UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES
ACATLÁN

Métodos de pronósticos de series de tiempo con R

TESINA Y EXAMEN PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE

LICENCIADO EN MATEMÁTICAS APLICADAS Y COMPUTACIÓN

PRESENTA

CARLOS ILICH CARRO LOZANO

ASESOR: DRA. MARICARMEN GÓNZALEZ VIDEGARAY

ABRIL 2015

Santa Cruz Acatlán, Estado de México

Todos los hombres, en algún momento de su vida, se sienten solos; y más: todos los hombres están solos. Vivir, es separarnos del que fuimos para internarnos en el que vamos a ser, futuro extraño siempre. La soledad es el fondo último de la condición humana. El hombre es el único ser que se siente solo y el único que es búsqueda de otro. Su naturaleza – si se puede hablar de naturaleza al referirse al hombre, el ser que, precisamente, se ha inventado a sí mismo al decirle "no" a la naturaleza – consiste en un aspirar a realizarse en otro. El hombre es nostalgia y búsqueda de comunión. Por eso cada vez que se siente a sí mismo se siente como carencia de otro, como soledad.

El trabajo sin fin, infinito, corresponde a la vida sin finalidad de la sociedad moderna. Y la soledad que engendra, soledad promiscua de los hoteles, de las oficinas, de los talleres y de los cines, no es una prueba que afine el alma, un necesario purgatorio. Es una condenación total, espejo de un mundo sin salida.

El hombre moderno tiene la pretensión de pensar despierto. Pero este despierto pensamiento nos ha llevado por los corredores de una sinuosa pesadilla, en donde los espejos de la razón multiplican las cámaras de tortura. Al salir, acaso, descubriremos que habíamos soñado con los ojos abiertos y que los sueños de la razón son atroces. Quizá, entonces, empezaremos a soñar otra vez con los ojos cerrados.

La plenitud, la reunión, que es reposo y dicha, concordancia con el mundo, nos esperan al fin del laberinto de la soledad.

Octavio Paz, *El laberinto de la soledad*

Dedicatorias

A mi madre, Silvia Lozano, por todo el amor y apoyo incondicional.

A mi padre, Nicolás Carro, por su apoyo y ejemplo.

A los catedráticos de la FES Acatlán por su dedicación constante y especialmente a la Mtra. Luz María Lavín Alanís por su paciencia y todo el apoyo.

A mi asesora, Dra. Maricarmen González Videgaray, por su valioso tiempo dedicado a revisar y mejorar el presente trabajo.

ÍNDICE GENERAL

INTRODUCCIÓN

En la actualidad la ciencia y la tecnología nos han llevado a un mundo que únicamente conocíamos en las novelas de ciencia ficción, y las telecomunicaciones prometen un mundo completamente conectado.

“Creemos que cualquier cosa que se beneficie al estar conectado, tendrá una conexión, inclusive un árbol” (Ericsson, 2014).

Así, una inquietud permanente en el ser humano es, ¿cómo podemos anticiparnos al futuro, y con qué elementos contamos?

Cuando aparecieron las computadoras de manera comercial, trajeron consigo una nueva subcultura, un nuevo estilo de vida, que no tardaría en trasladar aquellas preocupaciones de la vida social, al campo mismo de la computación. Actualmente existe un gran grupo de personas a nivel mundial que cree en la libertad que debe existir en los programas de cómputo. Esta libertad se asemeja a la libertad de expresión en la que es posible comunicarse sin restricción. Llevado al área de la informática, se cree en la libertad de poder ejecutar un programa, de saber cómo funciona y tener además la libertad para modificarlo, redistribuirlo y a su vez que cualquier copia pueda ser mejorada y también utilizada para cumplir con cualquier propósito.

Existe un gran número de programas computacionales que permiten trabajar con los métodos utilizados en el análisis de series de tiempo. La gran mayoría es software de paga, algunos ejemplos son EViews, SPSS, STATGRAPHICS, Minitab, entre otros. El software antes mencionado, es de gran calidad, por decir lo menos, y es muy probable que tenga funcionalidades que aún no se encuentren desarrolladas dentro del software libre. Es precisamente en este campo donde el software libre hace la diferencia; mientras que en el software de paga es preciso esperar por la nueva versión que incluya nuevas características, con el software libre dichas características se puede incluir en cualquier momento por el usuario.

De aquí que este trabajo plantea el uso de software libre aplicado a los métodos utilizados al pronosticar series de tiempo, por tal su objetivo general es el siguiente:

“Brindar elementos de apoyo didáctico que permitan pronosticar series de tiempo con y sin variación estacional utilizando R (lenguaje de programación y ambiente para cómputo estadístico y gráficos), a fin de promover el uso de software libre”.

Para cumplir con el objetivo, el trabajo actual se ha divido de la siguiente manera.

En el primer capítulo se describen brevemente algunos de los métodos existentes para pronosticar series de tiempo, métodos de promedios móviles, de suavizamiento y métodos que pueden aplicarse aún cuando la serie cuente con estacionalidad.

El segundo capítulo, da un bosquejo de lo que es el software libre, el software de código abierto y algunas reflexiones del por qué utilizarlo y promoverlo.

En el tercer capítulo se da una introducción al uso de R. Se explica brevemente cómo obtener e instalar el software, así como una rápida mirada a sus características. Se describe cómo usarlo para situaciones básicas y como sacar provecho del entorno de programación que incluye.

El cuarto y último capítulo, describe la manera de utilizar R para aplicar los métodos descritos en el capítulo uno. Dichos métodos se aplican a series de tiempo relacionadas con el desempleo, la compra de autos y el comportamiento de los matrimonios registrados en México.

Los datos utilizados en este trabajo fueron obtenidos del sitio oficial del Instituto Nacional de Estadística y Geografía, específicamente, del Banco de Información Económica (BIE).Elementos de series de tiempo

1. Elementos de series de tiempo

1.1 Serie de Tiempo

A diario, la gran mayoría de las personas, tomamos decisiones basadas en lo que a veces no alcanzamos a distinguir sino como una especie de intuición, sin percatarnos de que en realidad, nos hemos vuelto verdaderos ases en lo que algunos denominan como el arte de predecir el futuro (Cooray, 2008), esto es, los pronósticos.

Sucede cuando salimos de casa cargando el paraguas y afuera no llueve, o cargando un suéter y no hace frio. Ocurre cuando evitamos el banco los días de quincena, o mejor dicho, cuando lo visitamos más. Lo mismo en los periodos vacacionales, cuando compramos los boletos con anticipación o reservamos un cuarto de hotel en algún lugar que planeamos visitar.

Lo mismo a un nivel corporativo, desde el presupuesto destinado a un proyecto, hasta la cantidad de personas involucradas y su campo de acción. O a nivel de gobierno, cuando se planea el presupuesto de ingresos y egresos, lo mismo las proyecciones de crecimiento de la economía nacional. Claro está que en estos últimos casos conviene no seguir aquella simple intuición.

Las situaciones anteriores se relacionan con diversos aspectos que incluyen la hora del día, el estado del clima, la época del año o la situación financiera actual. Y es posible predecir su comportamiento, con algún grado de certidumbre, gracias a que contamos con datos tanto cualitativos como cuantitativos.

Un ejemplo es el calentamiento global. Los investigadores se han percatado, que la temperatura promedio mundial ha aumentado desde mediados del siglo pasado y atribuyen sus causas a los gases de efecto invernadero (National Geographic Society). Son precisamente las mediciones (datos recabados periódicamente), relacionadas con la temperatura, en este caso, las que forman una serie de tiempo. El estudio de dicha serie permitió notar que la temperatura está en aumento y también permitió estudiar la causa del fenómeno así como las medidas a seguir para frenarlo. Mediciones en los efectos del calentamiento global, que también forman una serie de tiempo, han mostrado que de seguir con la tendencia, se tendrán condiciones meteorológicas extremas en la tierra (National Geographic Society).

Como podemos ver, el estudio de las series de tiempo puede llegar a tener un gran impacto.

1.1.1 Definición

Existen varias definiciones de lo que es una serie de tiempo. Box y Jenkins (Box & Jenkins, 1970), la consideran "un conjunto de observaciones generadas de manera secuencial en el tiempo". Para Wei (Wei, 1990), son "una secuencia ordenada de observaciones". De acuerdo a la Dra. González Videgaray (González Videgaray, 2011), son "una colección de observaciones cronológicas, es decir, generadas en forma secuencial a través del tiempo". Además, todos los autores precisan que de acuerdo con el caso, las series pueden ser continuas o discretas.

1.2 Método de descomposición clásica

Una serie de tiempo puede verse como:

serie de tiempo = patrón + residuales

Si al patrón lo consideramos la tendencia, entonces a los residuales les llamamos variaciones alrededor de la tendencia. Podemos diferenciar dichas variaciones en:

- Tendencia,
- Ciclo,
- Variación estacional,
- Fluctuación aleatoria

(Cooray, 2008).

Es posible obtener los valores de la tendencia a través diversos métodos, tales como el método de semipromedios (*semi-averages*), promedios móviles (*moving averages*) y el método de mínimos cuadrados.

Este método considera que las series de tiempo pueden describirse mediante los componentes mencionados.

1.2.1 Componentes de una serie de tiempo

Tendencia (T). La tendencia es una especie de patrón, mejor descrito por la Dra. González Videgaray (González Videgaray, 2011), como "un cambio sistemático en el patrón de los datos y suele detectarse a simple vista en la gráfica de la serie original".

Variación estacional (S). Es evidente la presencia de este componente cuando en la gráfica de la serie se identifica un patrón que se repite en intervalos iguales de tiempo menores a un año, por ejemplo, las compras en un súper mercado cada quince días, la aglomeración en determinado punto de la ciudad los fines de semana debido al partido del equipo local, etc.

Ciclo (C). Los ciclos se pueden identificar cuando existen alzas y caídas en el comportamiento de los datos en periodos iguales de tiempo mayores a un año. En ocasiones resulta difícil identificar un ciclo debido a que se necesita contar con una considerable cantidad de datos recolectados a través de numerosos años.

Fluctuación aleatoria (I). Las fluctuaciones aleatorias son el resultado de los llamados choques o golpes aleatorios. Dichas fluctuaciones agrupan todo aquello que el modelo matemático no puede explicar. Para que un modelo sea bueno, las fluctuaciones aleatorias deben ser estadísticamente insignificantes (González Videgaray, 2011).

1.3 Descomposición clásica de las series de tiempo

Los modelos de descomposición clásica se distinguen como el modelo aditivo y el modelo multiplicativo.

1.3.1 Modelo Aditivo

El modelo aditivo se caracteriza por ser el resultado de la suma de los componentes de la serie de tiempo,

$$X = T + C + S + I + e$$

En el caso particular en el que cualquiera de los componentes no existiera, el valor de dicho componente es cero.

En el modelo aditivo, los ciclos y el componente estacional son independientes de la tendencia y por tal motivo, la magnitud de dichos componentes son constantes en el tiempo.

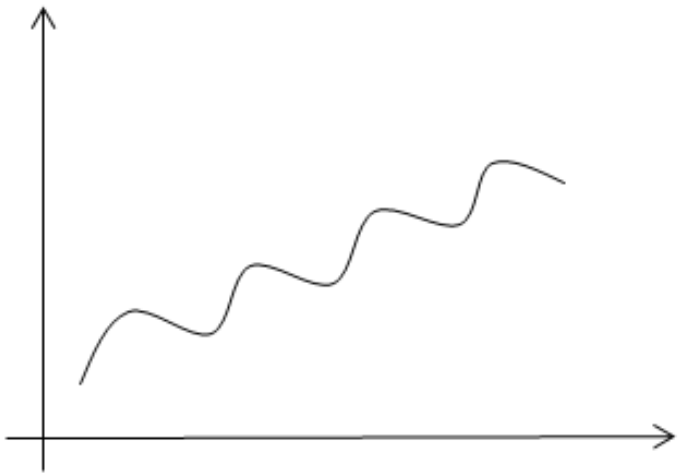


Figura 1: Modelo aditivo

Descomposición aditiva

Para la descomposición aditiva, lo primero es calcular un promedio móvil centrado (PMC) de tamaño L, donde L es el número de estaciones en el año. Al obtener un PMC en los datos, estamos aislando a la tendencia y el ciclo, $PMC = T_t + C_t$. Si sustraemos de los datos originales, el PMC, la diferencia que resulta está formada por el componente estacional y el error $S_t + e_t$. Para remover el error del último valor obtenido, se realiza un promedio para cada una de las estaciones. Estos promedios deben sumar cero, en caso contrario se deben normalizar, es decir, sustraer una constante $\sum \frac{\bar{S}_n}{L}$, de cada promedio. A continuación se desestacionaliza la serie original al restarle el componente estacional respectivo a cada periodo $d_i = x_i - S_n$. El último paso resulta al realizar un análisis de regresión a los datos desestacionalizados.

El pronóstico resulta de sumar cada uno de los estimadores calculados:

$$\hat{X}_t = T_t + S_t + C_t$$

Donde T_t es la ecuación encontrada en la regresión (Cooray, 2008).

1.3.2 Modelo multiplicativo

Los datos en el modelo multiplicativo son el resultado de la multiplicación de los distintos componentes.

$$X = T \times S \times C \times I + e$$

Si cualquier componente no existiera, su valor se asume como uno.

En este caso el factor del componente estacional es proporcional a la tendencia y por ello la magnitud de dicho componente decrecerá o se incrementará de acuerdo con el comportamiento de la tendencia.

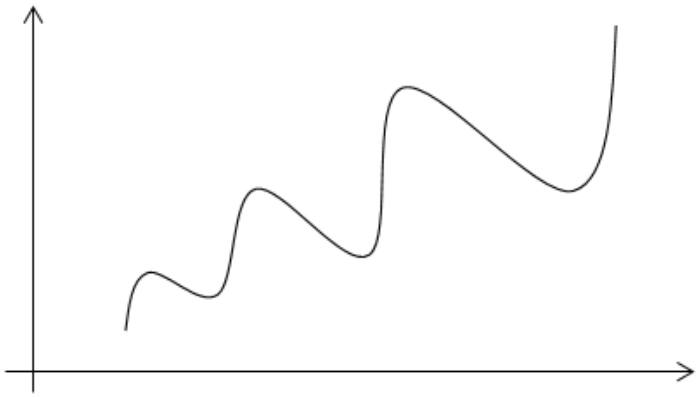


Figura 2: Modelo multiplicativo

Descomposición multiplicativa

El método de descomposición multiplicativa guarda una gran similitud con el método de descomposición aditiva.

Primero, se calcula un promedio móvil centrado (PMC) de tamaño L. Se dividen los datos originales entre el PMC aislándose de esta forma al componente estacional con el error, $S_n e_t$. Para aislar el error del componente estacional se calculan los promedios $\sum \frac{\bar{S}_n e_t}{L}$ para cada estación. En el modelo multiplicativo, la suma de estos promedios debe ser igual a L, en caso contrario se deben normalizar, esto implica multiplicar cada promedio por la constante $L/\sum \bar{S}_n$. Para desestacionalizar esta, se divide entre el componente estacional correspondiente, $d_t = X_t/S_t$. Por último se realiza un análisis de regresión en los datos desestacionalizados.

El pronóstico se obtiene al reemplazar cada componente en la ecuación

$$X_t = T_t S_t C_t e_t$$

donde T_t es la ecuación encontrada en la regresión (Cooray, 2008).

Puede resultar de ayuda al analista conocer la facilidad con la que cualquiera de los componentes se puede obtener. Por ejemplo, la variación estacional de una serie de tiempo es el componente más fácil de identificar debido a lo constante de su repetición.

Los ciclos por otro lado resultan ser el componente más difícil de modelar debido a la gran cantidad de datos necesarios para reconocerlos. Para poder obtener un buen modelo que incluya ciclos quizá sea necesario contar con alrededor de 40 años de datos. Debido a eso, cuando se cuente con pocos datos, los ciclos se pueden considerar parte de las fluctuaciones aleatorias. Dicho de otra manera, cuando se cuenten con pocos datos, el concepto de ciclo resulta sin utilidad.

1.4 Promedios móviles simples

Este método difiere a los de descomposición clásica en que no busca separar individualmente los componentes de la serie de tiempo. A su vez, pondera con iguales valores a los datos históricos que el dato a pronosticar. Este método es ideal para llevar a cabo pronósticos a corto plazo.

Para aplicar este método se asume que la serie de tiempo es estacionaria. En el caso en el que la serie no lo fuera, el método no toma en cuenta la tendencia, generando valores pronosticados muy alejados de los datos reales.

Se llaman promedios móviles ya que para calcular el cuarto valor de la serie, es necesario promediar los tres primeros valores, para calcular el quinto valor, se promedian los valores del segundo al cuarto y así sucesivamente.

Las siguientes ecuaciones reflejan lo anterior,

$$F_{t+1} = \frac{x_t + x_{t-1} + x_{t-2} + \cdots + x_{t-N+1}}{N}$$

$$F_{t+1} = \frac{1}{N} \sum_{i=t}^{t-N+1} x_i$$

1.5 Promedios móviles dobles

Este método realiza un nuevo promedio a los datos obtenidos aplicando el método de promedios móviles simples, de ahí su nombre. El método permite pronosticar series de tiempo no estacionarias. Es posible utilizar este método en series de tiempo que no tengan una tendencia lineal pero de igual forma que el método anterior, sólo es bueno para realizar pronósticos a corto plazo.

Las ecuaciones para poder aplicar el método de los promedios móviles dobles son las siguientes.

$$a_t = 2M'_t - M''_t$$

Donde M'_t es el promedio móvil simple y M''_t el promedio móvil doble.

La siguiente ecuación representa un factor de ajuste.

$$b_t = \frac{2}{N-1}(M'_t - M''_t)$$

Con la siguiente expresión podríamos aproximar cualquier valor de nuestra serie de tiempo.

$$F_{t+1} = a_t + b_t$$

Y finalmente la expresión,

$$F_{t+m} = a_t + mb_t$$

permite pronosticar “m” periodos en el futuro.

1.6 Suavizamiento exponencial

1.6.1 Suavizamiento exponencial simple

El método de suavizamiento exponencial se basa en promediar las observaciones anteriores de una serie de forma decreciente (exponencial). El método pondera las observaciones dando más peso a las últimas que a las primeras teniendo en cuenta que estas son de mayor relevancia para el pronóstico. Así el peso α es asignado a la observación más reciente, $\alpha(1-\alpha)$ a la siguiente, $\alpha\ (1\text{-}\alpha)^2$ a la próxima y así sucesivamente.

En el método de suavizamiento exponencial, el pronóstico para el periodo actual se basa en la suma del pronóstico anterior más α veces el error en el pronóstico anterior $(x_t - \hat{x}_t)$, donde x_t es el valor actual real y $\hat{x}_t$ es el valor actual estimado.

Por tal, la ecuación $\hat{x}_{t+1} = \hat{x}_t + \alpha e_t$, que representa lo anterior, puede ser vista como,

$$S_t = S_{t-1} + \alpha(x_t - S_{t-1})$$

Donde S_t es el nuevo valor estimado para el siguiente periodo, hecho en el periodo actual.

S_{t-1} es el valor estimado para el periodo actual, hecho en el periodo pasado.

x_t es el valor real en el actual periodo.

$(x_t\text{-}S_{t-1})$ es el error estimado para el actual periodo.

α es el peso asignado.

La ecuación de suavizamiento exponencial puede agruparse de la siguiente manera,

$$S_t = \alpha(x_t) + (1-\alpha)S_{t-1}$$

Para poder aplicar el método de suavizamiento exponencial a una serie de tiempo, es necesario que dicha serie sea estacionaria.

1.6.2 Suavizamiento exponencial doble

Este método, también llamado método de Brown, es utilizado para pronosticar series de tiempo con tendencia lineal.

Las técnicas para el doble suavizamiento exponencial son similares a las del suavizamiento exponencial simple.

