se pueden indicar en los parámetros para transformar una serie temporal. |
| **Parámetros** | 1. tserie. Objeto ts. Serie temporal.<br>2. homogenize.method. Carácter. Método para homogeneizar la varianza. |

| | |
|---|---|
| | Las opciones son “log”, “boxcox” y “none”. Por defecto: “log”.<br>3. detrend.method. Carácter. Método para eliminar la tendencia. Las opciones son “differencing”, “sfsm” y “none”. Por defecto: “differencing”.<br>4. nd. Número. Diferencias regulares a aplicar. Por defecto: NULL.<br>5. deseason.method. Carácter. Método para eliminar la estacionalidad. Las opciones son “differencing” y “none”.<br>6. nsd. Número. Diferencias estacionales a aplicar. Por defecto: NULL.<br>7. detrend.first. Variable lógica. Por defecto: TRUE. |
| **Resultado** | Objeto de la clase prep |
| **Funciones genéricas** | plot.pred()<br>summary.pred()<br>print.pred() |

**Tabla 5.1 Función prep()**

| **Función** | **prep.homogenize.log()** |
|---|---|
| **Descripción** | Transformación logarítmica sobre una serie temporal |
| **Parámetros** | 1. tserie. Objeto ts. Serie temporal |
| **Resultado** | Objeto ts. |

**Tabla 5.2 Función prep.homogenize.log()**

| **Función** | **prep.homogenize.boxcox()** |
|---|---|
| **Descripción** | Transformación Box-Cox sobre una serie temporal |
| **Parámetros** | 1. tserie. Objeto ts. Serie temporal |
| **Resultado** | Lista:<br>1. Objeto ts.<br>2. Número. Valor de lambda. |

**Tabla 5.3 Función prep.homogenize.boxcox()**

| **Función** | **prep.detrend.differencing()** |
|---|---|
| **Descripción** | Diferencias regulares para eliminar la tendencia de una serie temporal. Si no se especifica número de diferencias se estiman con tests de raíz unitaria. |
| **Parámetros** | 1. tserie. Objeto ts. Serie temporal |

| | |
|---|---|
| | 2. nd. Número. Diferencias regulares a aplicar. Por defecto: NULL. |
| **Resultado** | Lista:<br>1. tserie. Objeto ts.<br>2. nd. Número.<br>3. firstvalues. Número. Valores perdidos en la diferenciación. |

**Tabla 5.4 Función prep.detrend.differencing()**

| **Función** | **prep.detrend.sfsm()** |
|---|---|
| **Descripción** | Eliminación de la tendencia por medio del método de "Extracción de medias estacionales". |
| **Parámetros** | 1. tserie. Objeto ts. Serie temporal |
| **Resultado** | Lista:<br>1. tserie. Objeto ts.<br>2. means. Número. Vector de medias utilizado. |

**Tabla 5.5 Función prep.detrend.sfsm()**

| **Función** | **prep.deseason.differencing()** |
|---|---|
| **Descripción** | Eliminación de la estacionalidad por medio de la aplicación de diferencias estacionales. |
| **Parámetros** | 1. tserie. Objeto ts. Serie temporal<br>2. nsd. Número. Diferencias estacionales. |
| **Resultado** | Lista:<br>1. tserie. Objeto ts.<br>2. nsd. Número.<br>3. firstseasons. Número. Estaciones perdidas en la diferenciación. |

**Tabla 5.6 Función prep.deseason.differencing()**

| **Función** | **prep.check.acf()** |
|---|---|
| **Descripción** | Dibuja el gráfico de autocorrelación parcial. |
| **Parámetros** | 1. tserie. Objeto ts. Serie temporal |

**Tabla 5.7 Función prep.check.acf()**

| **Función** | **prep.check.adf()** |
|---|---|
| **Descripción** | Ejecuta el test Augmented Dickey-Fuller y muestra los resultados. |

| **Parámetros** | 1. tserie. Objeto ts. Serie temporal |
|---|---|

**Tabla 5.8 Función prep.check.adf()**

### 5.2.3.Módulo modl

El objetivo es construir un modelo predictivo. Actualmente están implementados los métodos ARIMA y de regresión de Minería de Datos.

En este módulo se crea el objeto modl en la función modl(), que almacena la información del modelo.

Entre los algoritmos de regresión de Minería de Datos tenemos un amplio abanico de posibilidades. Una lista completa de todos los algoritmos disponibles se encuentra en el github de caret, la biblioteca de la que se extraen[5]. Hay que tener en cuenta que sólo funcionarán los algoritmos marcados como aptos para problemas de regresión.

Para entrenar modelos con caret necesitamos un dataset de observaciones típico de minería de datos, del tipo {X1,…,Xk}{Y} siendo Xn los predictores e Y la clase. Por tanto debemos transformar la serie en un data frame. El parámetro formula está relacionado con esto. Es un vector de números de tamaño mayor que 6 que indicará los índices en la serie temporal de la primera observación que se tomará. Por ejemplo: un valor de formula {1,2,3,4,5,6,7,8} tendrá como consecuencia que la primera observación esté compuesta por los ocho primeros elementos de la serie. El último valor actuará como la clase. Si el valor de fórmula es {1,3,4,5,6,7,8}, no se incluirá el segundo valor de la serie en la primera observación, ni el tercer valor en la segunda observación, etc.

Por último señalar el significado de los parámetros horizon, initialWindow y fixedWindow en la función modl(). Son los parámetros de una división especial de los datos para la minería de series temporales[6].

Se han añadido unas medidas de errores personalizadas para los modelos de Minería de Datos para que coincidan con los de ARIMA y así poder comparar ambos tipos de modelos. Estas medidas de error son:

[5] http://topepo.github.io/caret/train-models-by-tag.html
[6] http://topepo.github.io/caret/data-splitting.html#data-splitting-for-time-series

- RMSE. Error cuadrático medio. Promedio de los errores al cuadrado.
- MAE. Error medio absoluto.
- MAPE. Error porcentual absoluto. Media del error porcentual.

Si el usuario quiere obtener las medidas de error del modelo de Minería de Datos predeterminadas de caret, deberá dar al parámetro givenSummary el valor FALSE en la función trControl().

Funciones:

| **Función** | **modl()** |
|---|---|
| **Descripción** | Construye un modelo predictivo para una serie temporal. |
| **Parámetros** | 1. tserie. Objeto ts/Objeto pred. Serie temporal.<br>2. method. Carácter. Los métodos disponibles son "arima" y "dataMining". Por defecto: "arima".<br>3. algorithm. Carácter. Algoritmo de regresión.<br>4. formula. Número. Por defecto: NULL.<br>5. initialWindow. Número. Por defecto: NULL.<br>6. horizon. Número. Por defecto: NULL.<br>7. fixedWindow. Variable lógica. Por defecto: NULL. |
| **Resultado** | Objeto de la clase modl |
| **Funciones genéricas** | summary.pred()<br>print.pred() |

**Tabla 5.9 Función modl()**

| **Función** | **modl.arima()** |
|---|---|
| **Descripción** | Construye un modelo ARIMA. |
| **Parámetros** | 1. tserie. Objeto ts/Objeto pred. Serie temporal. |
| **Resultado** | Objeto de clase forecast::Arima |

**Tabla 5.10 Función modl.arima()**

| **Función** | **modl.tsToDataFrame()** |
|---|---|
| **Descripción** | Transforma una serie temporal en un data frame adaptado para la Minería de Datos |
| **Parámetros** | 1. tserie. Objeto ts/Objeto pred. Serie temporal. |

| | |
|---|---|
| | 2. formula. Número. |
| **Resultado** | data.frame |

**Tabla 5.11 Función modl.tsToDataFrame()**

| **Función** | **modl.trControl()** |
|---|---|
| **Descripción** | Crea el objeto necesario para el control de la división de los datos. |
| **Parámetros** | 1. initialWindow. Número.<br>2. horizon. Número.<br>3. fixedWindow. Variable lógica.<br>4. givenSummary. Variable lógica. |
| **Resultado** | Objeto caret::trainControl |

**Tabla 5.12 Función modl.trControl()**

| **Función** | **modl.dataMining()** |
|---|---|
| **Descripción** | Construye el modelo predictivo con métodos de regresión de Minería de Datos. |
| **Parámetros** | 1. form. Objeto formula. Y ~ X1 + X2 + …<br>2. tserieDF. data.frame.<br>3. algorithm. Carácter.<br>4. timeControl. Objeto trainControl().<br>5. metric. Carácter. Indica la métrica utilizada para optimizar el modelo. Por defecto: “RMSE”.<br>6. maximize. Variable lógica. Indica si el objetivo es maximizar el error. Por defecto: FALSE. |
| **Resultado** | Objeto caret::train |

**Tabla 5.13 Función modl.dataMining()**

### 5.2.4.Módulo pred

Se calcularán las predicciones en base a un modelo predictivo.

En este módulo se crea un objeto pred a partir de la función(), que contiene la serie temporal y las predicciones que se obtienen de ella con el modelo.

Funciones:

| Función | pred() |
|---|---|
| **Descripción** | Predice valores futuros en base a un modelo predictivo. |
| **Parámetros** | 1. model. Objeto de tipo modl.<br>2. n.ahead. Número. Predicciones que se harán sobre el futuro. Máximo 100. Por defecto: 20.<br>3. tserie. Objeto ts. En caso de asignar directamente una serie temporal.<br>4. predictions. Objeto ts. En caso de asignar directamente las predicciones. |
| **Resultado** | Objeto pred |
| **Funciones genéricas** | plot.pred()<br>print.pred()<br>summary.pred() |

**Tabla 5.14 Función pred()**

| Función | pred.arima() |
|---|---|
| **Descripción** | Calcula predicciones para un modelo ARIMA. |
| **Parámetros** | 1. model. Objeto Arima.<br>2. n.ahead. Número. |
| **Resultado** | Objeto ts con las predicciones. |

**Tabla 5.15 Función pred.arima()**

| Función | pred.dataMining() |
|---|---|
| **Descripción** | Calcula predicciones para un modelo de Minería de Datos. |
| **Parámetros** | 1. model. Objeto modl.<br>2. n.ahead. Número. |
| **Resultado** | Objeto ts con las predicciones. |

**Tabla 5.16 Función pred.dataMining()**

| Función | pred.compareModels() |
|---|---|
| **Descripción** | Dibuja un gráfico comparativo entre una serie temporal y las distintas predicciones obtenidas a partir de distintos modelos (entre 2 y 5 distintos). |
| **Parámetros** | 1. originalTS. Objeto ts.<br>2. p_1. Objeto ts.<br>3. p_2. Objeto ts. |

4. p_3. Objeto ts. Por defecto:NULL.
5. p_4. Objeto ts. Por defecto:NULL.
6. p_5. Objeto ts. Por defecto:NULL.
7. legendNames. Carácter. Vector de nombres para la leyenda. Por defecto:NULL.
8. colors. Carácter. Vector de colores para las líneas. Por defecto:NULL.
9. legend. Variable lógica. ¿Queremos una leyenda? Por defecto:TRUE.
10. legendPosition. Carácter. Posición de la leyenda ("bottomright", "topleft", etc). Por defecto:NULL.
11. yAxis. Carácter. Nombre del eje y. Por defecto: "Values".
12. title. Carácter. Título de la gráfica. Por defecto: "Predictions".

**Tabla 5.17 Función pred.compareModels()**

### 5.2.5.Módulo postp

Este módulo tiene como único objetivo deshacer el pre-procesamiento de una serie tras aplicarle un modelo predictivo, de forma que tanto la serie como las predicciones tomen valores realistas.

Funciones:

| Función | postp() |
|---|---|
| **Descripción** | Deshace el pre-procesamiento a una serie y a sus predicciones. |
| **Parámetros** | 1. prd. Objeto pred.<br>2. pre. Objeto prep. |
| **Resultado** | Objeto pred con los valores actualizados. |

**Tabla 5.18 Función postp()**

| Función | postp.homogenize.log() |
|---|---|
| **Descripción** | Deshace la transformación logarítmica. |
| **Parámetros** | 1. tserie. Objeto ts. |
| **Resultado** | Objeto ts. |

**Tabla 5.19 Función postp.homogenize.log()**

| Función | postp.homogenize.boxcox() |
|---|---|
| **Descripción** | Deshace la transformación Box-Cox. |
| **Parámetros** | 1. tserie. Objeto ts.<br>2. lambda. Número. |
| **Resultado** | Objeto ts. |

**Tabla 5.20 Función postp.homogenize.boxcox()**

| Función | postp.detrend.differencing() |
|---|---|
| **Descripción** | Deshace la diferenciación regular. |
| **Parámetros** | 1. tserie. Objeto ts.<br>2. nd. Número.<br>3. firstvalues: Número. |
| **Resultado** | Objeto ts. |

**Tabla 5.21 Función postp.detrend.differencing()**

| Función | postp.detrend.sfsm() |
|---|---|
| **Descripción** | Deshace la extracción de medias estacionales. |
| **Parámetros** | 1. tserie. Objeto ts.<br>2. means. Número.<br>3. start. Fecha. Comienzo de la serie original.<br>4. frequency. Número. Frecuencia de la serie original |
| **Resultado** | Objeto ts. |

**Tabla 5.22 Función postp.detrend.sfsm()**

| Función | postp.deseason.differencing() |
|---|---|
| **Descripción** | Deshace la diferenciación estacional. |
| **Parámetros** | 1. tserie. Objeto ts.<br>2. nsd. Número.<br>3. firstseasons: Número. |
| **Resultado** | Objeto ts. |

**Tabla 5.23 Función postp.deseason.differencing()**

### 5.2.6.Flujo de trabajo

En este apartado se explica con un ejemplo el flujo de trabajo normal con la biblioteca predtoolsTS para un usuario medio.

Se utilizará la serie temporal co2, que es uno de los datasets de los que R dispone por defecto. Representa la concentración atmosférica de CO2 expresada en partes por millón (ppm).

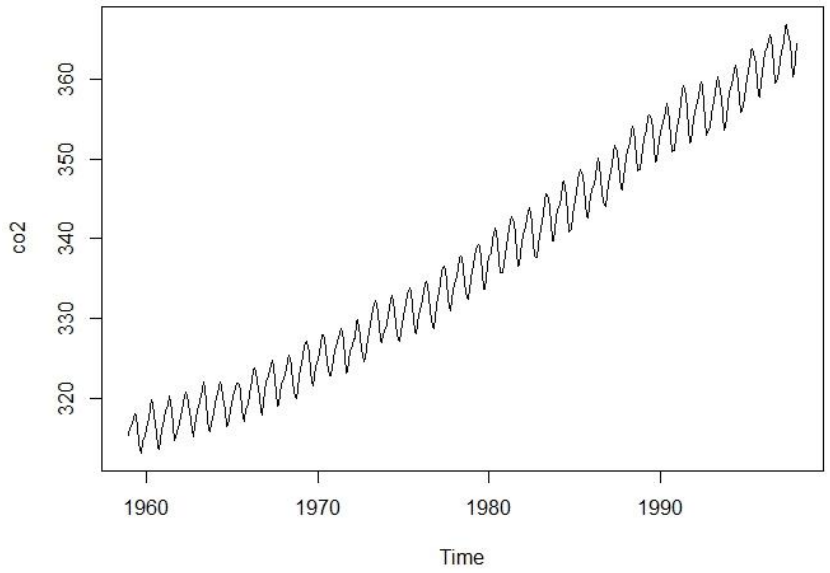



**Ilustración 5.2 Serie temporal: Concentración de CO2 en la atmósfera. 1959-1997**

A primera vista resulta obvio que la serie tiene una marcada tendencia al alza y una componente estacional. La varianza parece constante.