Si una serie de tiempo contiene una tendencia lineal, es posible modelarla a través de la ecuación de regresión lineal,

$$\hat{x}_t = a_t + b_t(T)$$

Donde a_t representa la diferencia entre los valores suavizados,

$$a_t = 2S_t^{(1)} - S_t^{(2)}$$

b_t representa el ajuste de la pendiente,

$$b_t = \frac{\alpha}{1-\alpha}(S_t^{(1)} - S_t^{(2)})$$

T representa los periodos en el futuro a pronosticar.

$S_t^{(1)}$ y $S_t^{(2)}$ representan los valores del suavizamiento exponencial simple y el suavizamiento exponencial doble, respectivamente.

1.7 Método de Winters

El método de Winters aplica las técnicas de suavizamiento exponencial, lineal y estacional utilizando una constante de atenuación para cada caso. Cada constante puede ser ajustada independientemente de las otras dos.

1.7.1 Método aditivo de Winters

El método aditivo de Winters, puede ser empleado para pronosticar series de tiempo que contengan tendencia lineal y variación estacional aditiva. Este método es útil para actualizar los parámetros obtenidos en el modelo de descomposición aditivo de acuerdo con las siguientes ecuaciones,

Ecuación para estimar el nuevo valor de la serie suavizada.

$$a_t = \alpha\big(x_t - S_t(t-L)\big) + (1-\alpha)(a_{t-1} + b_{t-1})$$

Donde a_t es el nuevo valor estimado de la serie suavizada.

α es la constante de atenuación de los datos.

$x_t - S_t(t-L)$ es el valor real sin variación estacional para el periodo t.

a_{t-1} es el valor estimado de la serie suavizada encontrado en el periodo $t-1$.

b_{t-1} es el valor estimado de la pendiente encontrado en el periodo $t-1$.

Ecuación para estimar la pendiente.

$$b_t = \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1}$$

Donde b_t es el nuevo estimado de la pendiente.

β es la constante de atenuación para la pendiente.

$a_t - a_{t-1}$ la diferencia entre el nuevo valor estimado de la serie suavizada y el anterior.

b_{t-1} el valor estimado anterior de la tendencia, encontrado den el periodo $t-1$.

Ecuación para estimar el componente estacional.

$$S_{t+L}(t) = \gamma(x_t - a_t) + (1-\gamma)S_t(t-L)$$

Donde $S_{t+L}(t)$ es el nuevo valor estimado del componente estacional en el periodo t, mientras que

γ es la constante de atenuación de la estimación de la estacionalidad.

$x_t - a_t$ es una medida del valor real de la variación estacional en los datos, obtenido al sustraer el nuevo valor del estimado de la serie suavizada del verdadero valor.

$S_t(t-L)$ es el valor estimado de la variación estacional encontrado en el periodo $t-L$ (del mismo periodo pero de un año anterior).

Después de haber calculado los valores anteriores, es posible pronosticar la serie con la siguiente ecuación,

$$\hat{x}_{t+1}(t) = [a_t + b_t] + S_{t+1}(t+1-L)$$

Donde $\hat{x}_{t+1}(t)$ es el pronóstico para el periodo $t+1$.

a_t es el valor estimado de la serie suavizada en el periodo t.

b_t es el valor estimado de la pendiente en el periodo t.

$S_{t+1}(t+1-L)$ es el valor estimado para la estación $t-1$ hecho en el periodo $t+1-L$ (un año antes).

1.7.2 Método Multiplicativo de Winters

Las ecuaciones del modelo multiplicativo de Winters son muy parecidas a las del modelo aditivo. Si el factor estacional no es una cantidad constante, sino una proporción constante o un porcentaje de los datos, es apropiado utilizar el método multiplicativo.

Para desestacionalizar los datos, los dividimos entre el factor estacional apropiado.

La ecuación del método multiplicativo puede verse de la siguiente manera.

$x = TSe_t$, donde $T = a + b_t$

Para actualizar el nivel del promedio suavizado utilizamos la siguiente ecuación,

$$a_t = \alpha\left[\frac{x_t}{S_t(T-L)}\right] + (1-\alpha)(a_{t-1} + b_{t-1})$$

Donde a_t es el nuevo valor estimado de la serie suavizada en el periodo t.

α es la constante de atenuación de los datos.

$\left[\frac{x_t}{S_t(T-L)}\right]$ es el valor real desestacionalizado para el periodo t.

a_{t-1} es el valor estimado anterior de la serie suavizada, encontrado en el periodo $t-1$.

b_{t-1} es el valor estimado anterior de la tendencia suavizada, encontrado en el periodo $t-1$.

Para actualizar el valor de la tendencia, utilizamos la siguiente ecuación,

$$b_t = \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1}$$

Donde b_t es el nuevo valor estimado para la tendencia en el periodo t.

β es la constante de atenuación de la tendencia.

$(a_t - a_{t-1})$ es la diferencia entre el valor estimado actual y el anterior de la serie suavizada.

b_{t-1} es el valor estimado suavizado anterior de la tendencia encontrado en el periodo $t-1$.

La ecuación para actualizar el componente estacional es la siguiente,

$$S_{i_{(t+1)}} = \gamma\left[\frac{x_t}{a_t}\right] + (1-\gamma)S_{i_t}(t-L)$$

Donde $S_{i_{(t+1)}}$ es el nuevo valor estimado para el componente estacional en el periodo t.

γ es la constante de atenuación del componente estacional.

$\left[\frac{x_t}{a_t}\right]$ es una medida de la variación estacional real en los datos, obtenida al dividir el valor real entre el valor estimado de la serie suavizada.

$S_{i_t}(t-L)$ es el estimador suavizado anterior del componente estacional encontrado en el periodo $t-L$ (el mismo periodo, un año antes).

La ecuación para obtener un pronóstico de un periodo hacia el futuro es la siguiente,

$\hat{x}_{t+1}(t) = [(a_{t-1} + b_{t-1})]S_{i_{(t+1)}}(t+1-L)$

Donde $\hat{x}_{t+1}(t)$es el pronóstico para el periodo $t+1$.

a_{t-1} es el valor estimado de la serie suavizada en el periodo $t-1$.

b_{t-1} es el valor estimado de la tendencia suavizada en el periodo $t-1$.

$S_{i_{(t+1)}}(t+1-L)$ es el valor estimado suavizado para la estación $t+1$, obtenido en el periodo $t+1-L$, un año atrás.

2. El software libre y sus ventajas

2.1 Free Software Foundation (FSF)

La Free Software Foundation, es una fundación no lucrativa cuya misión se centra en promover el uso libre de las computadoras y en defender la libertad de los usuarios de software libre (*free software*), como la misma fundación se define en su sitio web (Free Software Foundation, 2014).

El trabajo de la FSF se centra en mantener documentos históricos acerca de la filosofía y de la definición de software libre, con el propósito de mostrar, en forma clara, las condiciones que debe cumplir un programa para ser considerado software libre.

También es patrocinador del proyecto GNU, el cual es un esfuerzo para proveer de un sistema operativo completo, licenciado como software libre.

La creación de un sistema operativo totalmente libre (proyecto GNU), fue iniciado por Richard M. Stallman en 1983 y en 1985 se fundó la FSF.

2.2 ¿Qué es el software libre?

La FSF define al software libre como aquél que brinda la libertad al usuario de compartirlo, estudiarlo y modificarlo. Le llaman software libre dado que su uso es libre.

El proyecto GNU lo define de manera similar (GNU, 2014), abundando en el hecho de que el software libre brinda al usuario la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el programa.

La definición se actualiza cuando hay necesidad de clarificarla o cuando es necesario resolver problemas relacionados con cuestiones delicadas.

Como está descrito en la página del proyecto GNU (GNU, 2014), el software es libre si los usuarios cuentan con las cuatro libertades esenciales:

- Libertad 0. Libertad de ejecutar el programa para cualquier propósito.
- Libertad 1. Libertad de estudiar cómo funciona el programa y cambiarlo para que el programa haga lo que el usuario quiera.
- Libertad 2. Libertad de redistribuir copias para ayudar a su prójimo.
- Libertad 3. Libertad de distribuir copias de sus versiones modificadas a terceros.

De aquí, es posible hacer aclaraciones de ciertos puntos relacionados con las libertades en pro de un mejor entendimiento de su significado.

Es importante conceptualizar el software libre, como la “libertad de expresión” y no como “barra libre”. Esto puede parecer confuso, pero si ponemos atención en las libertades 2 y 3, no se menciona que la distribución y/o redistribución deban hacerse de manera gratuita. Lo que las libertades 2 y 3 dicen, es que uno es libre de distribuir y/o redistribuir el software, ya sea a cambio

de una remuneración o no, sin necesidad de ningún permiso. He ahí la libertad y no necesariamente que se trate de software gratis (aunque sí lo es).

La libertad 1 no condiciona dar a conocer los cambios hechos en el software ni mucho menos limita el uso una vez hechos.

La libertad 0 no excluye a persona u organización de ejecutar el software para cualquiera que sea su fin y no condiciona el uso del tipo de sistema de cómputo. A su vez, ni persona ni organización pueden condicionar la ejecución del mismo.

Es indispensable tener acceso al código fuente, ello para que las libertades 1 y 3 no carezcan de sentido.

Se conocen algunas prácticas en las que el término libre puede llegar a ser vago, tales como el brindar acceso al código fuente, pero este está ofuscado o versiones modificadas del software pero que privan el uso de una versión propia. Otro caso particular es cuando la licencia del software limita el uso de módulos adicionales con derechos de autor, a menos que el usuario sea el titular.

Tampoco se debe confundir al software libre con software de código abierto ya que, aunque evocan algo similar, el código abierto no necesariamente se refiere a la libertad del software pero es de recalcar que es una de las condiciones para que un programa se califique como libre.

2.3 ¿Qué es el software de código abierto?

El software de código abierto, como lo define The Open Source Initiative (OSI) (OSI, 2014), es el software que puede ser compartido, usado y modificado gratuitamente por cualquiera. Además, es aquel software del que se tiene acceso al código fuente y que cumple los siguientes criterios:

1. Redistribución libre. La licencia no debe restringir a ninguna de las partes de vender o regalar el software como componente de una distribución que contenga programas de diferentes fuentes.
2. Código fuente. El programa debe incluir el código fuente y debe permitir la distribución en forma de código fuente o en su forma compilada. Cuando el código fuente no acompañe al programa, debe existir una forma bien publicitada acerca de cómo obtenerlo por no más que un costo de reproducción razonable. Descargarlo desde internet no debe tener costo alguno.
3. Trabajos derivados. La licencia debe permitir modificaciones y trabajos derivados, además permitir que estos sean distribuidos bajo los mismos términos de la licencia original.
4. Integridad del código fuente del autor. La licencia puede restringir la distribución del código fuente modificado únicamente si permite la distribución de “parches” con el código, con el propósito de modificar el programa en tiempo de compilación. La licencia, explícitamente, debe permitir la distribución de software creado a partir de código fuente modificado. La licencia puede requerir trabajos derivados para poder tener un nombre o número de versión distinto del software original.

5. No discriminar personas o grupos. La licencia no debe discriminar a persona o grupo de personas.
6. No discriminar campos de aplicación. La licencia no debe restringir a nadie de usar el programa en algún campo de aplicación específico.
7. Distribución de licencia. Los derechos ligados al programa, deben aplicar a todos a quienes se ha distribuido sin la necesidad de una licencia adicional.
8. La licencia no debe ser específica de un producto. Los derechos ligados a un programa no deben depender de que el programa sea parte de una distribución de software particular. Si el programa es extraído de dicha distribución y usado y distribuido en los términos de la licencia de ese programa, todas las partes a las cuales se distribuyó, deben contar con los mismos derechos como aquellos conferidos a la distribución original.
9. La licencia no debe restringir otro software. La licencia no debe poner restricciones en otro software que sea distribuido junto con el software licenciado.
10. La licencia debe ser neutral tecnológicamente. Ninguna disposición de la licencia puede basarse en alguna tecnología individual o estilo de interface.

El criterio anterior permite que aquel software licenciado como de código abierto sea usado, distribuido y modificado para cumplir con los requisitos particulares de persona u organización.

En el caso particular de R (The R Project, 2014) , el código fuente es distribuido bajo los términos de la licencia pública general GNU (GNU General Public License). Este hecho es de gran trascendencia ya que gracias a esto es posible que numerosas personas alrededor del mundo colaboren en la creación de miles de paquetes de uso general y específico.

2.3.1 El ambiente de R

El término ambiente pretende caracterizar a R como un sistema completamente planeado y coherente (The R Project, 2014).

R está compuesto por una suite integrada de software para la manipulación de datos, cálculos y visualización de gráficos, que incluye,

- Manejo efectivo de datos y herramientas de almacenamiento.
- Una suite de operadores para cálculos con arreglos.
- Una larga y coherente colección integrada de herramientas para análisis de datos.
- Herramientas gráficas para análisis de datos y visualización en pantalla o de forma física.
- Un lenguaje de programación simple y efectivo.

El hecho de que R sea un software de código abierto ha permitido extender su funcionalidad a través de paquetes.

2.4 Aplicaciones de software libre

Actualmente existe una enorme cantidad de aplicaciones de software libre que van desde aplicaciones para escuchar música, reproducir videos, navegadores web, procesadores de texto, hojas de cálculo, hasta sistemas gestores de bases de datos, lenguajes de programación y desde luego R.

Hablaremos de algunas de las aplicaciones más conocidas y de sus usos.

LibreOffice. LibreOffice es una suite de aplicaciones que incluye procesador de textos, hojas de cálculo, presentaciones y edición de gráficos. Una de las ventajas de LibreOffice es la posibilidad de ejecutar la suite en sistemas operativos libres y también en versiones para Windows y Mac OS X (LibreOffice, 2014).

MySQL. MySQL es el gestor de base de datos libre más popular debido a que es confiable, fácil de utilizar y tiene buenos tiempos de respuesta. Empresas como Google y Facebook utilizan MySQL en sus operaciones diarias (Oracle, 2014).

Firefox. Frirefox es de uno de los navegadores web más utilizados. Entre sus capacidades se cuentan un alto rendimiento, seguridad avanzada y un alto grado de personalización (mozilla, 2014).

GIMP. GIMP es un programa de manipulación de imágenes (GNU Image Manipulation Program). GIMP es un software que lo mismo puede ser utilizado como el programa “Paint” de Windows, de la misma manera en que puede ser utilizado como un programa experto para el retoque de imágenes o como un convertidor de formatos (Gimp, 2013).

PHP. PHP es un lenguaje de programación diseñado para el desarrollo web. También es posible utilizar PHP como un lenguaje de propósito general. Es conocido por ser un lenguaje de codificación del lado del servidor (Php, 2014).

Como podemos ver, existen aplicaciones libres para cualquier propósito. Hablemos entonces de algunas de las ventajas de su uso y promoción.

2.5 Ventajas de usar y promover software libre

Además de la propia utilidad que pueda tener una aplicación de software libre, existen a su alrededor diferentes opiniones del por qué preferir su uso sobre el software propietario. Puntualicemos que el software libre no significa software no comercial, más allá de eso, el software libre debe estar disponible comercialmente, tanto para su uso como para su distribución. Recuérdese lo abordado en las libertades 2 y 3. Esta aclaración es útil ya que por lo general cuando se habla de las ventajas del software libre se contrastan sus virtudes contra lo negativo del software propietario. Para ser más claros, existen actualmente versiones de software libre licenciadas bajo un esquema de subscripción anual, las cuales cobran por el soporte, funciones especiales y por mantener la misma versión a lo largo de la subscripción, entre otras características.

Teniendo en cuenta lo anterior, podemos comenzar por mencionar la enorme aceptación frente al uso de software libre entre académicos e investigadores alrededor del mundo. Además de un gran número de entusiastas que están dispuestos a ayudar en numerosos foros de internet a cambio de un gracias en el mejor de los casos. Aquí contrasta el hecho en el que grandes corporaciones necesitan un incentivo monetario, directa o indirectamente, para ayudar al usuario común.

Otra gran ventaja es el hecho de que los productos libres son de gran calidad, inclusive mayor que mucho del software de propietario usado cotidianamente, a pesar de que en algunos sectores goce de mala reputación. Por poner dos ejemplos conocidos, MySQL y Firefox, que como se mencionó en la sección anterior, son de uso común en grandes empresas. Pero tampoco debemos ignorar que al inicio algunos de los productos pueden carecer de cualquiera de las ventajas que con el tiempo consiguen.

En el ámbito estudiantil, es frecuente el uso de software propietario, lo que sin lugar a dudas lo fortalece, limitando las opciones e incrementado la inversión de cualquier organización en dicho software, inversión que bien se puede destinar a mejores causas.

Otra poderosa razón para utilizar el software libre es su mantenimiento ya que por lo regular las versiones se actualizan con mayor frecuencia que la del software propietario. Probablemente esto se deba a la lentitud de ciertas empresas de software frente a la pro actividad de las organizaciones de software libre.

Es cierto que además de las anteriores existen muchas más razones por las cuales conviene el uso de software libre, pero probablemente las mejor justificadas son el hecho de que mientras con el software de propietario los usuarios no tienen la libertad de saber cómo funciona, con el software libre es posible saberlo, además de mejorarlo y poder distribuirlo.

La última razón es el costo; el software libre es gratis y, aunque parezca esta su debilidad, con lo anterior se ha intentado mostrar que esta es su mayor fortaleza.

3. Uso de R y manejo de funciones

3.1 ¿Qué es R?

R es un lenguaje de programación y un ambiente para cómputo estadístico y gráficos que provee de una amplia variedad de técnicas estadísticas y gráficas, tales como, modelado lineal y no lineal, pruebas estadísticas, análisis de series de tiempo, etcétera (The R Project, 2014).

3.2 Descarga de R

Para poder descargar e instalar R es necesario acceder a la página: http://www.r-project.org/.

Después de ingresar, buscamos la liga **download R** y escogemos un sitio de descarga. Es recomendable descargar R desde las ligas de México, sólo por practicidad. Una vez que escogimos el sitio de descarga, elegimos la opción que vaya de acuerdo con nuestro sistema operativo (SO).

En el caso de Linux, es posible encontrar R desde el "packet manager" aunque probablemente no sea la última versión, por tal es recomendable acceder al sitio oficial.

Para seguir adelante con el proceso de descarga, damos clic en la opción **install R for the first time**, y **Download R versión actual de R for Windows**. En esta última página es usual encontrar ligas de ayuda y de preguntas frecuentes. No tiene mucho caso dar indicaciones detalladas de las opciones disponibles, ya que los sitios en internet tienden a cambiar para mejorar la experiencia del usuario, y dar indicaciones precisas podría llevar a la confusión.

Al dar clic en **Download R version actual de R for Windows**, usualmente el navegador de internet comienza la descarga automáticamente, así que queda esperar y ubicar el archivo descargado.

3.3 Instalación de R

Los sistemas operativos actuales, por lo regular, se cercioran de que el usuario desee instalar un nuevo programa al dar doble clic sobre un archivo ejecutable y después de aceptar comienza las opciones de instalación de R. Lo primero es elegir el idioma en el que R mostrará ciertas opciones, a pesar de elegir español, siempre encontraremos mensajes en inglés.

Figura 3: Selección de idioma.

El sistema pregunta si se desea continuar con la instalación. Si es así, recomienda cerrar el resto de los programas que se encuentren en ejecución.