Vamos a convertir esta serie en estacionaria.

Primero lo intentaremos con el enfoque automático. Pasaremos a la función prep() la serie temporal y no indicaremos el resto de parámetros.

```
co2.prep = prep(co2)

plot(co2.prep)
```

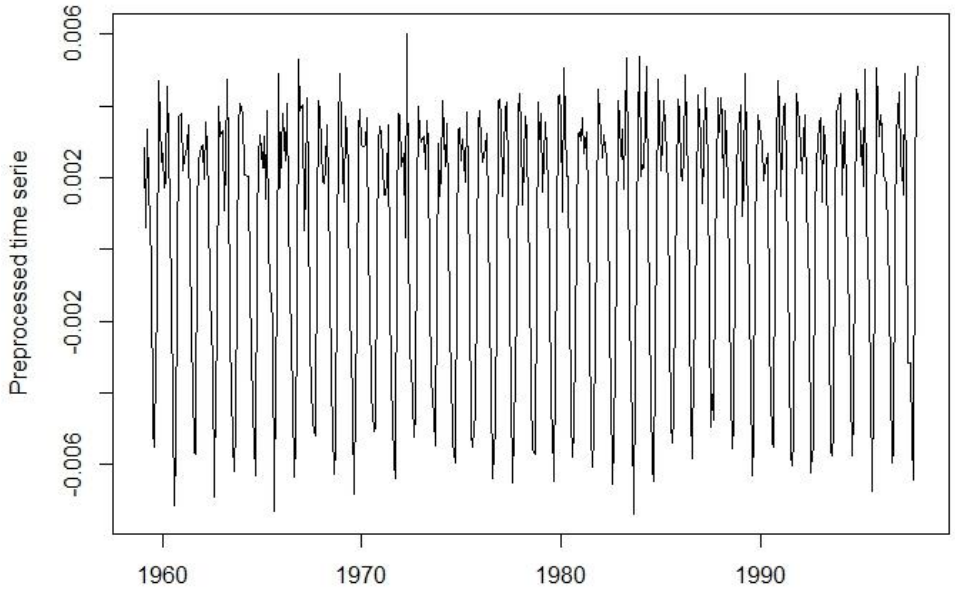



**Ilustración 5.3 Serie temporal co2 pre-procesada de forma automática**

Observando la serie parece que persiste la componente estacional. Vamos a comprobar si la serie es estacionaria apoyándonos en otras herramientas de las que disponemos.

```
prep.check.adf(co2.prep)
```

```
        Augmented Dickey-Fuller Test

data:  tserie$tserie
Dickey-Fuller = -30.215, Lag order = 7, p-value = 0.01
alternative hypothesis: stationary
```

**Ilustración 5.4 Test Augmented Dickey-Fuller**

El test ADF nos indica que la serie es estacionaria. Sin embargo ya sabemos que este tipo de pruebas no son infalibles, y siempre es mejor fijarnos en el gráfico ACF.

```
prep.check.acf(co2.prep)
```

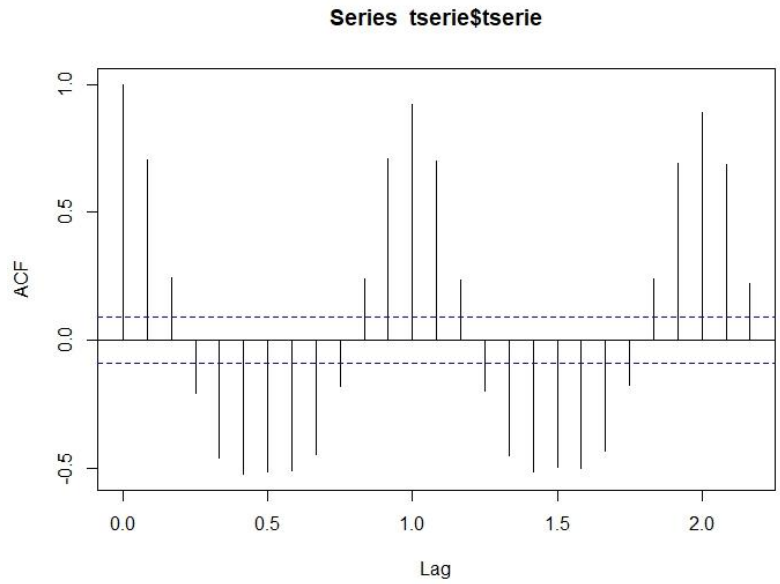



**Ilustración 5.5 ACF sobre la serie co2 pre-procesada de forma automática**

Ahora sí podemos concluir con seguridad que no es estacionaria.

Recordemos que el enfoque automático se basa en tests que con frecuencia pueden fallar. Por eso siempre debemos estar pendientes de los gráficos y asegurarnos con el ACF.

Vamos a intentarlo ahora modificando los parámetros de prep().

La varianza parece constante, pero hacer una transformación logarítmica nunca es malo para el sistema, por lo que dejaremos el valor por defecto por si acaso.

Tomaremos una diferencia regular para la tendencia y una diferencia estacional para la estacionalidad.

```
co2.prep=prep(co2,detrend.method='differencing',nd=1,deseason.method='differencing',nsd=1)

plot(co2.prep)
```

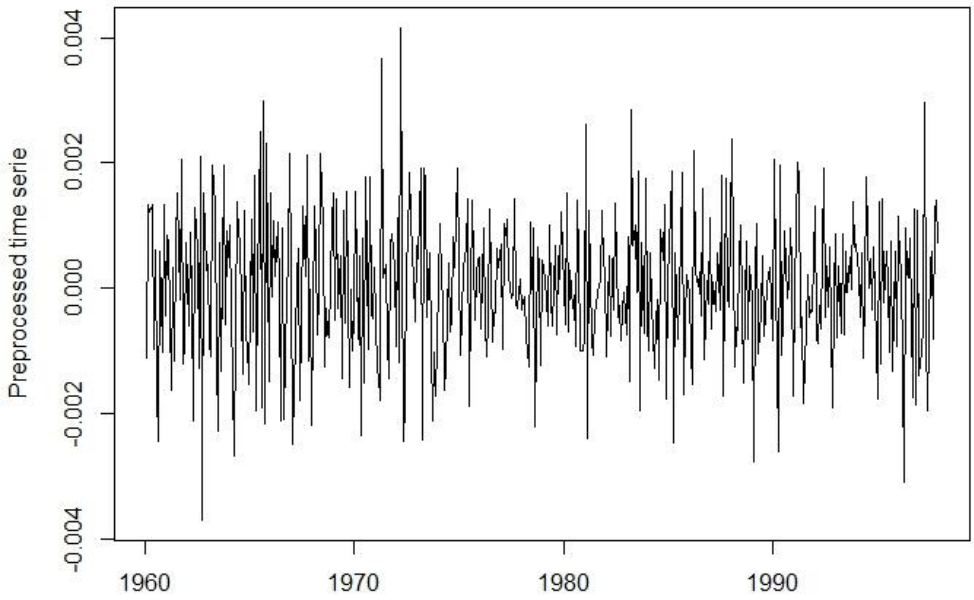



**Ilustración 5.6 Serie temporal co2 pre-procesada con especificación de parámetros**

Esta serie ya sí es estacionaria. Puede comprobarse con el ACF.

```
prep.check.acf(co2.prep)
```

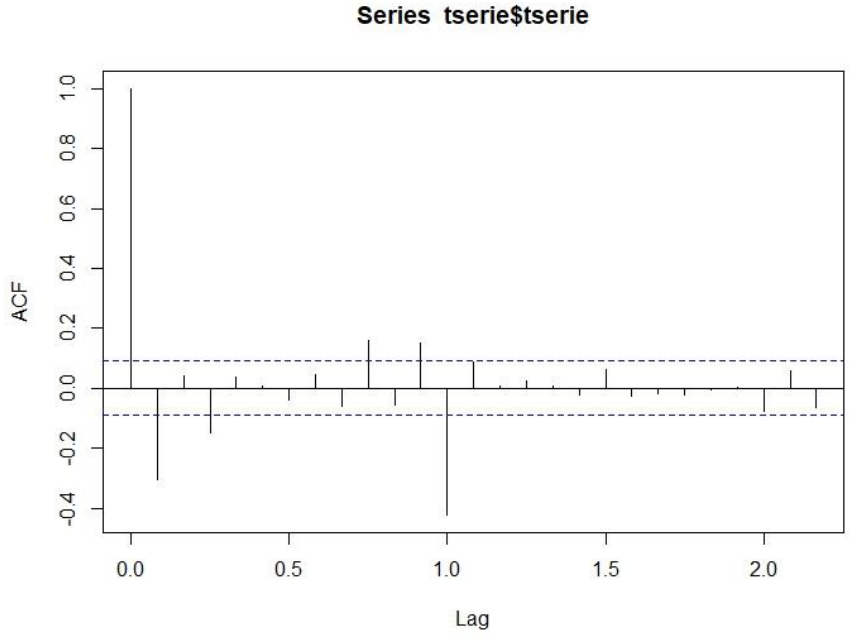



**Ilustración 5.7 ACF sobre co2 pre-procesada con especificación de parámetros**

Ahora que tenemos el objeto prep, que contiene la serie estacionaria, podemos pasar a la siguiente fase: el modelado.

Antes de ejecutar un algoritmo de minería deberíamos conocer el funcionamiento del algoritmo y si nuestra serie temporal es demasiado grande, pues podríamos concurrir en tiempos de ejecución muy altos. Podemos jugar con el tamaño del dataset generado a partir de la serie de la serie con el parámetro formula de modl(). Cuanto más largas sean las observaciones, más corta será la serie, y viceversa.

Otra forma de jugar con el volumen de operaciones es variando los parámetros para la división de datos "rolling origin forecast". Un fixedWindow=TRUE supondrá un volumen de datos menor que en su valor falso. Un initialWindow y un horizon altos también suponen menos iteraciones durante la ejecución.

Vamos a ejecutar el algoritmo Boosted Generalized Linear Model, que tiene la etiqueta "glmboost". Además vamos a hacer más grande el tamaño de las observaciones (el valor por defecto es 16).

```
co2.modl=modl(co2.prep,method='dataMining',algorithm='glmboost', formula=c(1:20))

summary(co2.modl)
```

```
Fitted model object
~Modelling method: Data mining
~Algorithm:  glmboost
~Original time serie:
 Time-Series [1:455] from 1960 to 1998: -0.001112 0.001323 0.001214 0.001338 -0.000968 ...
~Original time serie as data frame'data.frame': 436 obs. of  20 variables:
 $ c_1  : num  -0.001112 0.001323 0.001214 0.001338 -0.000968 ...
 $ c_2  : num  0.001323 0.001214 0.001338 -0.000968 0.00062 ...
 $ c_3  : num  0.001214 0.001338 -0.000968 0.00062 -0.001649 ...
 $ c_4  : num  0.001338 -0.000968 0.00062 -0.001649 -0.002439 ...
 $ c_5  : num  -0.000968 0.00062 -0.001649 -0.002439 0.000576 ...
 $ c_6  : num  0.00062 -0.001649 -0.002439 0.000576 -0.001023 ...
 $ c_7  : num  -0.001649 -0.002439 0.000576 -0.001023 0.001328 ...
 $ c_8  : num  -0.002439 0.000576 -0.001023 0.001328 -0.000447 ...
 $ c_9  : num  0.000576 -0.001023 0.001328 -0.000447 0.000848 ...
 $ c_10 : num  -0.001023 0.001328 -0.000447 0.000848 0.000718 ...
 $ c_11 : num  0.001328 -0.000447 0.000848 0.000718 -0.001641 ...
 $ c_12 : num  -0.000447 0.000848 0.000718 -0.001641 0.000339 ...
 $ c_13 : num  0.000848 0.000718 -0.001641 0.000339 -0.001155 ...
 $ c_14 : num  0.000718 -0.001641 0.000339 -0.001155 0.000725 ...
 $ c_15 : num  -0.001641 0.000339 -0.001155 0.000725 0.001526 ...
 $ c_16 : num  0.000339 -0.001155 0.000725 0.001526 -0.000175 ...
 $ c_17 : num  -0.001155 0.000725 0.001526 -0.000175 0.002067 ...
 $ c_18 : num  0.000725 0.001526 -0.000175 0.002067 -0.001219 ...
 $ c_19 : num  0.001526 -0.000175 0.002067 -0.001219 -0.000896 ...
 $ Class: num  -0.000175 0.002067 -0.001219 -0.000896 0.000718 ...
~Model (best tuning parameters):
  mstop prune
1    50    no
~Error measures:
         RMSE          MAE         MAPE
1.095684e-03 8.904621e-04 1.912201e+02
```

**Ilustración 5.8 Resultados del entrenamiento del modelo de Minería de Datos con el algoritmo "glmboost"**

Ya tenemos nuestro modelo predictivo. A continuación vamos a hacer predicciones sobre el modelo para calcular 40 observaciones en el futuro.

```
co2.pred=pred(co2.modl,n.ahead=40)

plot(co2.pred)
```

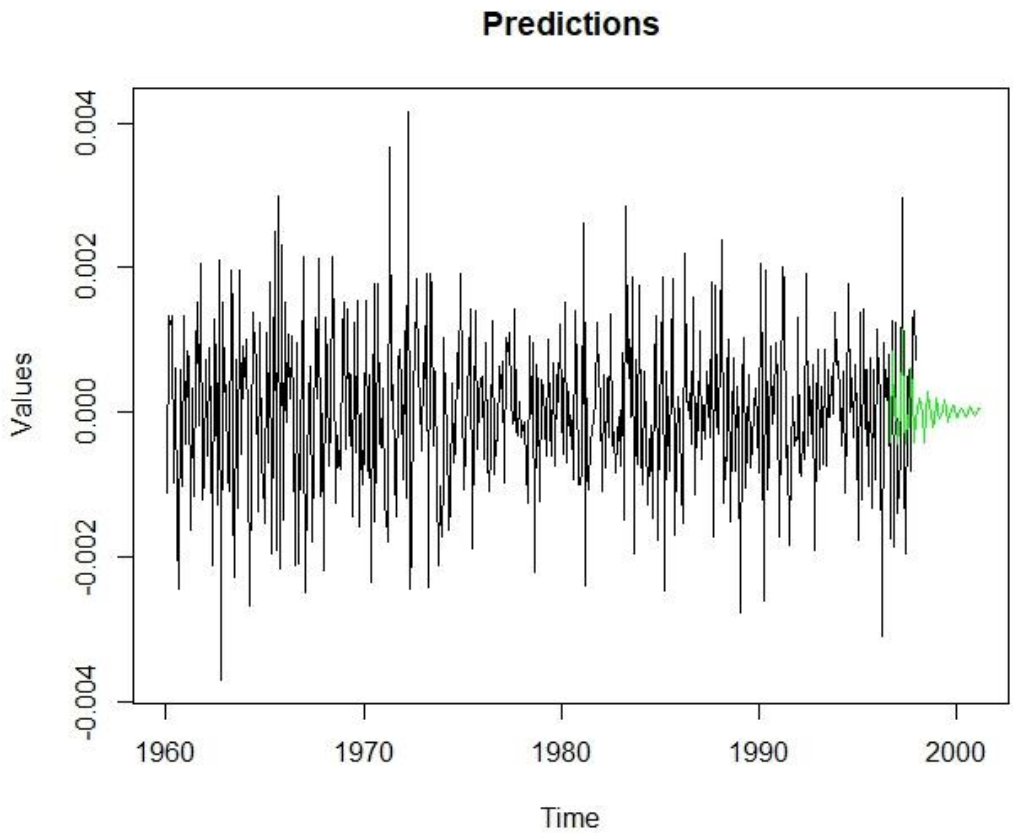



**Ilustración 5.9 Predicciones del modelo de Minería de Datos con el algoritmo "glmboost" sobre la serie co2 pre-procesada**

Podemos ver que la predicción se superpone con la serie original. Esto es debido a que durante la fase de modelado la última parte de la serie nunca es accedida por el conjunto de entrenamiento, dado el paradigma de "rolling origin forecast".