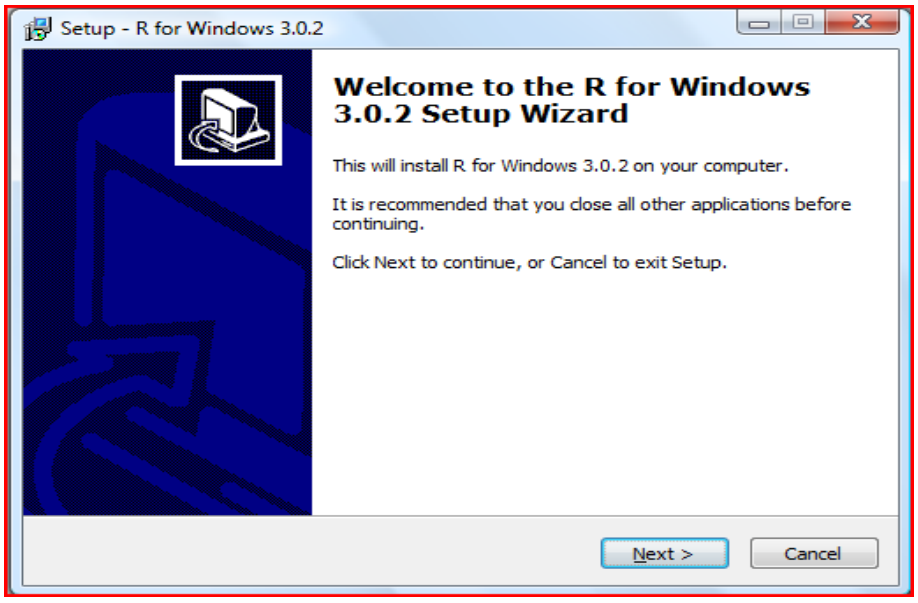


Figura 4: Mensaje de bienvenida del instalador, dar clic en next.

En la siguiente pantalla, aparecen los términos de la licencia del software.

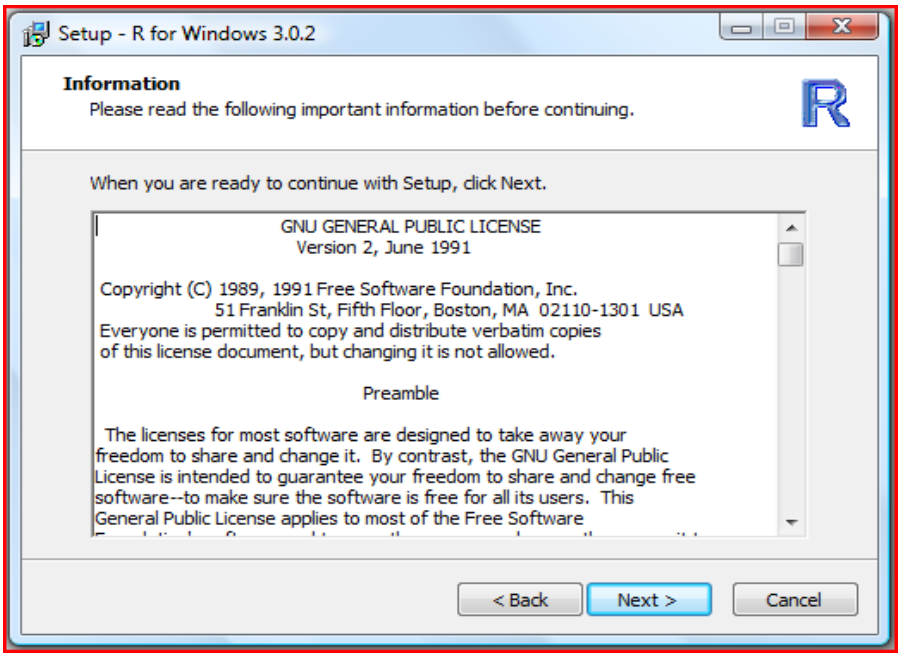


Figura 5: Términos de la licencia.

El instalador sugiere una ruta de instalación. Es recomendable seguir los parámetros sugeridos.

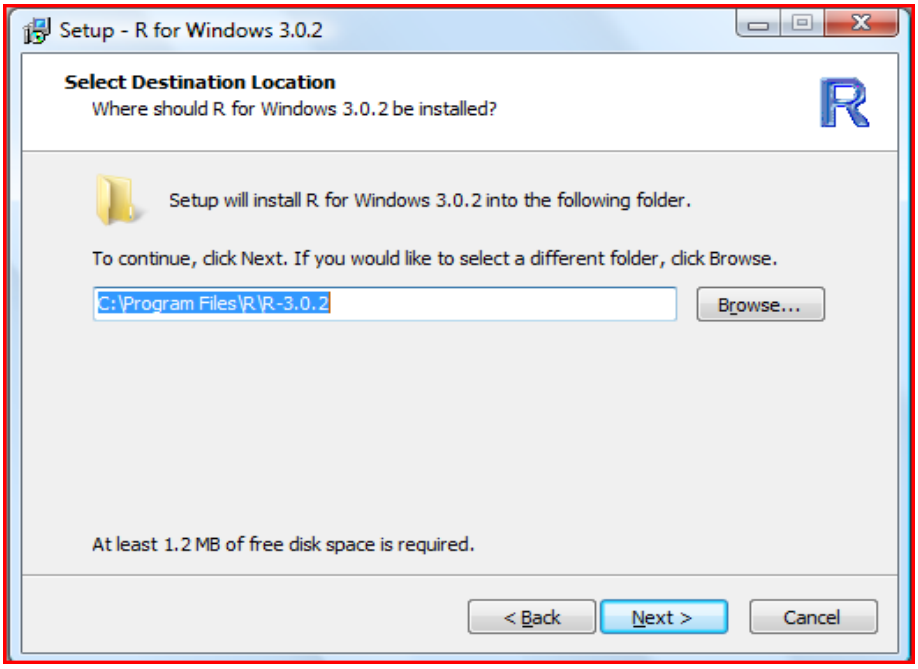


Figura 6: Ruta de instalación por defecto.

En la siguiente pantalla, el sistema pregunta por la versión que se desea instalar, la versión de 32 o 64 bits. Esto varía de acuerdo con el tipo de procesador y la versión del sistema operativo con que se cuente. Si existe duda, la opción de 32 bits funciona bien con cualquiera de las dos alternativas mencionadas.

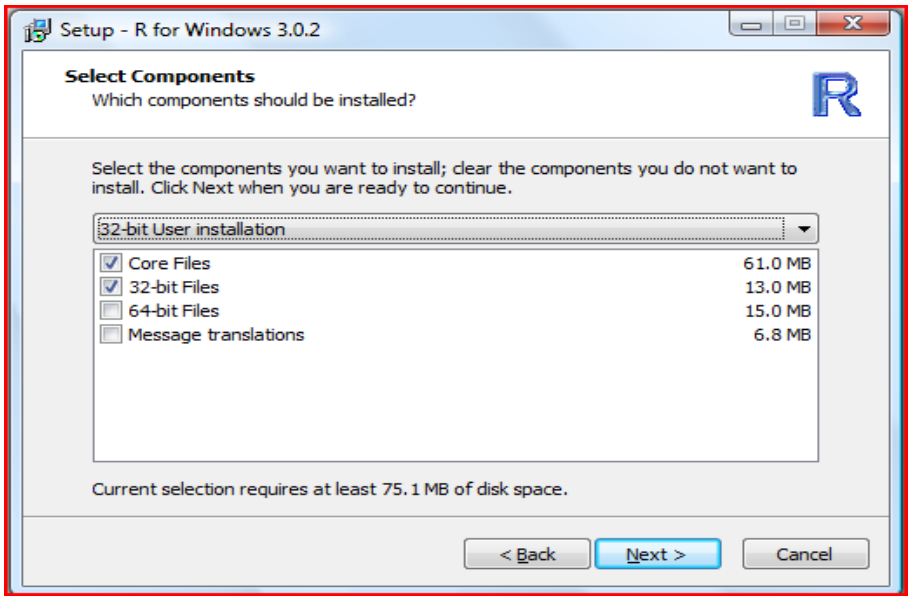


Figura 7: Componentes a instalar.

Las siguientes elecciones, hacen referencia a las opciones de inicio. Se recomienda seleccionar la opción por default (“No”).

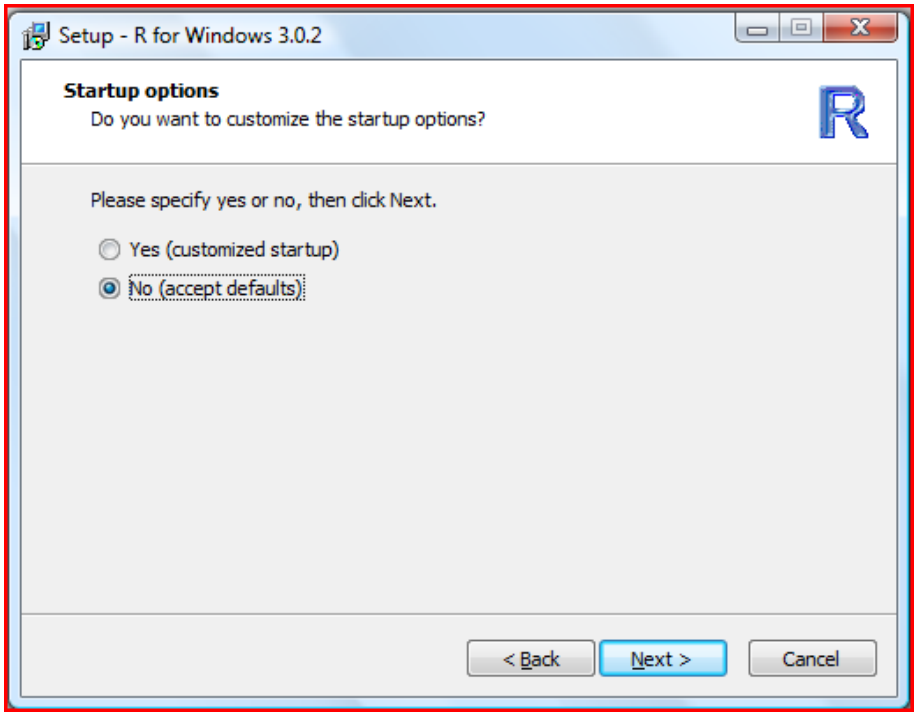


Figura 8: Opciones de inicio.

A continuación el instalador pregunta si se desea crear un atajo en la carpeta de inicio. Por lo regular los programas crean este por defecto.

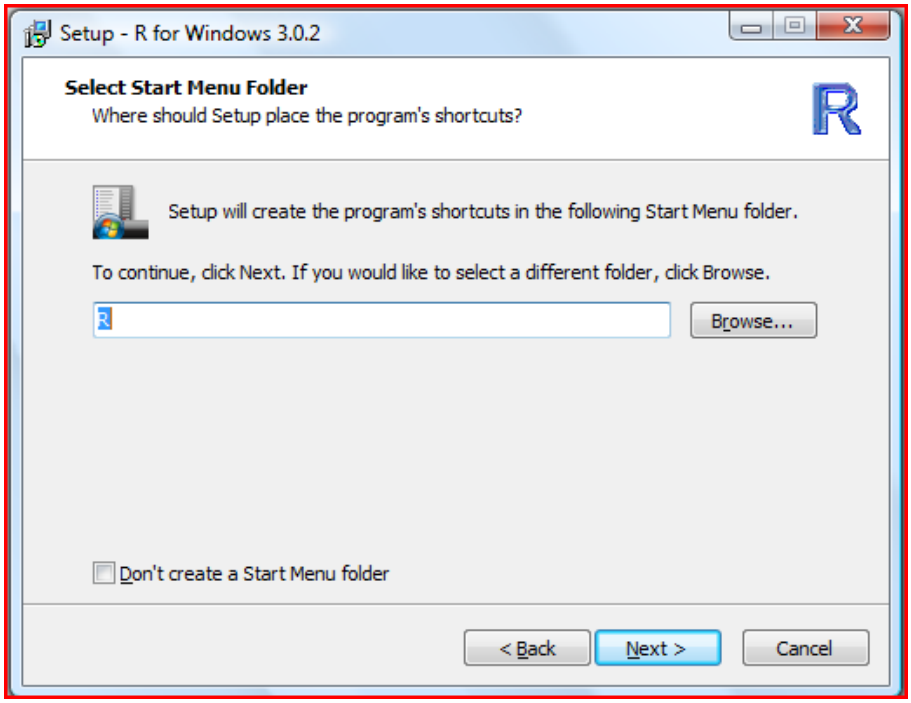


Figura 9: Atajo en la carpeta de inicio.

En la siguiente captura, el instalador pregunta si se desea crear un ícono de acceso directo en el escritorio y si se desea asociar los archivos .RData con R. Estas opciones son altamente recomendables.

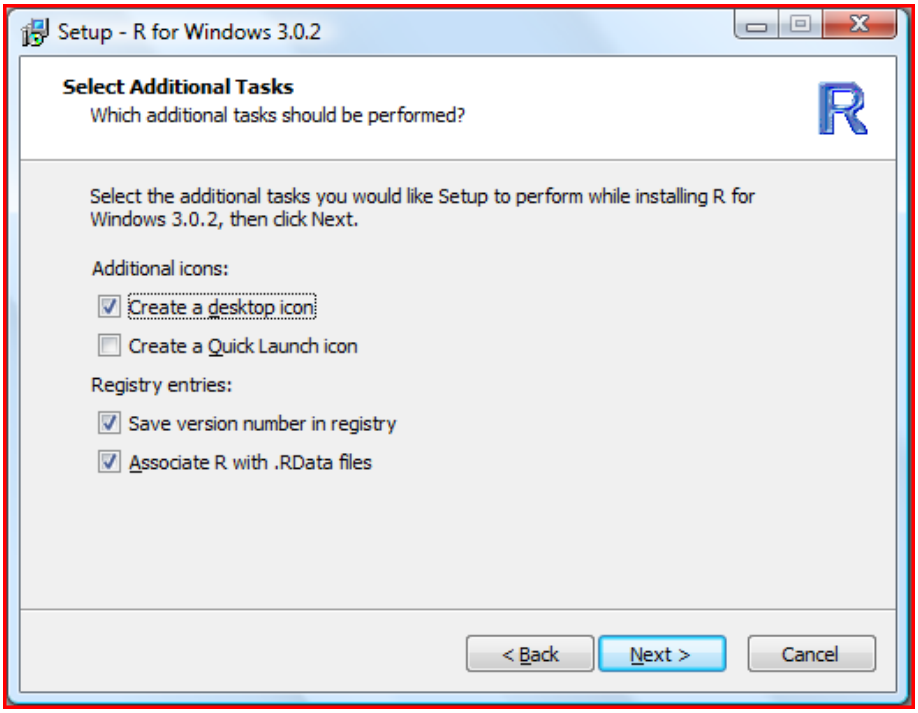


Figura 10: Crear ícono en el escritorio y asociar archivos .RData.

Después de aceptar dichos parámetros, la instalación comienza.

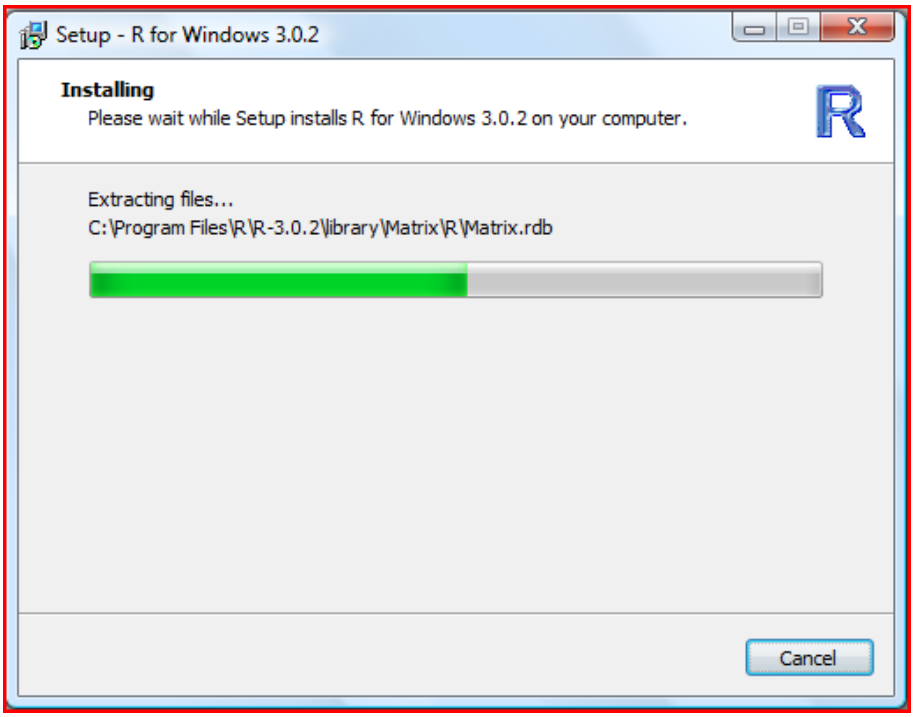


Figura 11: Instalación de R en progreso.

Si la instalación se ha llevado a cabo con éxito, el instalador muestra la siguiente pantalla.

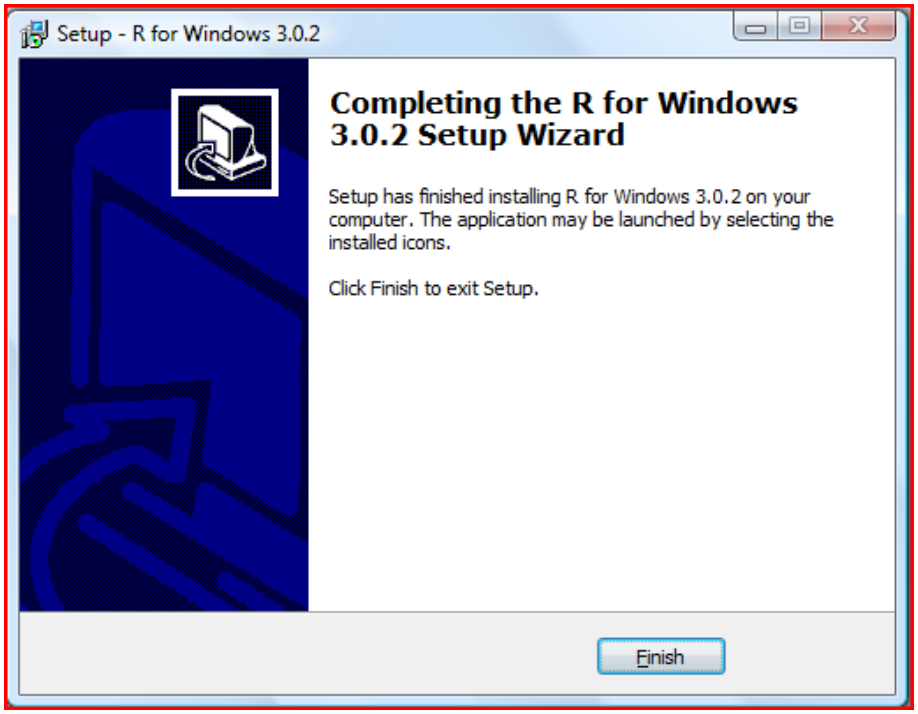


Figura 12: Instalación exitosa.

3.4 RStudio

RStudio es un entorno o ambiente de desarrollo integrado (Integrated Development Enviroment, IDE por sus siglas en inglés).

En pocas palabras, un IDE es una herramienta que permite la fácil interacción con un lenguaje de programación, dotando al usuario de un conjunto de herramientas enfocadas a mejorar la experiencia al proveer de una interface amigable, esto es, por lo regular, una interface gráfica.

En este sentido, RStudio es un IDE de R que permite trabajar con R de una manera más práctica que con la interface de R por defecto.

Existen numerosos IDEs para R, el uso de cualquiera de ellos o la interface por defecto de R es una decisión de los usuarios.

3.4.1 Instalación de RStudio IDE

Para instalar RStudio es necesario contar con una versión de R superior a 2.11.1. No deberíamos tener problemas si descargamos R de acuerdo con las instrucciones anteriores.

Los pasos para instalar RStudio son los siguientes,

1.- Ingresar al sitio http://www.rstudio.com/

2.- Ubicar la opción **Downloadnow**.

3.- Dar clic en la opción **DownloadRStudio Desktop**.