La predicción no parece muy prometedora. Vamos a post-procesar la serie para verla sobre los datos reales.

```
co2.pred2=postp(co2.pred,co2.prep)

plot(co2.pred2)
```

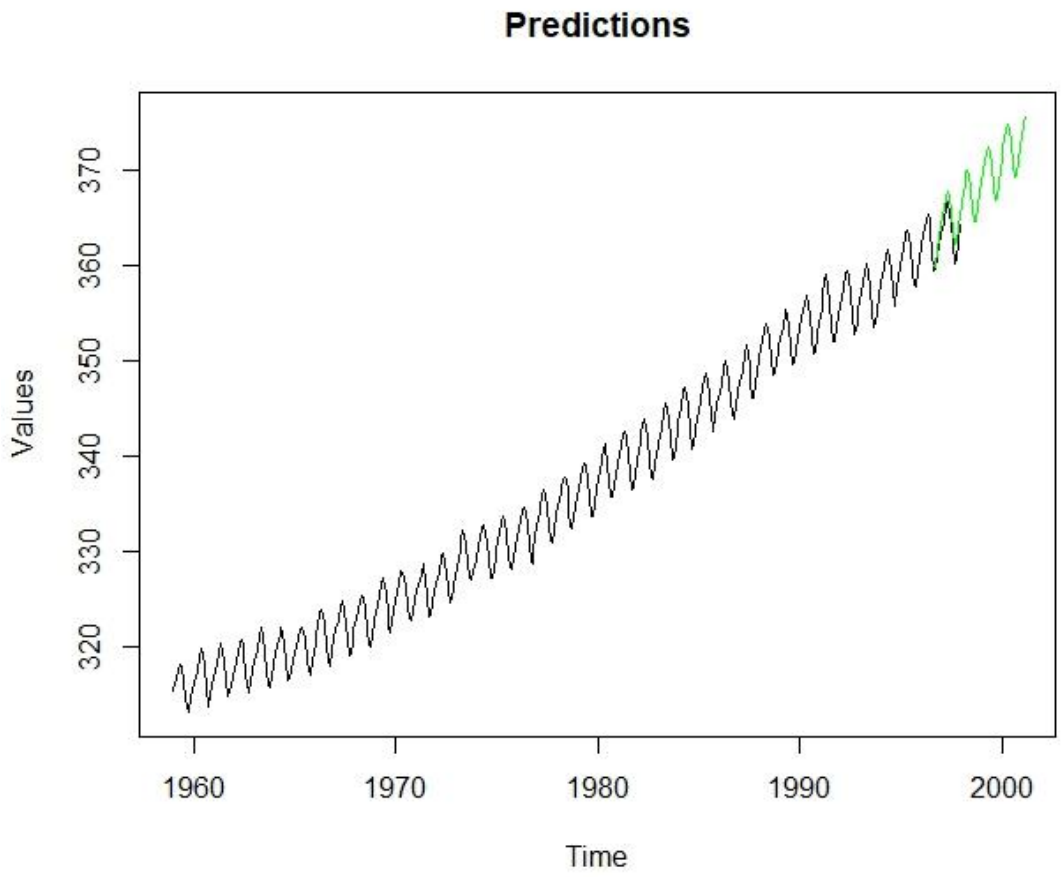



**Ilustración 5.10 Predicciones del modelo de Minería de Datos con el algoritmo "glmboost" sobre la serie original después de post-procesar**

Podemos ver la predicción más de cerca si acortamos la serie. Vamos a visualizar un subconjunto de la serie original que empezará en 1990 junto a las predicciones:

```
co2.pred3=pred(tserie=window(co2.pred2$tserie,start=1990),
predictions=co2.pred2$predictions)

plot(co2.pred3)
```

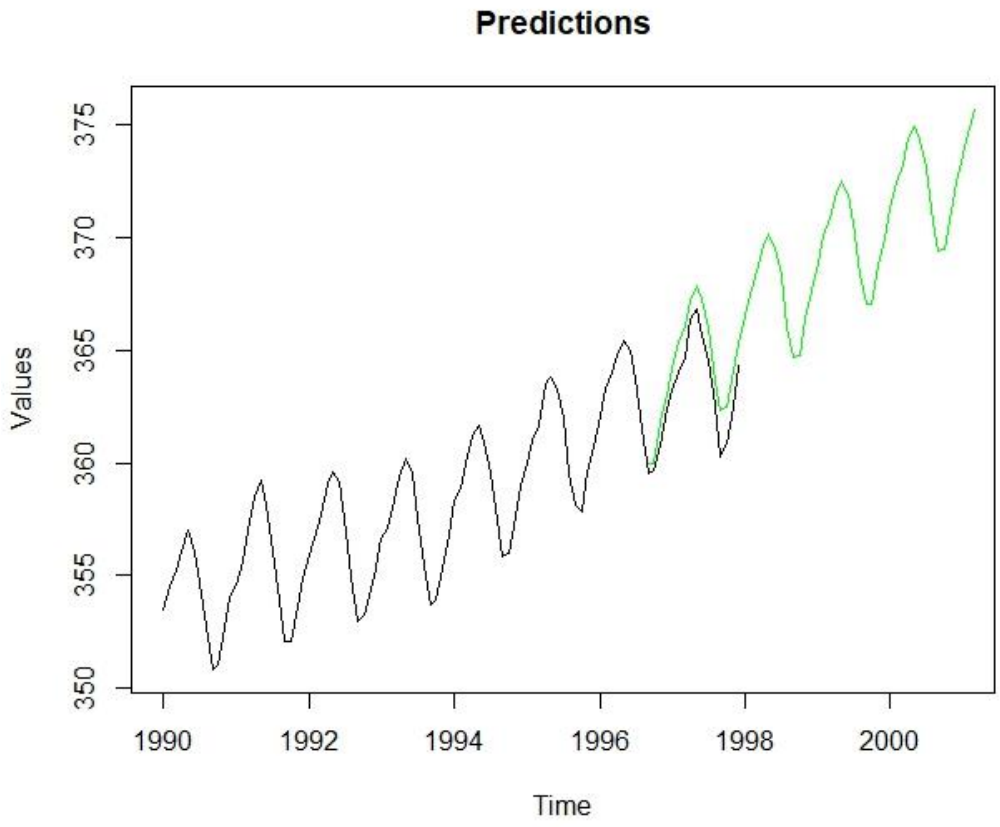



**Ilustración 5.11 Predicciones del modelo de Minería de Datos con el algoritmo "glmboost" sobre la serie acotada**

Como última tarea haremos dos modelos predictivos más sobre la serie temporal para comparar gráficamente las predicciones.

Para el primero utilizaremos la serie no estacionaria que obteníamos al principio con el pre-procesamiento automático con el mismo algoritmo para comprobar hasta qué punto influye la estacionariedad en el rendimiento.