El sitio de internet sugiere una versión de acuerdo con el sistema operativo que se esté usando. Si se desea descargar una versión diferente, se encuentran enlistadas todas las versiones disponibles bajo el título **AllPlatforms.**

La descarga comienza automáticamente después de dar clic en la opción que corresponda a nuestro sistema operativo. Como en el caso anterior, las instrucciones pueden cambiar sutilmente con el tiempo, pero el proceso de descarga resulta intuitivo.

3.5 Introducción a R

Cuando alguien aprende por primera vez un lenguaje de programación es usual comenzar con el programa "hola mundo". En R este programa no tiene el mismo significado dado que basta teclear "hola mundo" en la consola de R para poder visualizar el mismo mensaje. En R es preferible comenzar con algo distinto, algo más acorde a la estadística y que además nos permita cerciorarnos de que R está funcionando correctamente.

Por ejemplo:

Escribimos en la interface de R o en el IDE que se esté utilizando:

```
> mean(1:9)
```

y obtendremos como resultado

```
[1] 5
```

Lo que hicimos sencillamente fue calcular la media aritmética de los números del 1 al 9, cuyo valor es 5. En este caso `mean()`, es el nombre de la función que calcula la media, 1:9 es el argumento de esta función. El operador **:** crea un vector de números, empezando en el valor a la izquierda de los dos puntos, en este caso el 1, hasta el valor a la derecha, 9, incrementado el valor de uno en uno. Trataremos más del lenguaje a continuación.

3.5.1 Hablemos de cómo obtener ayuda en R

Existen varias formas de obtener ayuda en R. Primero, si se desea información acerca de una función o tipo de dato y se conoce el nombre, basta con escribir el símbolo `?`, seguido del nombre de la función.

Para encontrar alguna función de la cual se tenga algún indicio, se escribe `??`, seguido de una palabra relacionada con la función deseada.

Caracteres especiales, palabras reservadas y si se escribe más de una palabra para la búsqueda, deben escribirse entre comillas e ir precedidas, de igual forma por `??`.

Por ejemplo:

```
> ??mean
```

```
> ??"if"
```

También es posible obtener ayuda utilizando las funciones `help()` y `help.search()`. Realizan la misma operación que `?` y `??`, la diferencia es que para hacer uso de estas funciones, es necesario escribir los argumentos entre comillas, de la siguiente manera:

```
> help("mean")
> help.search("if")
```

Nota: el operador `??` sólo es capaz de encontrar elementos dentro de paquetes de R que ya se encuentren instalados. Si lo que se quiere es buscar dentro de cualquier paquete, esté o no instalado, se puede utilizar la función `RSiteSearch()` que corre un script que direcciona la búsqueda hacia la página: http://search.r-project.org en donde se pueden encontrar los elementos buscados.

3.6 Características básicas de R

R tiene la capacidad de cualquier calculadora, esto significa que puede realizar operaciones tan básicas como una suma.

En ocasiones, R, es llamado de "naturaleza vectorial", ya que es capaz de realizar operaciones con vectores sin necesidad de ciclos (loops). El usuario no debe declarar explícitamente un ciclo si se trata de una suma, resta, multiplicación, división, módulo e inclusive si la operación es lógica. Esto es cierto, incluso si los operandos son vectores y constantes.

Ejemplo, si se desea calcular una suma de vectores, basta con aplicar el operador suma, ya sea sobre vectores almacenados en una variable o vectores que se creen en tiempo de ejecución:

```
> 1:5 + 6:10
[1]  7  9 11 13 15
```

Debemos asegurarnos de que los vectores sean de la misma magnitud, esto es, que cuenten con el mismo número de elementos, de otra forma R mostrará un mensaje de error. En general, se siguen las reglas al realizar operaciones con vectores.

Al igual que varias aplicaciones, R tiene su talón de Aquiles. Al realizar operaciones con números de coma flotante, los resultados llegan a ser una aproximación a las respuestas reales. Dichos errores se vuelven evidentes al comparar operaciones de las cuales tenemos conocimiento de los resultados correctos.

Por ejemplo, al hacer la siguiente comparación, utilizando el operador ==, se obtiene lo siguiente:

```
> sqrt(2)^2 == 2
[1] FALSE
```

Sabemos que este resultado es verdadero, sin embargo R no está de acuerdo.

En el ejemplo anterior, utilizamos la función `sqrt()`, que calcula la raíz cuadrada de su argumento, en este caso de 2. El operador `^` eleva al operando a la potencia determinada por el número a su derecha, también 2. Y el operador == compara ambos resultados.

3.7 Asignación de variables

En R no debemos preocuparnos por el tipo de dato que almacenamos en una variable, ya no es necesario (Gardener, 2012).

Para crear una variable y almacenar un valor en ella, basta con utilizar los símbolos <− o =, aunque el primero es el más usual, de la siguiente manera,

```
> x <- 15  # se almacena 15 en la variable x
> x        # al escribir el nombre de la variable y teclear enter,
podemos observar el valor asignado.
[1] 15     # valor almacenado en la variable x.
```

En este ejemplo vale la pena mencionar que al asignar un elemento a una variable, R confirma haber realizado la tarea no mostrando mensajes y tampoco mostrando el valor actual de la variable pero dado el caso, mostrará si se incurrió en algún error. Para mostrar el valor almacenado, basta con escribir el nombre de la variable para poder visualizar el valor. Esto se debe a que R llama a la función `print()` automáticamente, a excepción de cuando trabajamos dentro de loops o funciones que el usuario declara, en esos casos, se tendrá que hacer uso explícito de la función `print()` o encerrar entre paréntesis la expresión que deba imprimirse en pantalla.

Se siguen algunas reglas para poder declarar los nombres de variables. Se pueden utilizar letras, números, puntos, guión bajo, pero no deben comenzar con un número o un punto seguido de un número. Tampoco se pueden llamar igual que las palabras reservadas de R como `for` e `if`. Llamando a la función `?make.names` obtenemos información de lo permitido en relación con los nombres de variables.

Para declarar variables globales utilizamos los símbolos <<− y se siguen las mismas reglas citadas.

Trabajar con variables puede complicarse al momento de recordarlas. Para poder visualizar las variables creadas, utilizamos la función `ls()`.

Y si se busca una variable o variables en particular que contengan letras en común, se puede utilizar `ls(pattern = "patrón_de_letras")`.

La función `rm(variable)` permite borrar la variable que se pase como parámetro. Para borrar todas las variables creadas de una sola vez, se utiliza la misma función con los siguientes argumentos: `rm(list=ls())`. Se debe tener cuidado cuando se utiliza de esta forma la función `rm()` ya que accidentalmente podríamos borrar todos los elementos creados. Es recomendable borrar una variable a la vez o enlistarlas manualmente dentro de `rm()` separadas por comas. Las variables, en este caso, no se escriben entre paréntesis.

```
> ls()  # muestra las variables almacenadas
[1] "dato1"   "orden"   "tipo"   "datos"    "muestra" # variables
almacenadas
>ls(pattern = "da")      # muestra las variables almacenadas que
contengan el patrón "da"
[1] "dato1"  "datos"
> rm(dato1, datos)          #  borra las variables dato1 y datos
```

El símbolo de la almohadilla o gato `#`, antecede a los comentarios que un usuario desee agregar para hacer más legible el código o como parte de la documentación de los programas. Recordemos que los comentarios en programación persiguen varios fines, entre ellos hacer que el código se vuelva comprensible para otros o para recordarle al mismo programador las intenciones que tuvo al codificar dicho programa.

3.8 Funciones

La definición de funciones en R es muy parecida a otros lenguajes de programación. Por ejemplo, si se desea crear una función que calcule el área de un círculo, basta con crear un nombre para dicha función y asignarle a dicho nombre la palabra reservada `function` con los parámetros que dicha función aceptará:

```
area_circulo <- function(x)
{
print(pi*x^2)
}
```

Como se observa, el nombre la función es `area_circulo`, y acepta un valor como parámetro, en este caso el radio del círculo del que se desea calcular el área.

En el caso particular en el que la definición de la función conste de solo una línea, como en el ejemplo, las llaves `{ }` no son necesarias pero es una buena práctica ponerlas. Lo que se encuentra dentro de las llaves se conoce como el cuerpo de la función.

Recordemos que al codificar en R debemos hacer uso de la función `print()` o encerrar entre paréntesis la sentencia si deseamos que el resultado de dicha función se imprima en pantalla.

Para poder llamar a esta función, basta con escribir su nombre y cambiar `x` por el valor del radio:

```
area_circulo(5)
[1] 78.53982
```

3.9 Estructuras de control

3.9.1 Las sentencias `if` y `else`

Las estructuras de control hacen el papel de reguladores del flujo de la ejecución de un programa o una función. Las estructuras de control se dividen en estructuras de secuencia, selección y repetición (Joyanes Aguilar & Zahonero, 2010).

La estructura `if`, es una estructura de selección y su funcionamiento se basa en validar una condición. Si dicha condición resulta ser cierta, la sentencia que la procede se ejecutará, de manera contraria, el programa continuará con la siguiente sentencia.

Su sintaxis es la siguiente,

```
if (condición) sentencia
```

La estructura `if – else` permite ejecutar una segunda sentencia de resultar ser falsa la condición.

```
if (condición) sentencia1 else sentencia2
```

Si la condición es verdadera, se ejecuta la sentencia1, de lo contrario, se ejecuta la sentencia2.

Veamos el siguiente ejemplo.

```
salario_impuesto <- function(salario) # creación de la función.
{
if(salario<5000)      # se    realiza  la  operación  lógica  para
determinar si salario es menor a 5,000.
{
  (salario_neto = salario)    # si salario es menor, salario_neto
es igual a salario.
}else
{
  (salario_neto = salario - (salario * .16)) # si salario es mayor
a 5,000, se le resta un impuesto.
}
}
```

Es importante notar que la palabra `else` se encuentra a continuación de la llave que cierra al `if`. Esta es la forma correcta de utilizar la estructura if-else, de lo contrario el programa arrojará un error.

En la estructura de control `if`, es común que se quiera realizar una operación lógica dentro de la condición comparando un valor escalar con un vector, y a pesar de la naturaleza vectorial de R,

esta operación no se realizará con éxito. Lo que probablemente sucederá es que la operación lógica se llevará a cabo entre el valor escalar y el primer elemento del vector en cuestión, no así con todos los elementos del vector.

Para estos casos existe la estructura de control especial `ifelse`. Esta estructura toma tres vectores como parámetros, el primero es lógico de condiciones, el segundo contiene valores que se regresan en caso de que el primero sea verdadero. Y el tercer parámetro, contiene valores que se regresan, en caso de que el primer vector sea falso.

El tamaño del segundo y tercer vector deben ser el mismo que el del primero, de lo contrario se pueden obtener resultados no esperados.

3.9.2 La sentencia `switch`

La estructura de control `switch` es de gran utilidad cuando se trata de seleccionar una de entre varias alternativas. Su uso puede ser reemplazado por sentencias anidadas `if – else` pero anidar dichas sentencias puede llevar a la confusión fácilmente.

Su sintaxis es

```
switch (sentencia, lista_opciones)
```

Sentencia se evalúa, obteniéndose un resultado. Si el resultado está entre 1 y el tamaño de `lista_opciones`, entonces el valor correspondiente en `lista_opciones` se evalúa, arrojando el resultado que corresponda a dicha opción. Si el valor de sentencia se encuentra fuera del rango de `lista_opciones`, el resultado de dicha operación será `NULL`.

Se puede nombrar a los elementos de `lista_opciones`, igualándolos a cierto valor u operación; así, sentencia debe ser exactamente igual a alguno de los elementos en `lista_opciones`. En este caso es posible definir un resultado en caso de que exista un valor fuera del rango de `lista_opciones`. Basta con enlistar dicho valor pero sin asociarlo a un nombre.

Los menús de opciones son un uso común de `switch`.

El siguiente ejemplo ilustra su uso.

```
opciones_switch <- function (x)  # declaración de la función
{
switch(
   x,        # parámetro de la función, se compara con la lista de
opciones.
alpha = 1,   #si   x es igual a    "alpha", el resultado es 1. Así
mismo con el resto de opciones
beta  = sqrt(4),
gamma =
{
 a <- sin(pi / 3)
4 * a ^ 2
},
   "Fuera de rango"     # valor por default.
)
}
```

3.10 Ciclos o loops

Los ciclos o loops, son estructuras de control de repetición. En R existen tres tipos de ciclos, `repeat`, `while` and `for`. En R el uso de estructuras de repetición es menos frecuente que en la mayoría de los lenguajes de programación debido a que su tipo de dato básico es un vector.

3.10.1 Ciclo `repeat`

`repeat` es un ciclo de gran facilidad de uso. En general, ejecutará el mismo código una y otra vez hasta que se le ordene lo contrario, presionando la tecla escape, saliendo de R o incluyendo una sentencia `break`. De forma contraria, las sentencias dentro de `repeat` se ejecutarán infinitamente.

Su sintaxis es la siguiente.

```
repeat sentencia
```

donde `sentencia` es regularmente un bloque de código.

El siguiente ejemplo clarifica el uso de `repeat` utilizando la sentencia `break` para detener el ciclo.

```
repeat
  {
message("Mensaje, me repito hasta el break")
vector_a<- sample(
      c(
        "primera linea",
        "segunda linea",
        "tercera linea",
```

```
        "ultima linea"
        ),
      1
    )
message("vector_a = ", vector_a)
if(vector_a == "ultima linea") break
  }
```

La función `sample` elige al azar elementos dentro del vector que se pasa como parámetro, del tamaño especificado (en este caso el tamaño es 1).

3.10.2 Ciclo `while`

A diferencia de `repeat`, el ciclo `while` primero evalúa la condición y de ser verdadera ejecuta el código dentro de su cuerpo. Es por ello que este ciclo puede llegar a no ejecutarse.

El ejemplo utilizado para ejemplificar el ciclo `repeat`, se puede emplear para demostrar la diferencia que guarda con el ciclo `while`.

```
vector_a<- sample(
  c(
    "primera linea",
    "segunda linea",
    "tercera linea",
    "ultima linea"
  ),
  1
)
while(vector_a != "ultima linea")
{
message("Mensaje: me repito hasta encontrar break")
vector_a<- sample(
    c(
      "primera linea",
      "segunda linea",
      "tercera linea",
      "ultima linea"
),
    1
  )
message("vector_a = ", vector_a)
}
```

Si el elemento dentro de `vector_a`, fuera igual a "`ultima linea`", el ciclo no se ejecutaría. El resto del ejemplo simula la misma situación que el utilizado en el ciclo `repeat`.

Usualmente es posible representar un ciclo `repeat` utilizando un ciclo `while` y viceversa. Lo más práctico es utilizar el que resulte más claro dependiendo de la situación.

3.10.3 Ciclo `for`

Los ciclos `for` son utilizados cuando de antemano se conoce el número de iteraciones que se realizarán.

Su sintaxis es la siguiente

```
for (nombre in vector) sentencia
```

En donde `vector` puede ser una lista o un vector, `nombre` es una variable que toma uno a uno los valores dentro de `vector` y se evalúa sentencia. Un efecto secundario es que la variable `nombre` sigue existiendo aún después de concluir el ciclo, conteniendo el último valor dentro de vector.

R es muy flexible en los ciclos `for` en el sentido de que no se limita sólo a números enteros, también es posible utilizar vectores de caracteres, vectores lógicos y listas como se ha mencionado anteriormente.

Ejemplo del uso del ciclo `for`.

```
for (letra in LETTERS)
message("La letra ", letra)
```

El código anterior imprime en pantalla las letras en mayúscula del abecedario romano, que se encuentran almacenadas dentro de la constante `LETTERS`. En R existen constantes de uso común, tales como pi, las letras del abecedario romano, en mayúsculas y minúsculas y los nombres de los meses en inglés. Las constantes son `pi`, `LETTERS`, `letters` y `month.name` respectivamente.

Es posible dejarle saber a R hasta que letra imprimir dentro del ciclo `for`, señalando el rango de elementos deseados.

```
for (letra in LETTERS[1:10])
message("La letra ", letra)
```

Recuérdese que R es capaz de realizar operaciones entre vectores sin necesidad de utilizar ciclos `for`, y que hacerlo utilizando ciclos resulta más lento. Es recomendable evitar los ciclos en lo posible. En el ejemplo anterior, bastaba con lo siguiente:

```
sprintf("%s %s", "La letra ", LETTERS[1:10])
```

y el ciclo es implícito.

`sprintf()`,es una función que permite visualizar los resultados utilizando el formato del lenguaje de programación C.

3.11 Paquetes de funciones

Un paquete es una colección de funciones de R, esto es, funciones disponibles para su uso que generalmente están agrupadas en tareas afines.

Uno de los beneficios más grandes que se obtienen al utilizar R es el de poder aprovechar un esfuerzo común. Existen miles de paquetes que extienden la funcionalidad de R al proveer de funciones que otros usuarios, incluyendo las que el equipo de programadores "R CoreTeam", definan.

La mayoría de los paquetes actuales se encuentran instalados en un repositorio en línea llamado CRAN (por sus siglas en inglés, the Comprehensive R Archive Network).

Para cargar un paquete que ya se encuentra instalado en el equipo, basta con llamar a la función `library()`, pasando como argumento el nombre del paquete que se desee cargar, sin comillas.

Si se intenta cargar un paquete que no se encuentra instalado, R arrojará un error advirtiendo el hecho.

La función `search()` nos permite cerciorarnos de qué paquetes se encuentran cargados.

Si no se han cargado paquetes extra desde la instalación de R, es muy probable que sean los mismos a los que se enlistan a continuación.

```
> search()
 [1] ".GlobalEnv"        "tools:rstudio"     "package:stats"
"package:graphics"
 [5] "package:grDevices" "package:utils"     "package:datasets"
"package:methods"
[9] "Autoloads"         "package:base"
```

El orden en el que aparecen listados los paquetes, es el mismo que R utiliza cuando busca una variable o función. En este caso, primero busca en "`.GlocalEnv`" y por último en "`package:base`".

La función `installed.packages()` permite obtener información acerca de los paquetes que R conoce.

3.11.1 Instalación y carga de paquetes

Las versiones recientes de R están programadas para acceder al repositorio de paquetes CRAN y al repositorio CRANextra si se utiliza el sistema operativo Windows.

Es posible acceder a repositorios adicionales utilizando la función `setRepositories()`. Si se utiliza un IDE, es usual que muestre un mensaje en la línea de comandos parecido al siguiente:

```
> setRepositories( )
--- Please select repositories for use in this session ---


1: + CRAN
2: + CRAN (extras)
3:   BioC software
4:   BioC annotation
5:   BioC experiment
6:   BioC extra
7:   Omegahat
8:   R-Forge
9:   rforge.net

Enter one or more numbers separated by spaces, or an empty line to cancel
1:
```

Esperando a que el usuario teclee una de las posibles opciones.

Si se utiliza la interfaz gráfica que viene por defecto con la instalación de R, aparece una ventana como la siguiente, en donde es posible seleccionar el o los repositorios deseados.

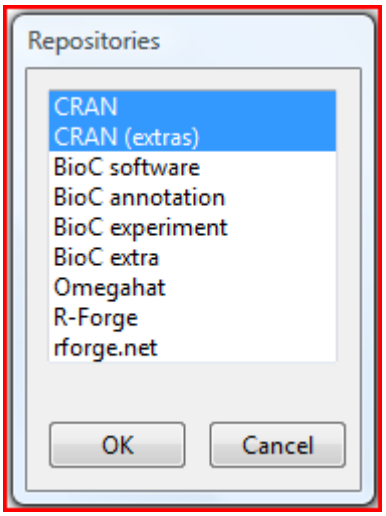


Figura 13: Repositorios de R.

Los repositorios BioC (por Bioconduntor), son repositorios con temas afines a la genómica y a la biología molecular, R-Forge y rforge, contienen paquetes en desarrollo que en algún momento estarán disponibles en CRAN.

Si se desea información acerca de los paquetes que están disponibles dentro del o los repositorios que han sido seleccionados, se llama a la función `available.packages()`, pero se debe tener en cuenta que llegan a ser miles de paquetes, lo que puede tomar considerable tiempo de ejecución.