Para el segundo utilizaremos un modelo ARIMA.

Vamos a comparar los modelos:

```
pred.compareModels(co2.pred3$tserie, co2.pred3$predictions, nonStationaryPred$predictions,
co2.arimaPred$predictions, colors=c("blue","red","green","gold"),
legendNames=c("CO2","GLMBOOST estacionaria", "GLMBOOST no
estacionaria","ARIMA"))
```

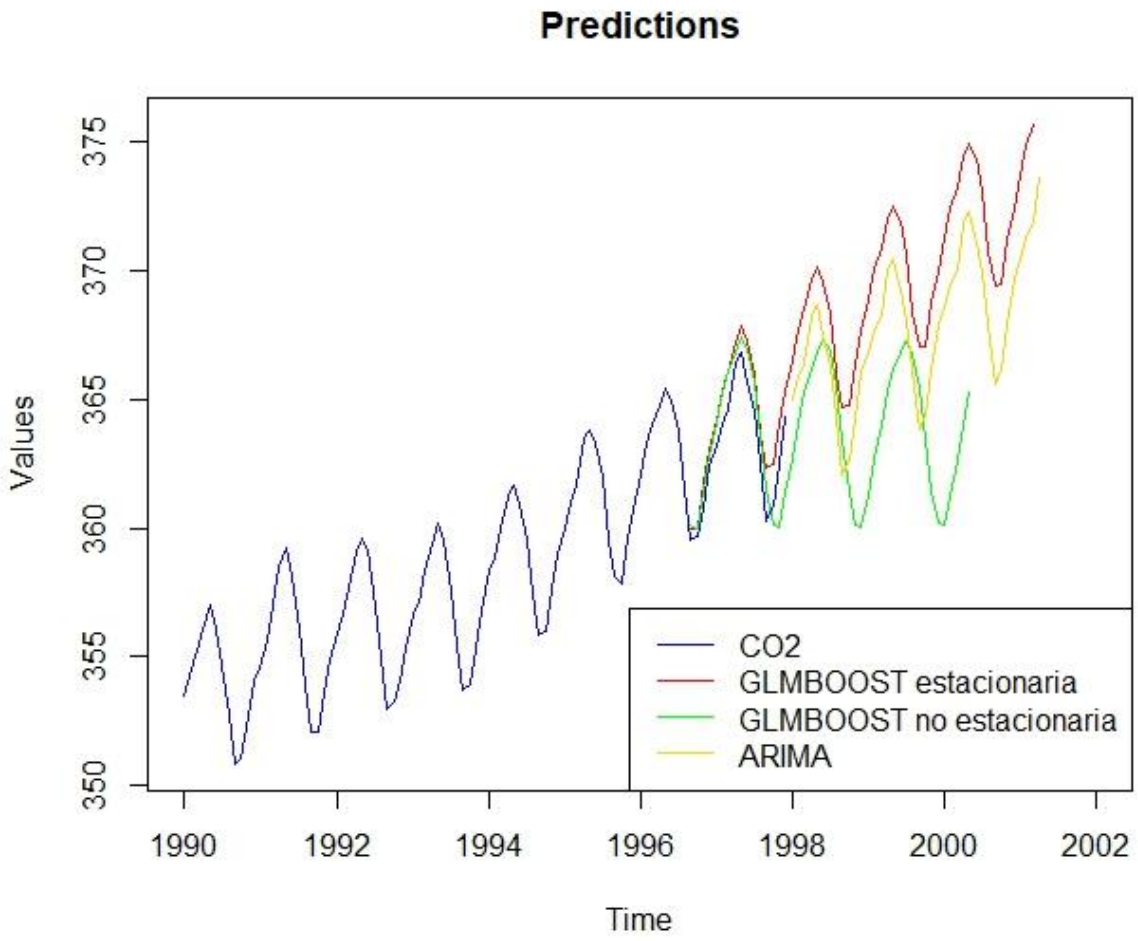



**Ilustración 5.12 Comparativa de modelos predictivos sobre la serie co2 acotada**

Podemos observar como las predicciones sobre el modelo no estacionario se desvían completamente del rumbo lógico de la gráfica. Además parece que las predicciones del primer modelo son bastante más optimistas que las obtenidas por el modelo ARIMA. En este caso, podríamos decir que el modelo ARIMA nos da las predicciones más realistas.

Si queremos más información también podemos extraer los errores de los tres modelos (recordemos que se almacenan en el objeto modl).

```
> co2.modl$errors
         RMSE          MAE          MAPE
1.095684e-03 8.904621e-04 1.912201e+02
> arima.modl$errors
         RMSE          MAE          MAPE
1.108748e-03 8.898244e-04 1.426017e+02
> nonStationary.modl$errors
         RMSE          MAE          MAPE
1.072378e-03 8.762045e-04 5.435937e+01
```

**Ilustración 5.13 Medidas de error de los modelos predictivos**

Curiosamente vemos que las mejores medidas de error las tiene el modelo no estacionario. Probablemente esto es debido a un sobreajuste del modelo, ya que recordemos que la Minería de Datos busca patrones, y la serie no estacionaria tenía muchos (la componente estacional, como vimos en la ilustración 5.2). Por eso, da resultados muy buenos cuando las predicciones se alejan poco de la serie original. Recordemos que estas medidas de error se toman sobre el conjunto de test, que es consecutivo al conjunto de entrenamiento. Sin embargo, cuanto más se aleja más se desvía del camino correcto.

# 6. Pruebas y experimentación

En este apartado llevaremos a la práctica la utilidad del paquete predtoolsTS: evaluaremos modelos predictivos, los compararemos entre sí, y sacaremos conclusiones sobre los resultados.

## 6.1. Evaluación del método de extracción de medias estacionales.

En este primer experimentos comprobaremos el rendimiento de la extracción de medias estacionales para eliminar la tendencia de una serie comparándolo con un modelo de diferenciación regular.

Para ello utilizaremos una serie con tendencia, como “AirPassengers”. Esta serie recoge el total de pasajeros al mes de una aerolínea. De 1949 a 1960.

Aplicaremos los dos métodos paralelamente para eliminar la tendencia, y con cada uno construiremos un modelo con los mismos parámetros. Se aplicará un algoritmo basado en árboles llamado M5.

Después de hacer las predicciones y efectuar el post-procesamiento obtenemos la siguiente gráfica comparativa:

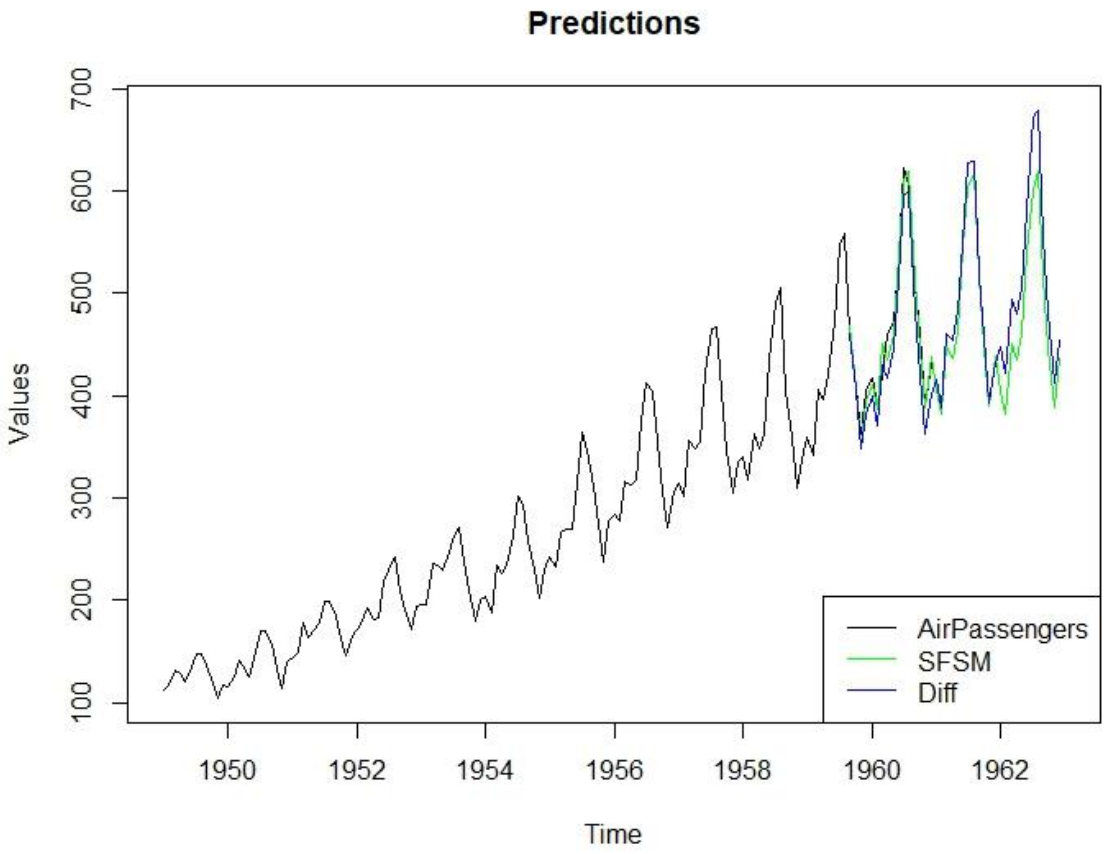



**Ilustración 6.1 Comparativa de modelo SFSM y Diff sobre la serie AirPassengers**

Es difícil de dilucidar, no hay mucha diferencia entre las predicciones, pero da la sensación de que el método de la extracción de medias estacionales hace que las predicciones dejen de crecer con el tiempo; parece que su gráfica se tuerce hacia abajo.

Veamos la tabla de errores de los modelos predictivos:

| Modelo | RMSE | MAE | MAPE |
|---|---|---|---|
| **SFSM** | 0.03344244 | 0.02678642 | 200.40479984 |
| **Diff** | 0.03609261 | 0.02996697 | 163.90524725 |

**Tabla 6.1 Medidas de error de modelo SFSM y Diff sobre la serie AirPassengers**

## 6.2. Comparativa de modelos predictivos

Compararemos la eficacia de 5 modelos distintos sobre la serie temporal "café", que recoge los gastos en cafes y restaurantes en Australia entre 1982 y 2010. Los datos se han tomado con frecuencia cuatrimestral.

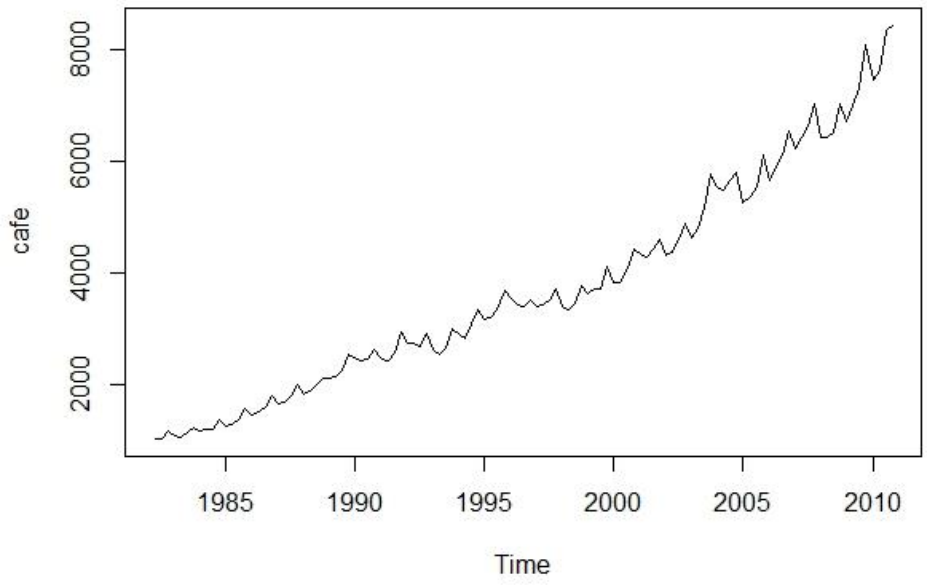



**Ilustración 6.2 Serie temporal: gastos en cafes y restaurantes en Australia. 1982-2010**

Deteniéndonos un momento en la gráfica, podemos observar un repunte estacional que coincide con el último cuatrimestre del año. Además es obvia la tendencia al alza. La varianza también parece ser demasiado heterogénea.

Convertimos la serie en estacionaria aplicando transformación logarítmica, una diferencia regular y una diferencia estacional.

A continuación crearemos todos los modelos. Para este experimento no variaremos ningún parámetro, sólo compararemos el rendimiento de los algoritmos. Utilizaremos los siguientes:

- Modelo ARIMA.
- Algoritmo CART.

- Algoritmo Bagged MARS.
- Algoritmo Red Neuronal.
- Algoritmo Modelo Lineal Generalizado de Bayes.

La comparativa gráfica de los modelos es la siguiente:

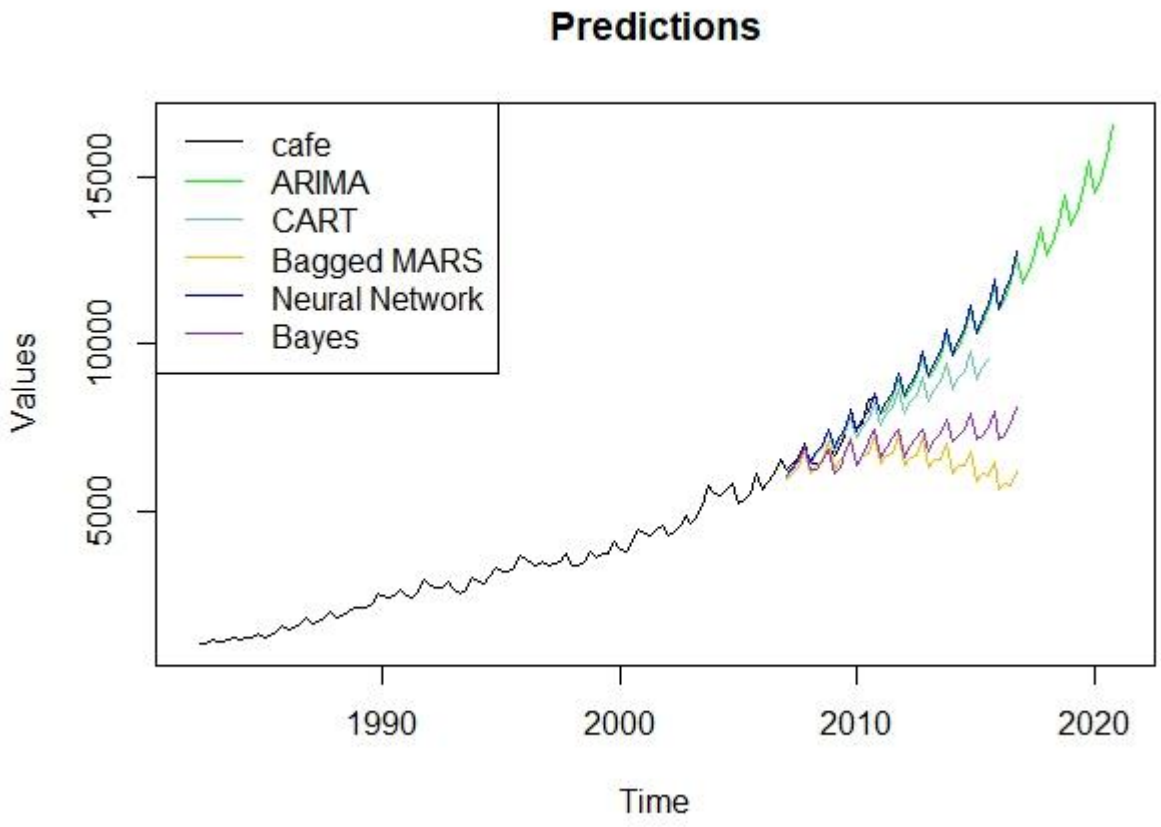



**Ilustración 6.3 Predicciones de cinco modelos distintos sobre la serie cafe**

De la tabla podemos suponer que ARIMA y el algoritmo de redes neuronales son los que mejor se comportan, o los que se ajustan mejor a los valores previstos. El Bagged MARS y el de Bayes son muy pesimistas. Bagged MARS parece que se comporta especialmente mal, pues se aleja de manera exponencial de la curva esperada.