Para instalar un paquete utilizando la interfaz gráfica de R, basta con seleccionar la pestaña **Packages**, decidiendo entre "Installpackage(s)..." o "Inastallpackage(s) from local zip files...". la diferencia entre uno y otro es que al seleccionar la primera opción la interface muestra una ventana preguntando por el servidor de descarga enlistados por país. Al seleccionar el servidor muestra una lista de los paquetes disponibles.

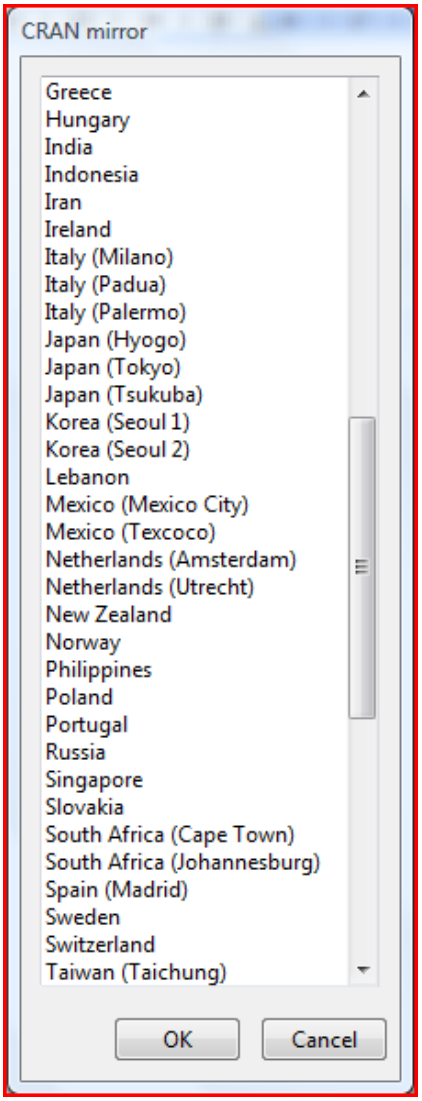


Figura 14: Servidores de descarga enlistados por país.

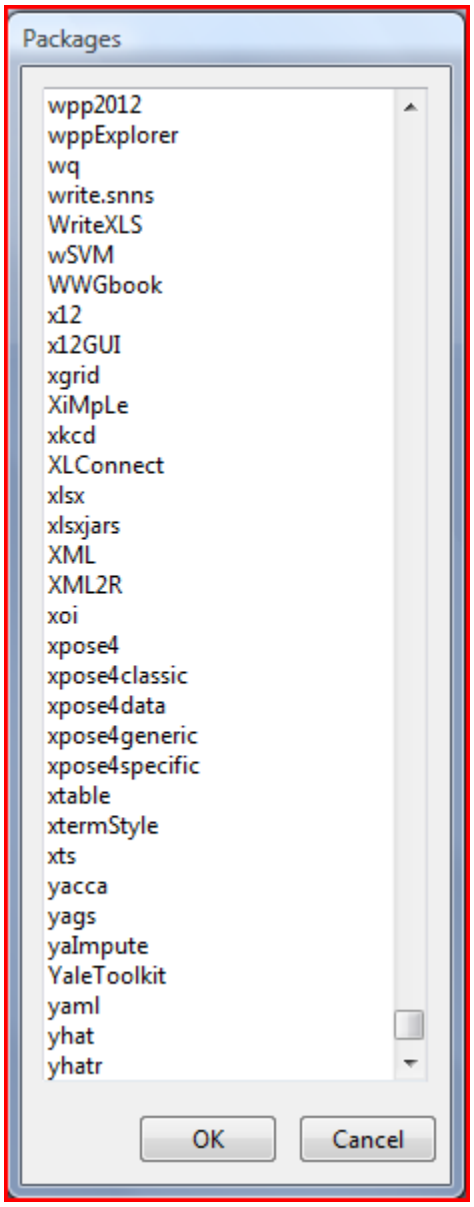


Figura 15. Paquetes disponibles.

Y la segunda opción, "Inastallpackage(s) from local zip files...", permite instalar un paquete ya descargado y buscarlo dentro los archivos en el equipo.

La opción en línea de comandos es igual de simple pero es necesario conocer el nombre del paquete y la dirección web del servidor desde donde se desea la descarga, utilizando la función `install.packages()`

```
install.packages("xts", repos = "http://cran.itam.mx/")
```

"`xts`" es un paquete de análisis de series de tiempo y la dirección web corresponde al servidor del Instituto Tecnológico Autónomo de México (ITAM).

En el sitio web http://cran.r-project.org/mirrors.html se encuentra la lista de servidores disponibles.

Al finalizar la instalación del paquete se recibe un mensaje de instalación satisfactoria y la dirección dentro de los archivos del equipo en donde se almacenó dicho paquete. Si la palabra "successfully" no aparece, es probable que haya ocurrido un error.

El mensaje es como el siguiente,

```
package 'xts' successfully unpacked and MD5 sums checked
```

Las funciones `library()` o `require()` permiten cargar los paquetes instalados. Así, después de pasar como argumento el paquete "`xts`", y revisar los paquetes cargados con `search()`, se aprecia el nombre del paquete "`xts`" cargado y listo para su uso.

```
> library(xts)
> search()
 [1] ".GlobalEnv"        "package:xts"       "package:zoo"
"tools:rstudio"
 [5] "package:stats"     "package:graphics"  "package:grDevices"
"package:utils"
 [9] "package:datasets"  "package:methods"   "Autoloads"
"package:base"
```

Es posible actualizar los paquetes instalados para obtener las últimas versiones, utilizando la función `update.packages()`. Después de llamar a la función, R pregunta `update (y/n/c)?`, donde "y" significa sí (yes), "n", no (no) y "c" cancelar (cancel).

También es posible eliminar algún paquete. Esto se puede llevar a cabo eliminando directamente la carpeta en donde el paquete se había almacenado o utilizando la función `remove.packages()` pasando como argumento el nombre del paquete entrecomillado.

4. Métodos de series de tiempo con R

4.1 Cómo leer datos

Una forma sencilla de leer los datos en R, desde archivos .txt y .csv, es haciendo uso de la función `scan()`. En su forma básica, `scan()` permite leer datos pasando como referencia la ubicación del archivo, ya sea en la computadora del usuario o en alguna parte de la internet.

Las opciones completas que proporciona la función `scan()`, son las siguientes;

```
Scan (file = "", what = double(), nmax = -1L, n = -1L, sep = "",
    quote = if (identical(sep, "\n")) "" else "'\"", dec = ".",
    skip = 0L, nlines = 0L, na.strings = "NA", flush = FALSE,
    fill = FALSE, strip.white = FALSE, quiet = FALSE,
blank.lines.skip = TRUE,
    multi.line = TRUE, comment.char = "", allowEscapes = FALSE,
    fileEncoding = "", encoding = "unknown", text)
```

De entre las opciones anteriores destacan `what`, que permite especificar el tipo de dato a ser leído; puede ser entero, lógico, numérico, complejo, lista, carácter y raw. `sep`, que especifica el caracter con el que se delimitará cada dato en el archivo a ser leído y `skip`, que permite indicar el número de líneas que se omitirán y a partir de las cuales se leerán los datos. La descripción completa de las opciones, se puede consultar en, https://stat.ethz.ch/R-manual/R-devel/library/base/html/scan.html.

Otra opción es utilizar la función `file.choose()` junto con la función `read.csv()`. Dichas funciones permiten cargar un archivo .cvs a R con la ventaja de poder visualizar el archivo.

Se puede utilizar de la siguiente manera,

```
DatosMetro<-read.csv(file.choose())
```

Al ejecutar la instrucción anterior, aparece un cuadro de dialogo en donde es preciso seleccionar el archivo:

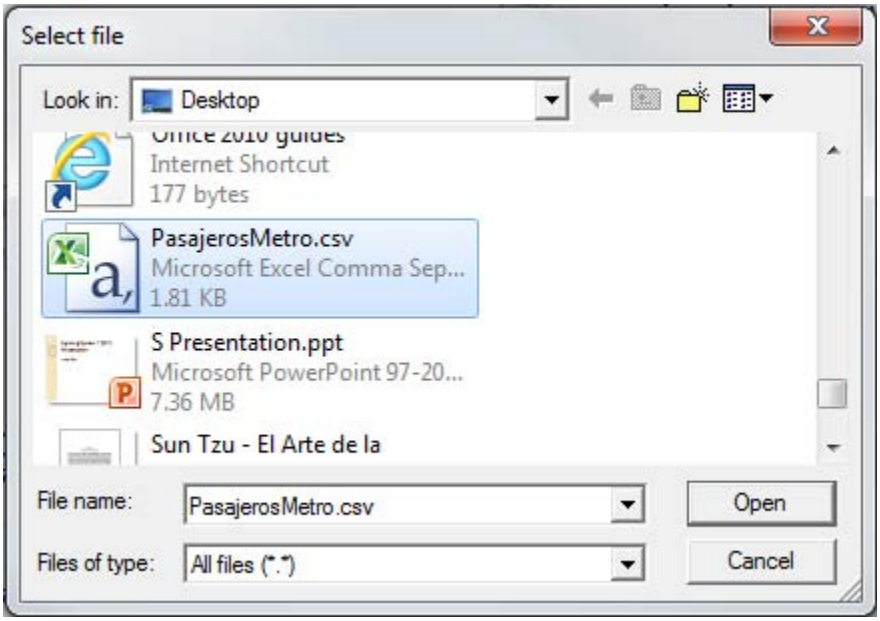


Figura 16: Uso de las funciones read.csv() y file.choose()

El siguiente paso después de leer los datos, es convertirlos a un objeto del tipo "serie de tiempo". Esto se logra utilizando la función **`ts( )`**. Dicha función cuenta con las siguientes opciones,

```
ts (data = NA, start = 1, end = numeric(), frequency = 1,
    deltat = 1, ts.eps = getOption("ts.eps"), class = if (nseries
>
        1) c("mts", "ts", "matrix") else "ts", names = if
(!is.null(dimnames(data))) colnames(data) else paste("Series",
        seq(nseries)))
```

Las opciones que destacan por su uso son `data`, que se refiere a los datos en concreto, pueden estar almacenados en una variable. `start`, la fecha de la primera observación. `end`, la fecha de la última observación y `frequency`, el número de observaciones por unidad de tiempo. La descripción completa de dichas opciones se pueden consultar en, http://stat.ethz.ch/R-manual/R-patched/library/stats/html/ts.html.

Visualicemos lo anterior utilizando la serie que se refiere al número de usuarios del Sistema de Transporte Colectivo Metro del Distrito Federal, a partir del año 1986 hasta el año 2013. La frecuencia de los datos es mensual.

La siguiente imagen muestra los datos almacenados en un archivo .txt.

```
Comunicaciones y transportes > Principales características del sistema de trans
Unidad de medida: Miles de pasajeros, Periodicidad: Mensual
1986/01 - 2013/12
Notas:
a/ Promedio diario de pasajeros.
Cifras preliminares:
p/ A partir de 2012/01
Fuente: Gobierno del Distrito Federal. Sistema de Transporte Colectivo Metro

3556.3
3513.3
3354.5
3758.4
3808.3
3824.5
4009.3
3874.4
3837.7
3875.1
3723
3513.8
3739.4
3832.2
3929.4
3663.3
3751.9
3886.4
3881
3910.5
3909.8
4061.3
4475.1
3854
3871.5
4050.9
3991.5
3882.4
4042.9
4138
3967
4144.1
4043.1
4157.9
4182
3929
4166.8
4270.5
```

Figura 17: Ejemplo de datos almacenados en un archivo .txt.

Para leer los datos anteriores, utilizamos el comando,

```
Pasajeros <-
ts(scan("C:/Users/usuario/Desktop/PasajerosMetro.csv", skip = 9),
frequency = 12, start = c(1986, 1))
```

Asignamos a la variable Pasajeros, el archivo ubicado en `C:/Users/usuario/Desktop/PasajerosMetro.csv`, `skip=9` (le dice a R que comience a leer los datos a partir de la décima línea). `frequency=12` se refiere al número de observaciones por año a considerar; como los datos son mensuales, el valor de `frequency` es `12`. `start` indica el año y periodo a partir del cual comienzan los datos, en este caso enero de 1986. Así, si los datos se leyeron con éxito, se visualiza un mensaje en la consola indicando el número de datos leídos. En nuestro caso, se cuenta con 336 datos, por lo que R muestra el siguiente mensaje,

```
Read 336 items
```

4.2 Graficar Series de Tiempo

Graficar una serie de tiempo es muy sencillo. Se utiliza la función `plot.ts()`, pasándose como parámetro un objeto de tipo serie de tiempo.

Para el ejemplo anterior, utilizamos la función `plot.ts()` de la siguiente manera,

```
plot.ts(Pasajeros, main="Usuarios del Metro", ylab="Número de usuarios (miles)", xlab="Año")
```

la opción `main` permite poner título a las gráficas, `ylab` y `xlab` permiten nombrar los ejes verical y horizantal respectivamente. Ver figura 18.

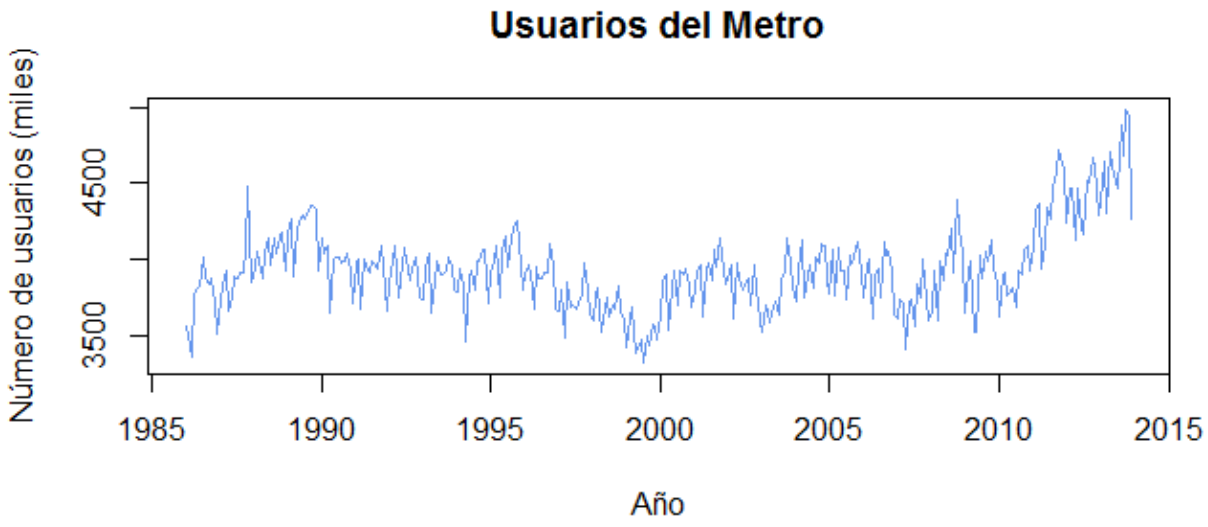


Figura 18: Usuarios del STC Metro.

Es posible observar que la media de la serie parece rondar el valor de 4,000 y que a partir del año 2010 el número de usuarios tiene una tendencia positiva, y que en los últimos meses de 2013 se alcanzan niveles aproximados de 5,000,000 de usuarios.

Trabajemos ahora la serie de tiempo relacionada con el porcentaje de desocupación de la población económicamente activa. La serie contiene los datos recabados mensualmente a partir de enero de 2009 a febrero de 2014 [1].

[1] INEGI. *Tasa de desocupación mensual.* Periodo de enero de 2009 a febrero de 2014. Fuente (INSTITUTO NACIONAL DE ESTADÍSTICA Y GEOGRAFÍA, 2014c)

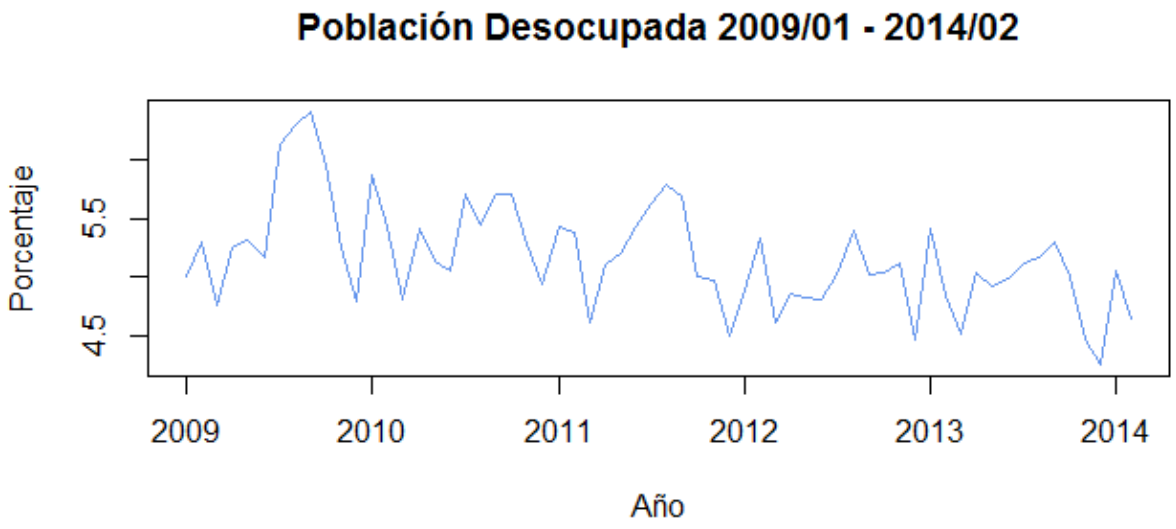


Figura 19: Tasa de desocupación mensual en México.

El gráfico muestra la existencia datos atípicos, la tendencia es un tanto negativa y disminuye lentamente.

La función de autocorrelación (ACF) indica la posible existencia de variación estacional cada 12 meses,

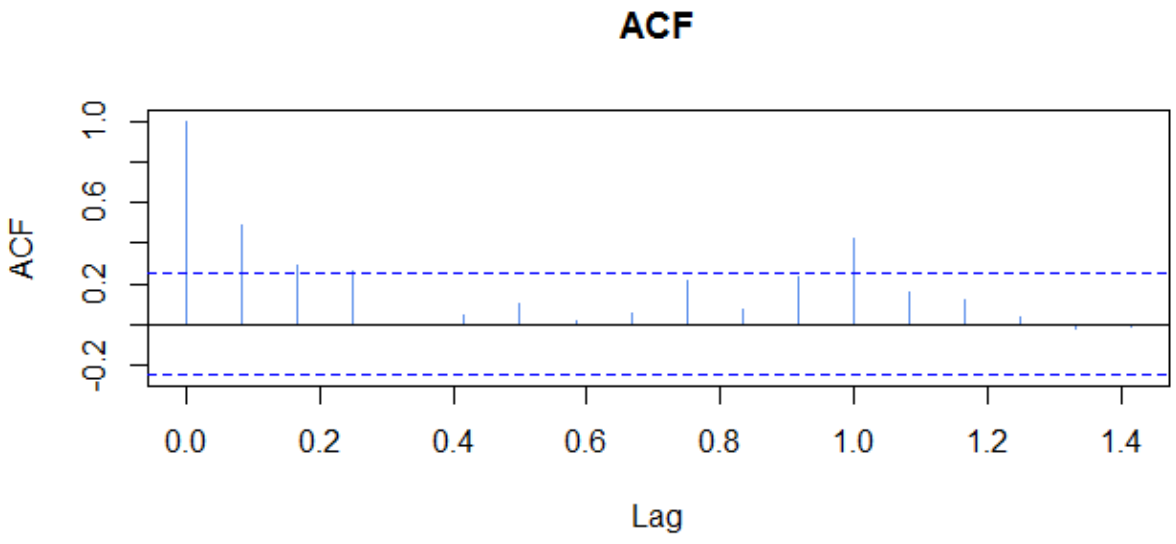


Figura 20: ACF de la tasa de desocupación mensual.

La función `acf( )` en R, permite graficar la ACF.

```
acf(PoblacionD, main = "ACF", lag.max = 18)
```

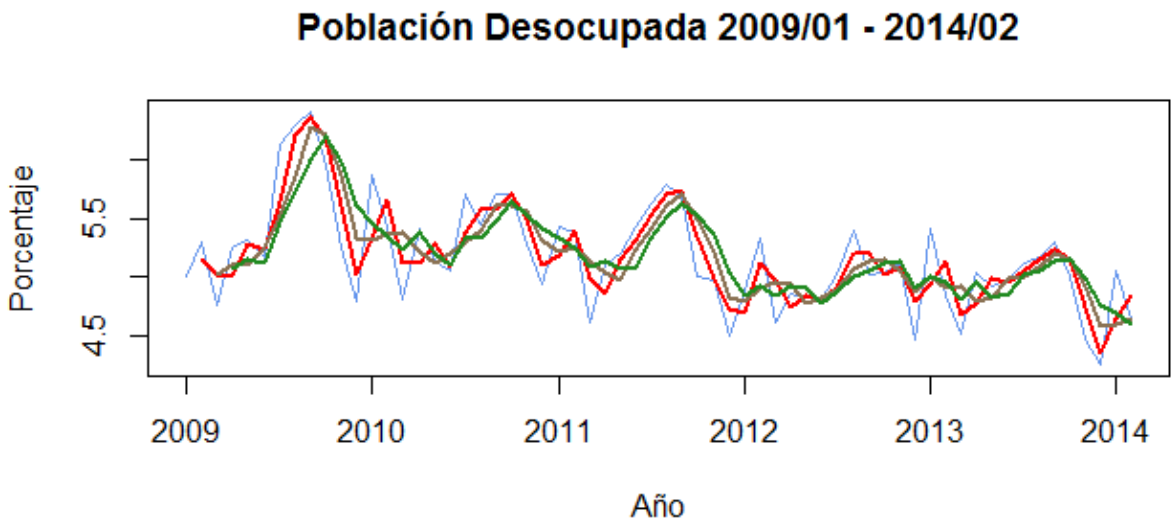
El primer argumento es la serie de tiempo, `main` da título al gráfico y `lag.max`, es el retardo máximo a calcular. La función se grafica por default, no es necesario hacer uso de otra función para poder visualizarla.

4.3 Aplicación promedios móviles simples

Para aplicar el método de promedios moviles simples al ejemplo anterior, haremos uso de la función `SMA()`, que es el acrónimo en ingles del nombre del método (Simple Moving Averages). Es necesario instalar y después cargar el paquete "TTR" (Technical Trading Rules).

La función recibe dos párametros, el primero es un objeto de tipo serie de tiempo y "n", que es el número de periodos a promediar. Recordemos primero que este método ofrece mejores resultados cuando la serie es estacionaria.

Aplicando la función SMA con n igual a 2, 3, 4 y graficando los resultados a la vez, tenemos




```
plot.ts(PoblacionD, main="Población Desocupada 2009/01 - 2014/02",
ylab="Porcentaje", xlab="Año", col="cornflowerblue")
lines(SMA(PoblacionD, 2), col = "red", lwd=2)
lines(SMA(PoblacionD, 3), col = "burlywood4", lwd=2)
lines(SMA(PoblacionD, 4), col = "forestgreen", lwd=2)
```

En primer lugar la función `lines()` nos permite agregar información a un gráfico. En el caso anterior pasamos como parámetros la función `SMA`, aplicada a la serie original. Un color distintivo

para los diferentes índices de promedios móviles y el parámetro `lwd`, que nos da la opción de cambiar el ancho de la línea.

También es util hacer uso de la función `tail()`. Esta nos permite acceder a los "n" últimos elementos de un vector, serie de tiempo o a los "n" últimos renglones de una matriz. Su contraparte es la función `head()`.

```
tail(PoblacionD, n=1)
tail(SMA(PoblacionD, 2), n=1)
tail(SMA(PoblacionD, 3), n=1)
tail(SMA(PoblacionD, 4), n=1)
```

cuyos resultados son 4.646367, último valor en la serie original, 4.849573, 4.650796 y 4.605834 son los valores pronosticados para el próximo periodo, respectivamente.

La gráfica comparativa del método, nos permite vizualizar el error cometido. En el caso del porcentaje de desocupación, el valor más adecuado para n parece ser n = 2, que es la línea roja, ya que parece ajustarse mejor al comportamiento de la gráfica original, color azul. En estos casos resulta útil utilizar el porcentaje de error medio absoluto (PEMA).

$$PEMA = \frac{\sum_{i=1}^{n}\left|\frac{e_t}{x_t}\right| * 100}{n}$$

Aquí hay dos opciones, encontrar una función que ya haya sido programada para calular el PEMA o programar una rutina que lo calcule y que además se adecue al ejemplo en particular. Una opción es la siguiente, que si bien calcula el PEMA cuando se trata del método de promedios móviles simples, es probable que se deban hacer cambios para utilizarla con otro método.

```
PEMASMA<-function (seriea, serieb, n)
{
 e<-seriea[(n+1):length(seriea)]-serieb[n:(length(seriea)-1)]
 suma<-sum(abs(e)/seriea[(n+1):length(seriea)])
 print(suma/length(e)*100)
}
```

Los parámetros que toma son 3: la serie original (llamada seriea), la serie suavizada (llamada serieb) y el orden de suavizamiento n. Cabe hacer notar que no fue necesario hacer uso de alguna estructura de repetición, dada la naturaleza de R.

Para poder utilizar la función anterior basta llamarla y pasarle los distintos parámetros.

```
PEMASMA(PoblacionD, SMA(PoblacionD, 2), 2)
PEMASMA(PoblacionD, SMA(PoblacionD, 3), 3)
PEMASMA(PoblacionD, SMA(PoblacionD, 4), 4)
```

Cuyos resultados son, 6.807118%, 6.407433% y 6.579305%, respectivamente, para n igual a 2,3 y 4.

Gráficamente el método parecía arrojar mejores resultados con n=2, pero después de calcular el PEMA, el menor porcentaje obtenido es cuando aplicamos el método con n=3.

Por lo tanto se puede concluir que los mejores pronósticos utilizando el método de los promedios móviles simples, para el caso de la tasa de población desocupada, son los obtenidos cuando aplicamos el método con n igual a 3 y que el próximo valor de la serie, para el periodo 2014/03, es 4.650796% con un error aproximado del 6.407433%.

4.4 Aplicación promedios móviles dobles

Para aplicar el método de promedios móviles dobles basta con utilizar la función SMA a los datos ya suavizados. Recuerdese que además de esto modelamos la tendencia con las ecuaciones,

$$a_t = 2M'_t - M''_t$$

$$b_t = \frac{2}{N-1}(M'_t - M''_t)$$

Y obtenemos el pronóstico de m periodos con la ecuación,

$$F_{t+m} = a_t + mb_t$$

Utilizando la siguiente función en R, obtenemos el siguiente valor de la serie en el ejemplo de la tasa de desocupación utilizando promedios móviles dobles con n=2,

```
F_t <- function (PMS, PMD, n)
{
  a_t <- function (PMS, PMD, n)
  {
    return(2*PMS[(n+1):length(PMS)]-PMD[(n+1):length(PMD)])
  }
  b_t <- function (PMS, PMD, n)
  {
    return((2/(n-1))*(PMS[(n+1):length(PMS)]-
PMD[(n+1):length(PMD)]))
  }
  return(a_t(PMS, PMD, n)+b_t(PMS, PMD, n))
}
```

La función `F_t()`, permite calcular los valores de la ecuación $F_{t+m} = a_t + mb_t$.

Cuyo valor es 5.144424%.

Gráficamente,

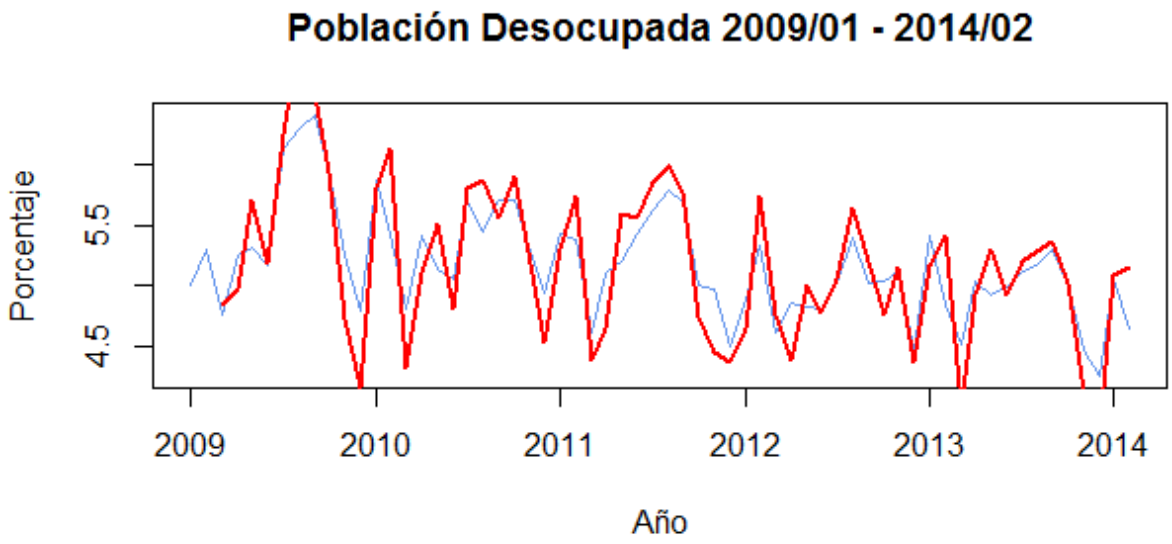


Figura 21: Serie original (azul) vs serie pronóstico (rojo), n=2.

Donde la líena azul es la serie original y la línea roja es la serie pronóstico.

Y con n=3, el pronóstico es de 4.382535%. Gráficamente:

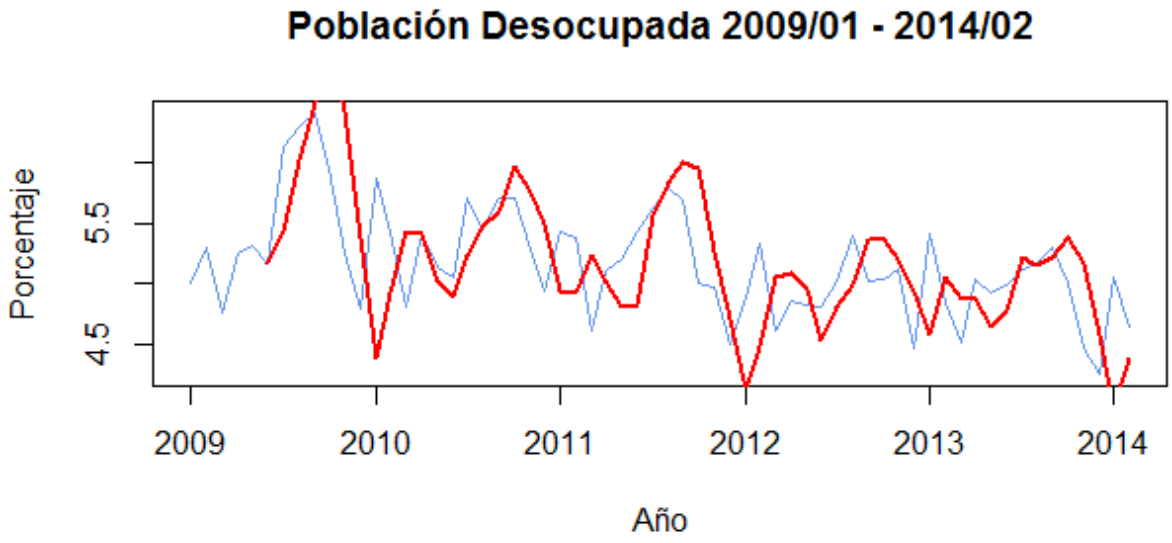


Figura 22: Serie original (azul) vs serie pronóstico (rojo), para n=3.

Se aprecia que la serie, entre mayor sea el grado de suavizamiento, va eliminando más las fluctuaciones. Así, si se aplicara el método con una n cada vez mayor, la serie se converiría en una línea constante.

Veamos qué sucede al aplicar este método a la serie del número de vehículos particulares de motor registrados en circulación en el país, de enero de 2008 a enero de 2014[2].

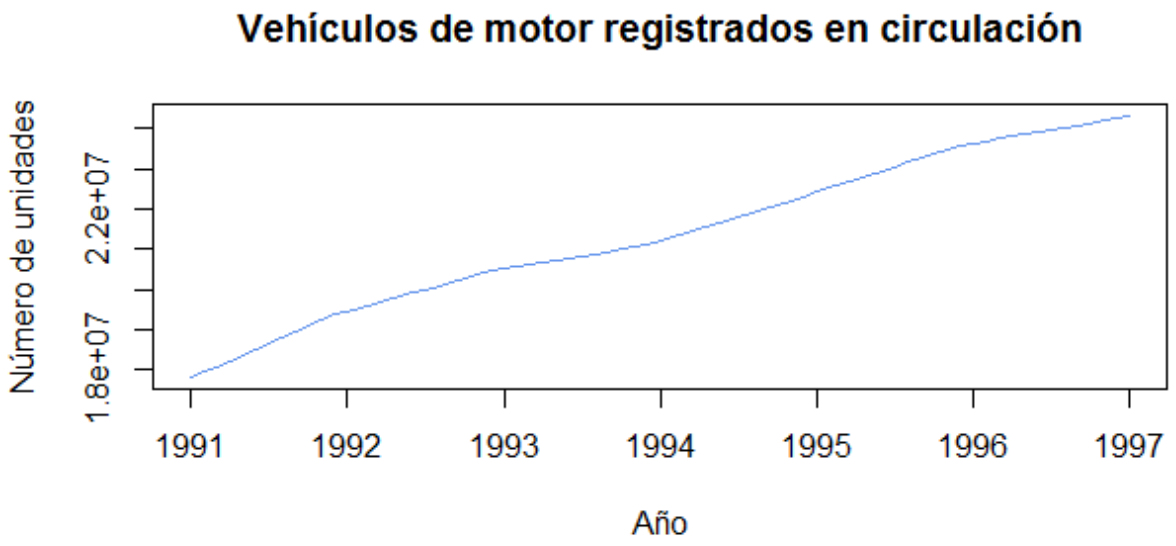


Figura 23: Vehículos de motor registrados en circulación.

Se aprecia que la serie tiene una tendencia positiva y una varianza constante. Al aplicar el método de promedios móviles dobles con n=2 y n=3 se obtiene el siguiente gráfico,

[2] INEGI. *Número de vehículos particulares de motor registrados en circulación en México de enero de 2008 a enero de 2014.* Fuente (INSTITUTO NACIONAL DE ESTADÍSTICA Y GEOGRAFÍA, 2014b)

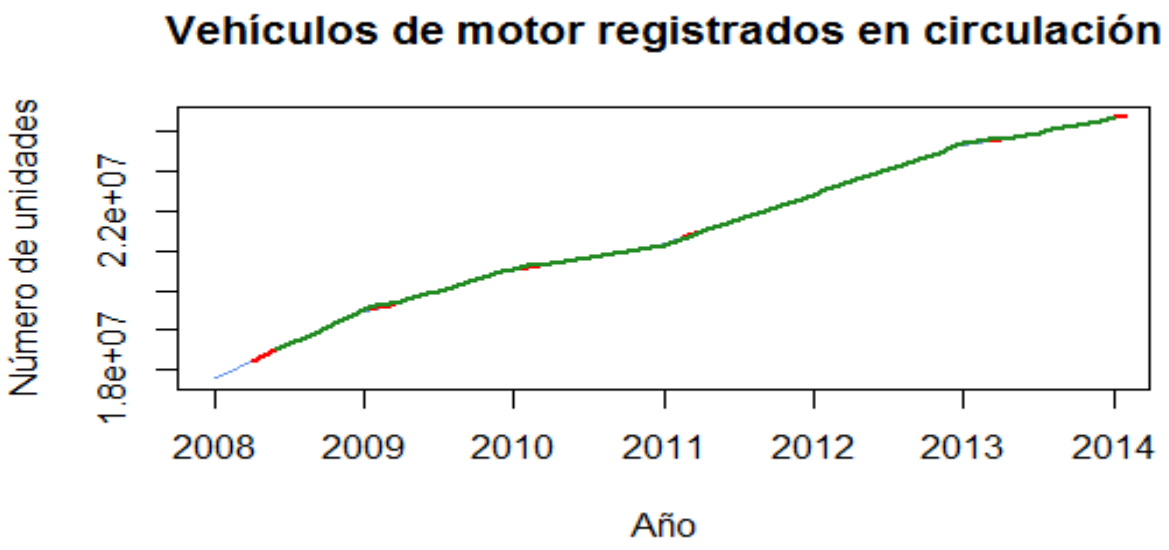


Figura 24: Comparativo entre serie original y series pronóstico.

Los resultados del método se superponen a la gráfica original de los datos, por lo que es necesario utilizar nuevamente el PEMA para determinar qué valor de n se ajusta mejor a los datos. Los valores pronósticados para n=2 y n=3 son 24,413,281 y 24,339,711 respectivamente.

La función de PEMA se modifica ligeramente para poder manejar los elementos de las series , ya que al promediar se pierden datos.

Calculando el PEMA para n=2 obtenemos 0.0284% y con n=3, 0.3908%

Basados en los resultados anteriores, se puede concluir, que el mejor pronóstico aplicando el método de promedios móviles dobles, se obtine al utilizar una n=2 con un error del 0.0284%, para el caso del número de vehículos de motor registrados en circulación.

4.5 Aplicación suavizamiento exponencial simple

El siguiente gráfico está basado en el número de matrimonios registrados entre los años 1990 - 2012[3]. Los datos fueron recabados anualmente. Se debe señalar que los datos a partir del 2010, incluyen el número de matrimonios entre personas del mismo sexo.

[3] INEGI. *Número de matrimonios registrados anualmente en México del año 1990 al 2012*. Fuente (INSTITUTO NACIONAL DE ESTADÍSTICA Y GEOGRAFÍA, 2014b)

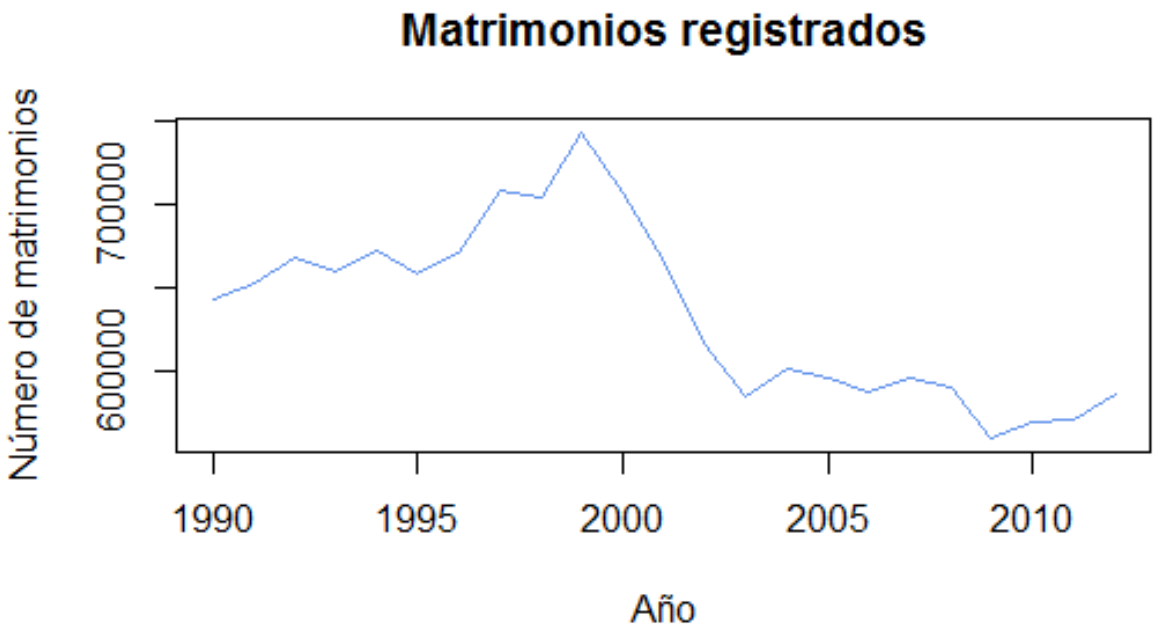


Figura 25: Matrimonios registrados entre 1990 - 2012.