Para ayudarnos a sacar conclusiones podemos comparar las medidas de error en el modelado.

Tabla de medidas de error:

| Modelo | RMSE | MAE | MAPE |
|---|---|---|---|
| **ARIMA** | 0.02813997 | 0.02318880 | -66.61981053 |
| **CART** | 0.03504137 | 0.02775857 | 361.91141520 |
| **Bagged MARS** | 0.03371144 | 0.02799415 | 207.88690603 |
| **Neural Network** | 0.03648905 | 0.03148952 | 100.60229670 |
| **Bayes** | 0.03400491 | 0.02791308 | 269.73615190 |

**Tabla 6.2 Medidas de error de cinco modelos distintos sobre la serie AirPassengers**

El mejor modelo es sin duda ARIMA.

Resulta curioso cómo Bagged MARS se comporta, como si trazara un arco; cuando está cerca de los datos de entrenamiento los resultados son buenos, pero decrecen rápidamente.

El de redes neuronales se comporta mal al principio pero se ajusta rápidamente y saca prácticamente las mismas conclusiones que ARIMA.

# 7. Conclusiones

Se han podido cumplir todos los objetivos que nos pusimos en las etapas tempranas del T.F.G. Tenemos una biblioteca con el potencial de integrar un número indeterminado de métodos de procesamiento y predicción sobre series temporales.

El siguiente paso es trabajar para que el paquete acabe en CRAN, corrigiendo los errores y añadiendo las funcionalidades que nos sugiera la comunidad de usuarios de R. Una vez en CRAN, lo ideal sería promocionar el paquete entre los foros y blogs relacionados con la materia, de forma que se vaya expandiendo su uso y los usuarios colaboren completándolo.

En cuanto a la investigación sobre la eficacia de los métodos de Minería de Datos en predicción de series temporales, y concordando con lo que había leído algunas veces mientras me documentaba, podemos decir que estos algoritmos son inferiores en precisión y eficiencia al modelo ARIMA; una de las pocas excepciones es el de las Redes Neuronales, que rinde casi al nivel de ARIMA, aunque según el tamaño del dataset puede resultar mucho más costoso computacionalmente. De todas formas, es interesante disponer de una herramienta como esta biblioteca para comparar y experimentar con los modelos.

A nivel personal, la elaboración de este T.F.G. me ha aportado el aprendizaje en profundidad de un lenguaje de programación como R, el conocimiento de la teoría, el potencial y la importancia de las series temporales a día de hoy y me ha hecho adentrarme un poco más en el mundo de la Minería de Datos y la extracción de conocimiento, campos del Tratamiento Inteligente de la Información que ya me interesaban y hacia donde me planteo orientar mi futuro laboral.

# Bibliografía

1. **Rouse, Margaret.** Business Intelligence (BI). *searchdatamanagement.techtarget.com.* [Online] http://searchdatamanagement.techtarget.com/definition/business-intelligence.

2. **Konar, Amit and Bhattacharya, Diptendu.** *Time Series Prediction and Applications: a Machine Learning Approach.* 2017. 1.

3. **Esling, Philippe and Agon, Carlos.** *Time Series Data Mining.* 2012.

4. **Pérez Recuerda, Pedro.** *Métodos para el Análisis y la Predicción de Series Temporales del Precio del Aceite de Oliva en España.* 2008.

5. **Hyndman, Rob and Athanasopoulos, George.** *Forecasting: Principles and Practice.* 2017.

6. **Srivastava, Tavish.** A Complete Tutorial on Time Series Modeling in R. *analyticsvidhya.com.* [Online] 2015. https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/.

7. **Scott, David.** *Box-Cox Transformations.*

8. **Lundholm, Michael.** *Introduction to R's Time Series Facilities.* 2011.

9. **Yan, Weizhong.** *Towards Automatic Time-Series Forecasting using Neural Networks.* 2012.

10. **Justel, Ana and Cayuela, Luis.** *Series Temporales en R.* 2012.

11. **de la Fuente Fernández, Santiago.** *Series Temporales: Modelo ARIMA.*

12. **Mayo, Matthew.** Data Science Basics: Data Mining vs. Statistics. *kdnuggets.com.* [Online] 2016. http://www.kdnuggets.com/2016/09/data-science-basics-data-mining-statistics.html.

13. **Universidad Politécnica de Tlaxcala.** *Modelos Predictivos y Descriptivos en Minería de Datos.* 2015.

14. *From Data Mining to Knowledge Discovery in Data Bases.* **Usama Fayyad, Gregory Piatetsky-Shapiro and Smyth, Padhraic.** 1997.

15. **Hawkins, Douglas.** The Problem of Overfitting. *Journal of chemical information and computer sciences.* 2004.

16. **Kuhn, Max.** *The Caret Package.* 2017.

17. **Ríos, Sánchez and Sergio.** *Análisis de Requerimientos.* 2011.

18. **Junta de Andalucía.** Guía para la redacción de casos de uso. *juntadeandalucia.es.* [Online] http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/416.

19. **Matloff, Norman.** *The Art of R Programming.* 2009.

20. **R Core Team.** What is R? *cran.r-project.org.* [Online] https://www.r-project.org/about.html.

21. **Programiz.** Learn R Programming: The Definitive Guide. *programiz.com.* [Online] https://www.programiz.com/r-programming.

22. **Paulson, Josh.** Developing Packages with RStudio. *support.rstudio.com.* [Online] https://support.rstudio.com/hc/en-us/articles/200486488-Developing-Packages-with-RStudio.

23. **Hyndman, Rob.** Package Forecast. *cran.r-project.org.* [Online] 2017. https://cran.r-project.org/web/packages/forecast/forecast.pdf.

24. **Wickham, Hadley.** *R Packages. Organize, Test, Document and Share your Code.* 2015.

25. —. Introduction to Roxygen2. *cran.r-project.org.* [Online] 2017. https://cran.r-project.org/web/packages/roxygen2/vignettes/roxygen2.html.

26. **Carmona, Francesc.** *Creación de Paquetes de R en Windows.* 2013.

27. **Broman, Karl.** R package primer. A minimal tutorial. *kbroman.org.* [Online] http://kbroman.org/pkg_primer/.

28. **Leisch, Friedrich.** Creating R Packages: a Tutorial. *cran.r-project.org.* [Online] 2009. https://cran.r-project.org/doc/contrib/Leisch-CreatingPackages.pdf.

29. **R Core Team.** Writing R Extensions. *cran.r-project.org.* [Online] 2017. https://cran.r-project.org/doc/manuals/R-exts.html.

30. **Coghlan, Avril.** A Little Book of R for Time Series. *a-little-book-of-r-for-time-series.readthedocs.io.* [Online] http://a-little-book-of-r-for-time-series.readthedocs.io.