Se aprecia una notable decaída en el número de matrimonios alrededor del año 1999 hasta aproximadamente el año 2004.

Para poder aplicar el método de suavizamiento exponencial simple, podemos utilizar la función `HoltWinters()`, que se encuentra en el paquete stats, instalado por defecto en R, cuyos parámetros son,

```
HoltWinters(x, alpha = NULL, beta = NULL, gamma = NULL,
            seasonal = c("additive", "multiplicative"),
            start.periods = 2, l.start = NULL, b.start = NULL,
            s.start = NULL,
            optim.start = c(alpha = 0.3, beta = 0.1, gamma = 0.1),
            optim.control = list())
```

De entre los parámetros están `x`, que debe ser un objeto de tipo serie de tiempo, `alpha`, `beta` y `gamma` que son los parámetros de suavizamiento, nivel y estacionalidad respectivamente. Con el parámetro `seasonal` podemos elegir el tipo de modelo a utilizar, ya sea aditivo o multiplicativo. `l.start` y `b.start` que son los valores de inicio para el nivel de la serie y la estacionalidad respectivamente.

Es posible aplicar el método de suavizamiento exponencial simple utilizando la función mencionada al no tomar en cuenta los valores para `beta` y `gamma;` en términos de R, fijar dichos parámetros a su estado "`FALSE`".

Utilizando la función y asignando el resultado a la variable `MatrimoniosPro`, tenemos,

```
MatrimoniosPro <- HoltWinters(Matrimonios, beta=FALSE,
gamma=FALSE)
```

El resultado que arroja R al llamar a la variable MatrimoniosPro es el siguiente:

```
Holt-Winters exponential smoothing without trend and without seasonal component.

Call:
HoltWinters(x = Matrimonios, beta = FALSE, gamma = FALSE)

Smoothing parameters:
 alpha: 0.9999434
 beta : FALSE
 gamma: FALSE

Coefficients:
      [,1]
a 585433.2
```

Figura 26: Resultado de la llamar a la variable MatrimoniosPro.

El resultado en consola, muestra que el valor estimado para alpha es de .9999, muy cercano a 1.

La función `HoltWinters()` pronóstica únicamente para los mismos periodos que la serie de tiempo original. Los pronósticos se almacenan en una variable llamada "`fitted`" y para poder visulizar los valores utilizamos el nombre de la variable a la que le asignamos el resultado de aplicar el método, seguido por `$fitted`,

```
> MatrimoniosPro$fitted
Time Series:
Start = 1991
End = 2012
Frequency = 1
         xhat    level
1991 642201.0 642201.0
1992 652171.4 652171.4
1993 667597.1 667597.1
1994 659567.5 659567.5
1995 671639.3 671639.3
1996 658114.8 658114.8
1997 670522.3 670522.3
1998 707837.9 707837.9
1999 704456.2 704456.2
2000 743853.8 743853.8
2001 707424.1 707424.1
2002 665436.4 665436.4
2003 616656.8 616656.8
2004 584143.8 584143.8
2005 600562.1 600562.1
2006 595713.3 595713.3
2007 586978.5 586978.5
2008 595208.5 595208.5
2009 589352.3 589352.3
2010 558914.7 558914.7
2011 568631.5 568631.5
2012 570953.9 570953.9
```

Figura 27: Valores almacenados en la variable MatrimoniosPro$fitted.

Como se observa, el último año pronosticado es el último año incluido en la serie original.

Abajo observamos el gráfico de la serie original, en negro, contra el del pronóstico, en rojo.

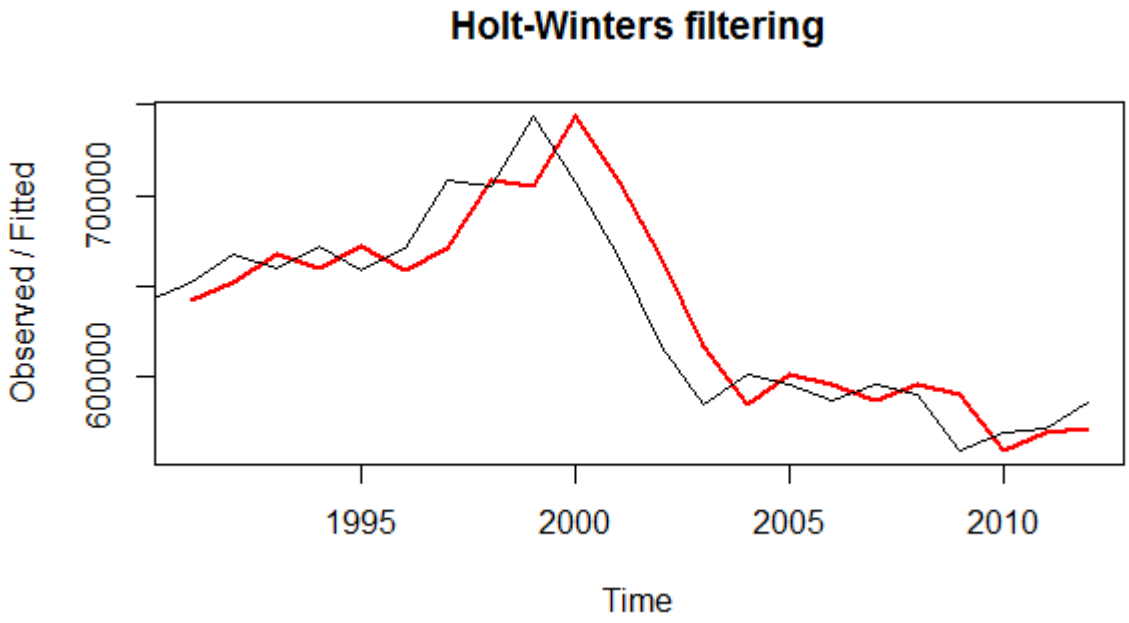


Figura 28: Serie original vs serie pronóstico.

Una medida para determinar el grado de presición del pronóstico, es la suma de los errores al cuadrado (SSE, por sus siglas en inglés), que se obtiene al sumar los cuadrados de los errores residuales. La función `HoltWinters()` también calcula la suma de los errores al cuadrado y lo almacena en la variable `SSE`, a la cual se puede acceder de la misma forma en que se hace para `fitted`

```
> MatrimoniosPro$SSE
[1] 12075783786
```

Como se mencionó anteriormente, esta función no ha pronosticado ningún periodo hacia el futuro. Para poder pronosticar "h" periodos hacia adelante, es necesario utilizar la función `forecast.HoltWinters()`, que se encuentra dentro del paquete `forecast`, en combinación con la función `HoltWinters()`, que, como ya se ha mencionado anteriormente, está dentro del paquete `stats` .

```
forecast(object,
h=ifelse(frequency(object$x)>1,2*frequency(object$x),10),
level=c(80,95),fan=FALSE,lambda=NULL,...)
```

donde `object` es un objeto de la clase `HoltWinters`, resultado de llamar a la función `HoltWinters`, `h` es el número de periodos a pronósticar y `level` son los intervalos de confianza para los pronósticos.

Así, llamando a la funcion `forecast.HoltWinters()` sobre la variable `MatrimoniosPro`, que es donde previamente se almacenaron los valores de la llamada a la función `HoltWinters()`, y asignando el resultado a la variable `MatrimoniosHW`, se obtienen los siguientes resultados,

```
> MatrimoniosHW<-forecast.HoltWinters(MatrimoniosPro, h=2, level=c(80,95))
> MatrimoniosHW
     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2013       585433.2 554888.6 615977.7 538719.3 632147.0
2014       585433.2 542237.9 628628.5 519371.7 651494.7
```

Dado que se pasó como valor del parámetro "h" el valor de 2, el método pronóstica dos valores hacia el futuro.

Bajo `Forecast` están los valores pronosticados para los años 2013 y 2014. Los resultados también se incluyen con intervalos de confianza del 80 y 95%.

Para graficar los pronósticos e intervalos, se utiliza la función `plot.forecast()`.

```
plot.forecast(MatrimoniosHW)
```

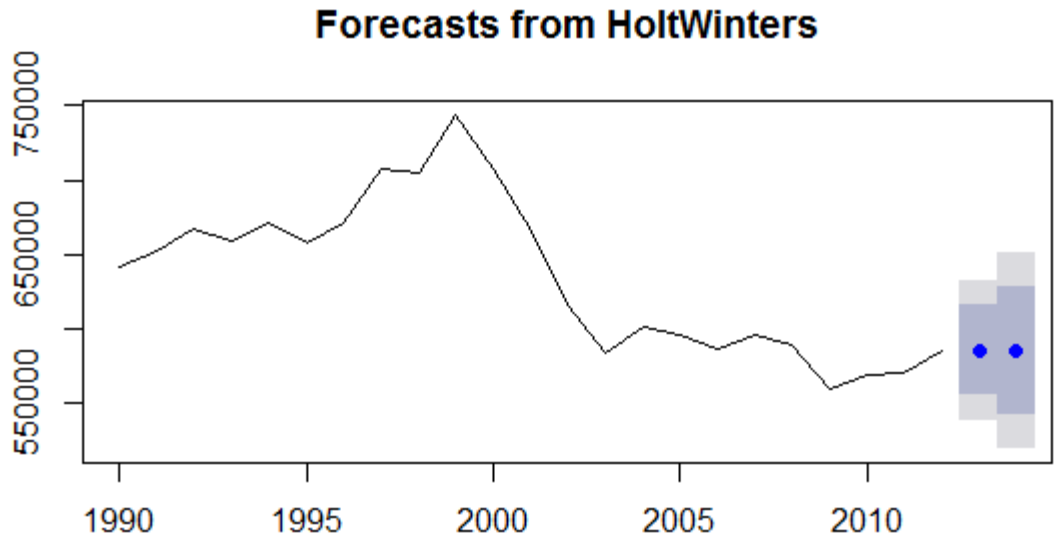


Figura 29: Pronóstico del número de matrimonios

Dentro del sombreado claro se encuentran los valores del pronóstico con un intervalo de confianza del 95% y en el sombreado oscuro, los valores pronosticados con un intervalo de confianza del 80%. Los puntos azules son los valores pronosticados.

Al aplicar la función `forecast.HoltWinters()`, los residuales se almacenan en el elemento llamado `residuals`, al cual podemos acceder de la misma manera en que accedemos a `fitted` y a `SSE`.

```
MatrimoniosHW$residuals
plot.ts(MatrimoniosHW$residuals, main="Residuales")
```

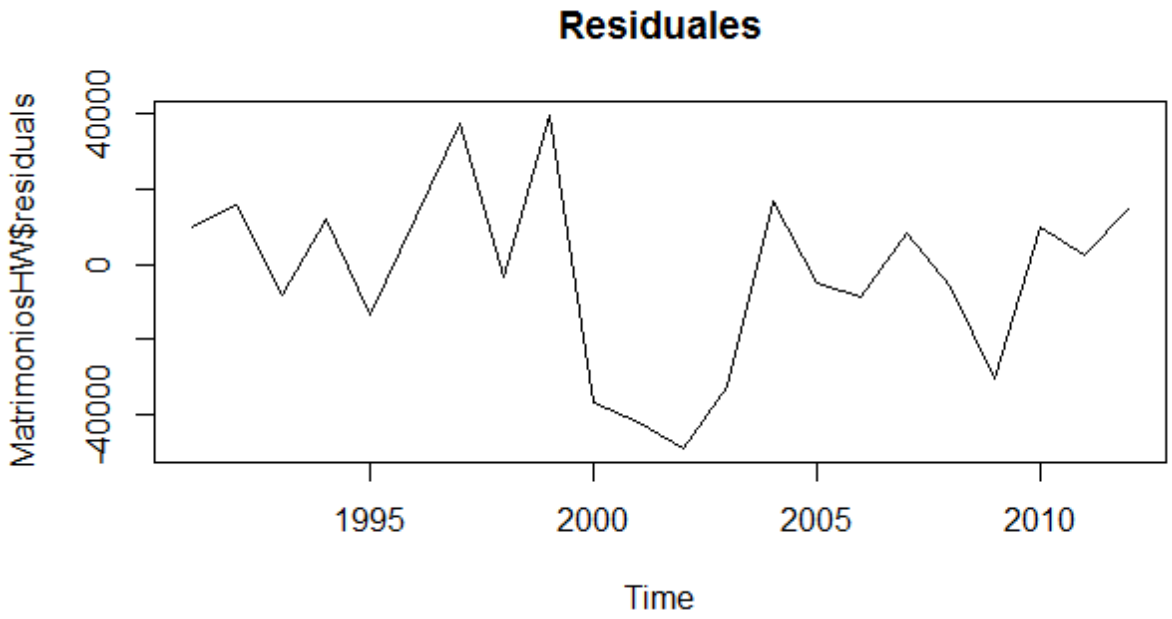


Figura30: Residuales.

Se observa que los residuales se comportan como ruido blaco, esto es, con media cero y varianza constante.

Con esto en mente podríamos aplicar alguna prueba de normalidad a los residuales, como el test de Kolmogorov-Smirnov o el test de Anderson-Darling.

Test de Kolmogorov-Smirnov.

Para aplicar esta prueba llamamos a la función `lillie.test()`, que se encuentra dentro del paquete `nortest`, pasando como único parámetro la variable donde hemos almacenado los residuales.

```
> lillie.test(MatrimoniosHW$residuals)

 Lilliefors (Kolmogorov-Smirnov) normality test

data:  MatrimoniosHW$residuals
D = 0.1295, p-value = 0.4386
```

Test de Anderson-Darling.

La función para aplicar esta prueba se llama `ad.test` y también se encuentra dentro del paquete `nortest`. Se pasa, como único parámetro , el conjunto de residuales.

```
> ad.test(MatrimoniosHW$residuals)

 Anderson-Darling normality test

data:  MatrimoniosHW$residuals
A = 0.4825, p-value = 0.2073
```

Periodograma integrado.

Con la función `cpgram()`, que se encuentra en el paquete `stats`, obtenemos el periodograma integrado.

```
cpgram(ts, taper = 0.1, main = paste("Series: ",
deparse(substitute(ts))), ci.col = "blue")
```

su primer parámetro es una serie de tiempo, en nuestro caso el valor de los residuales; `taper`, la proporción cónica en la formación del periodograma; `main`, se refiere al título que tendrá el gráfico, y `ci.col`, determina el color en los límites de la banda de confianza.

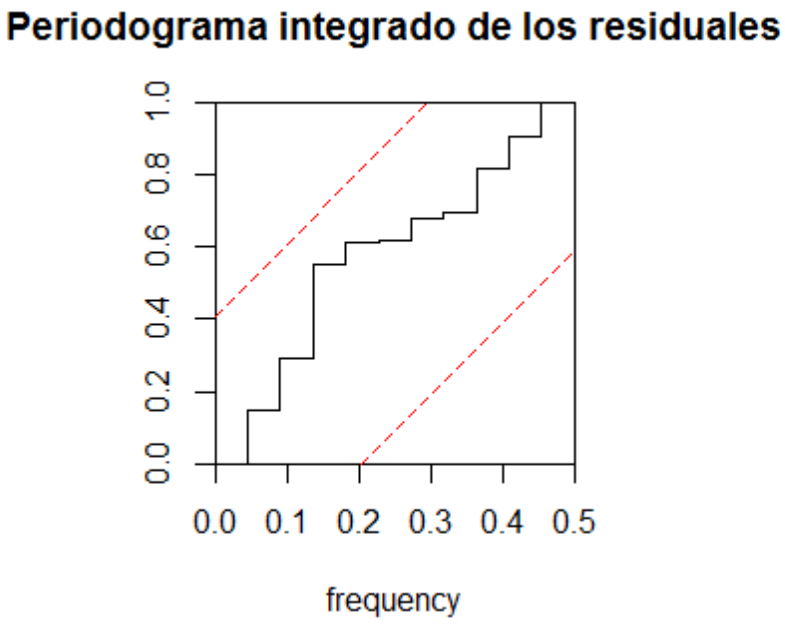


Figura 31: Periodograma integrado de los residuales

```
cpgram(MatrimoniosHW$residuals, main="Periodograma integrado de los residuales", ci.col="red")
```

Antes de concluir, mencionaremos a la función `ses()`, dentro del paquete `forecast`, que permite igualmente aplicar el método de suavizamiento exponencial simple a una serie de tiempo.

```
ses(x, h=10, level=c(80,95), fan=FALSE,
initial=c("optimal","simple"), alpha=NULL, ...)
```

en donde `x` es un vector o serie de tiempo, `h`, los periodos a pronosticar, `level`, los intervalos de confianza, `fan`, si dicho parámetro se pone al estado `TRUE`, el nivel de la serie se fija a `seq(50,99,by=1)`, `initial`, método usado para seleccionar los valores de inicio y `alpha` que es el parámetro de suavizamiento.

```
> MatrimoniosSES<-ses(Matrimonios, initial="simple", h=2)
> MatrimoniosSES
     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2013         585434 556069.4 614798.6 540524.7 630343.3
2014         585434 543906.2 626961.8 521922.7 648945.3
```

Que nos da valores muy parecidos al utilizar la función `forecast.HoltWinters()`.

Basado en los resultados anteriores, se puede concluir que aplicando el método de suavizamiento exponencial simple a la serie de tiempo "matrimonios registrados entre el año 1990 - 2012", el valor pronosticado para el siguiente periodo, año 2013, es de 585,434 matrimonios.

4.6 Aplicación Winters

Para aplicar el método de Winters aditivo y multiplicativo utilizamos la función `HoltWinters()`, que utilizamos en el ejemplo anterior, pero en este caso no fijamos los valores de `beta` y `gamma` a su estado `FALSE`.

Utilicemos la serie de tiempo relacionada con la venta mensual al público del total de automoviles en México[4], cuyo gráfico es el siguiente,

[4] INEGI. *Venta al público del total de automoviles en México del año 2009 a 2014.* Fuente (INSTITUTO NACIONAL DE ESTADÍSTICA Y GEOGRAFÍA, 2014d)

Figura 292: Venta mensual al público de automóviles en México.

Como se aprecia en el gráfico, la serie muestra tendencia positiva y lo que parece ser un patrón estacional. Aparentemente la venta de automóviles sube cada fin de año.

Se puede verificar lo anterior gráficamente, al utilizar la función `decompose`, que se encuentra dentro del paquete `stats`, y sus parámetros son los siguientes,

```
decompose(x, type = c("additive", "multiplicative"), filter =
NULL)
```

donde `x` es una serie de tiempo, `type`, es el tipo de componente estacional, aditivo o multiplicativo y `filter` es un vector de coeficientes usado para filtrar el componente estacional.

La función `decompose()`, separa los componentes de la serie original y los almacena en las variables `$seasonal`, `$trend` y `$ramdom` cuyos valores son los del componente estacional, la tendencia y fuctuaciones aleatorias, respectivamente.

La función por sí sola nos arroja los valores de los diferentes componentes y los podemos ver de forma gráfica si aplicamos la función `plot()` a una variable donde se almacenen los datos, o bien aplicándolo a la llamada de la función `decompose()`.

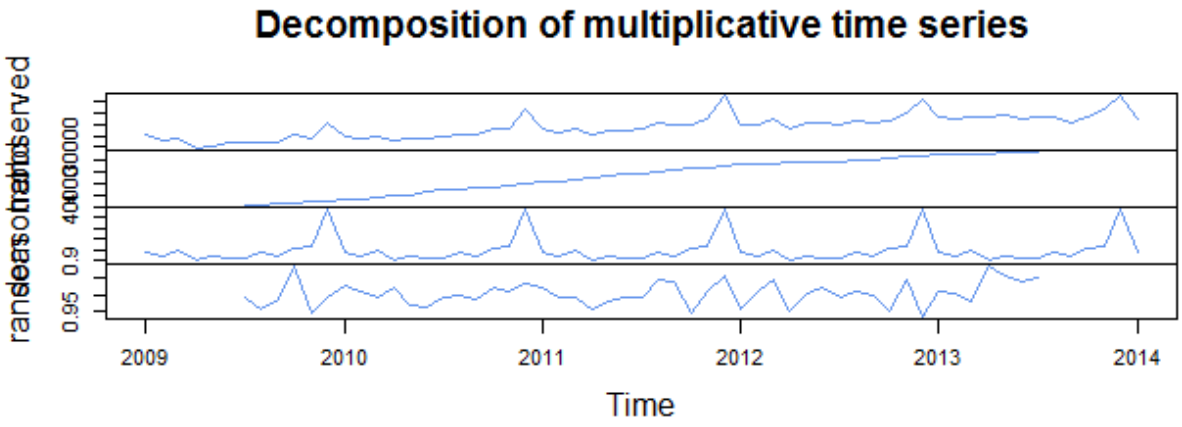


Figura 33: La función decompose().

```
plot(decompose(Autos,type="multiplicative"), col="cornflowerblue")
```

En el gráfico anterior, de arriba hacia abajo, se aprecian la serie original, la tendencia, el componente estacional y las fluctuaciones aleatorias.

Otra función que nos ayuda a visualizar los componente de una serie de tiempo es `stl()`,

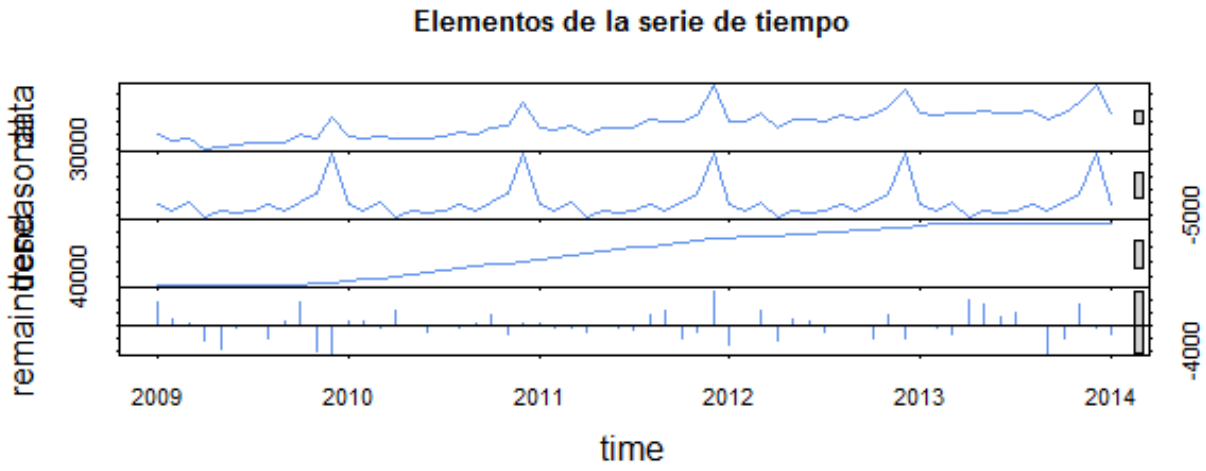


Figura 34: La función stl().

```
plot(stl(Autos, s.window="periodic"), main="Elementos de la serie de tiempo")
```

Como en el caso anterior, la función `forecast.HoltWinters()`, aplicada a la función `HoltWinters()`, nos permite pronosticar el siguiente valor de la serie.

```
> AutosProHW<-forecast.HoltWinters(HoltWinters(Autos, seasonal=c("multiplicative")), h=2, level=c(80, 95))
> AutosProHW
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Feb 2014       54873.31 51675.30 58071.31 49982.38 59764.23
Mar 2014       57948.70 54475.79 61421.61 52637.35 63260.05
```

Si almacenamos la llamada a la función HoltWinters en la variable `AutosHW`, obtenemos los valores de los parámetros,

```
> AutosHW<-HoltWinters(Autos, seasonal=c("multiplicative"))
> AutosHW
Holt-Winters exponential smoothing with trend and multiplicative seasonal component.

Call:
HoltWinters(x = Autos, seasonal = c("multiplicative"))

Smoothing parameters:
 alpha: 0.3103113
 beta : 0.1196327
 gamma: 0.806493
```

El gráfico de la serie original contra el de los valores pronosticados por la función `HoltWinters()` es el siguiente,

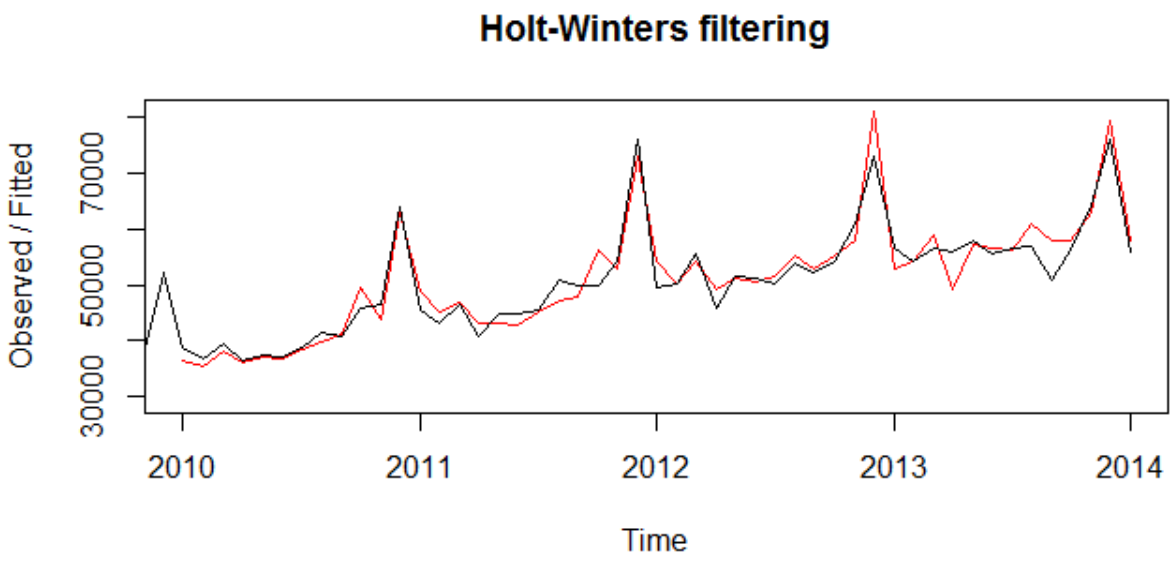


Figura 35: Serie original vs serie pronóstico.

Para verificar gráficamente el comportamiento de los residuales, llamamos a la función `plot()`, pasándole como parámetro el valor almacenado en la variable `AutosProHW$residuals`

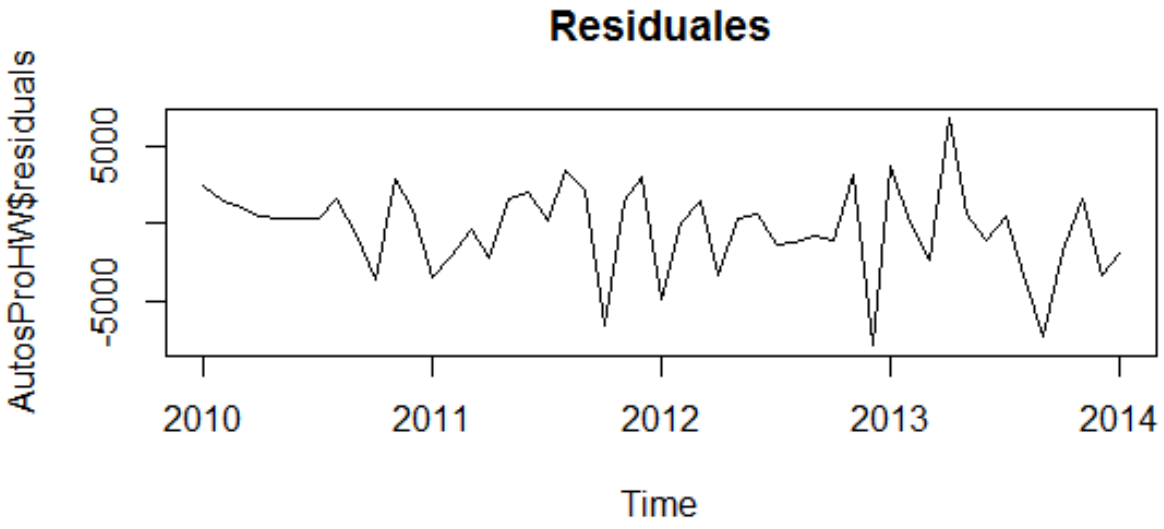


Figura 36: Residuales.

El siguiente gráfico muestra los intervalos de confianza y los valores 54,873 y 57,949, que son los valores pronósticados para los periodos de febrero y marzo de 2014, respectivamente.

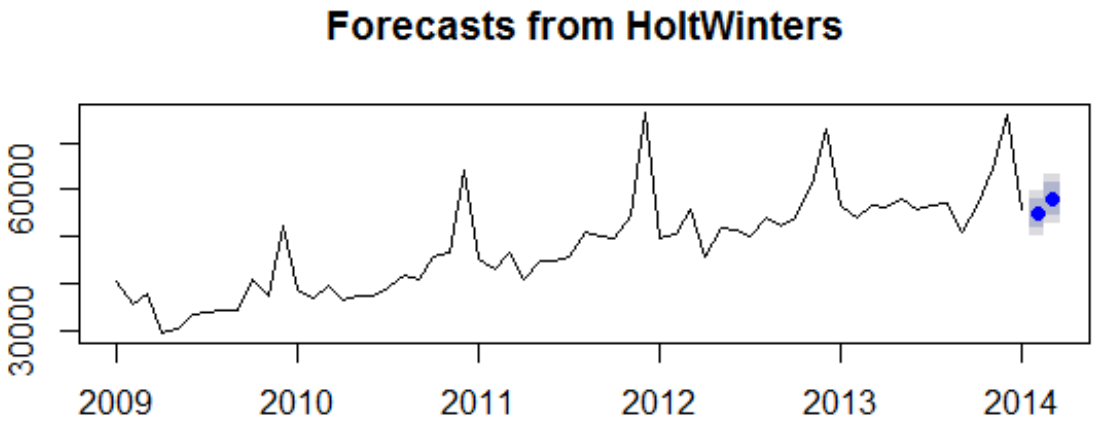


Figura 37: Pronóstico con intervalos de confianza.

Por tanto, se puede concluir que aplicando el método de Winters a la serie relacionada con la venta al público de automoviles, los valores pronosticados para los siguientes periodos son 54,873 y 57,949 unidades, respectivamente.

4.7 Metodología de Box-Jenkins

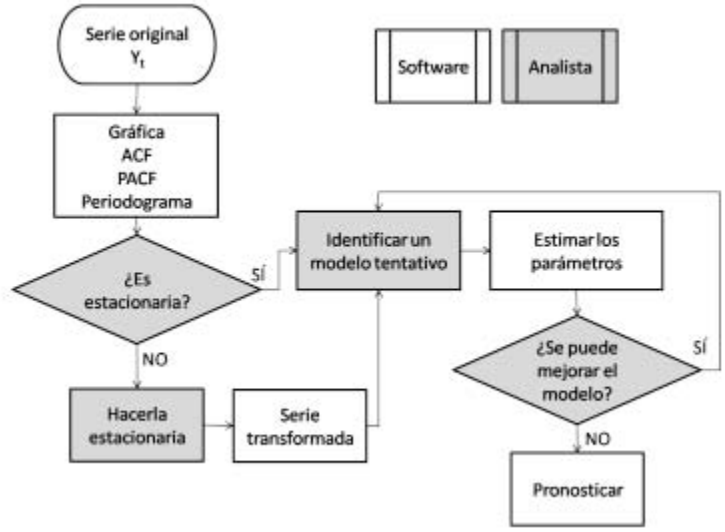


Figura 308: Diagrama funcional de la metodología Box-Jenkins

El diagrama anterior muestra los pasos a seguir para aplicar la metodología de Box-Jenkins (González Videgaray, 2011). Utilizaremos la serie relacionada con el volumen de importación de productos petrolíferos y gas natural[5] de enero de 1990 a febrero de 2014, para ejemplificar el método utilizando R.

[5] INEGI. *volumen de importación de productos petrolíferos y gas natural de enero de 1990 a febrero de 2014*. Fuente (INSTITUTO NACIONAL DE ESTADÍSTICA Y GEOGRAFÍA, 2014e)

El gráfico de la serie es el siguiente,

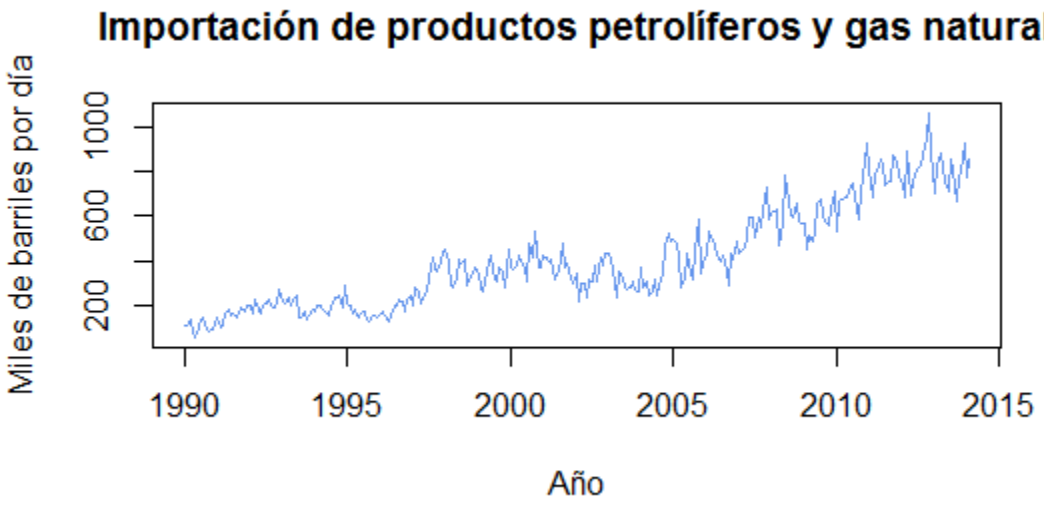


Figura 39: Volumen de importación de productos petrolíferos y gas natural

```
ImportPetroGas<-
ts(scan("C:/Users/usuario/Desktop/productos_petrolíferos_y_gas_nat
ural.csv", skip=5), frequency=12, start=c(1990, 1))

plot.ts(ImportPetroGas, main="Importación de productos
petrolíferos y gas natural", xlab="Año", ylab = "Miles de barriles
por día", col= "cornflowerblue", lwd=1)
```

ACF y PACF

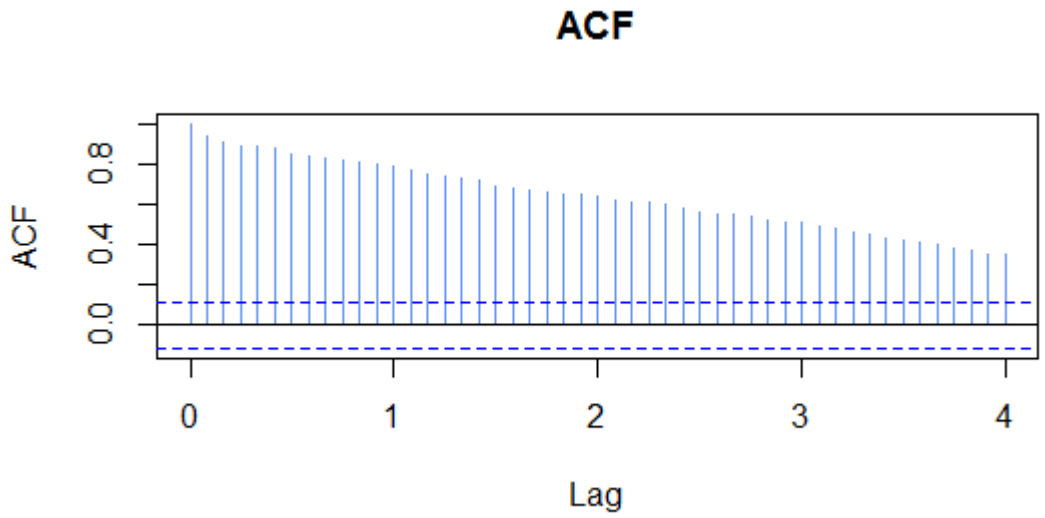


Figura 40: ACF de la serie original

```
acf(ImportPetroGas, main = "ACF", lag.max = 48, col =
"cornflowerblue")
```

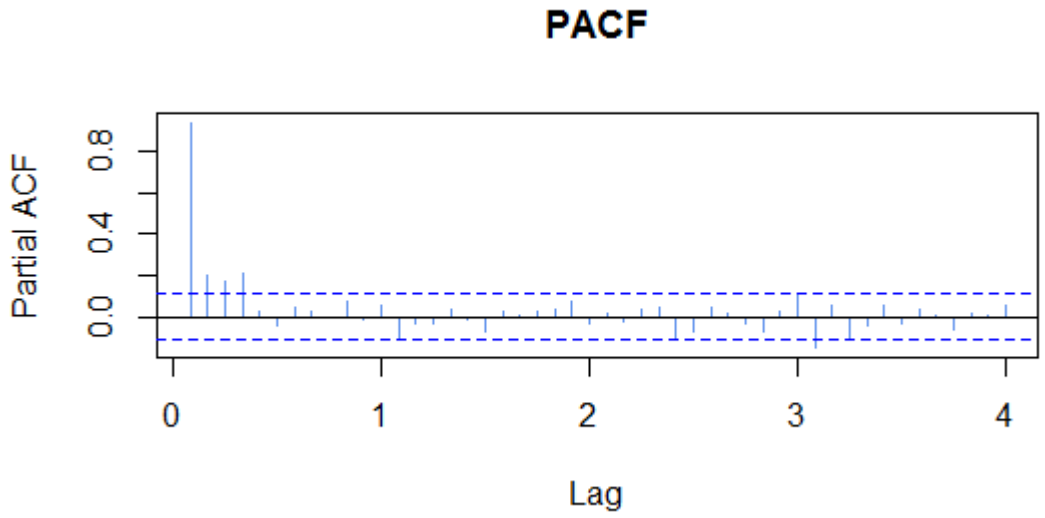


Figura 41: PACF de la serie original

```
pacf (ImportPetroGas, main="PACF",lag.max = 48, col =
"cornflowerblue")
```

Las funciones `acf()` y `pacf()` se encuentran dentro del paquete stats.

Test de Dickey–Fuller Aumentado

```
> adf.test(ImportPetroGas, alternative = "stationary")

 Augmented Dickey-Fuller Test

data:  ImportPetroGas
Dickey-Fuller = -2.7691, Lag order = 6, p-value = 0.2521
alternative hypothesis: stationary
```

La función `adf.test()` se encuentra dentro del paquete tseries.

Tanto la gráfica de la ACF como el test aumentado de Dickey – Fuller, nos permiten corroborar que la serie no es estacionaria, por lo que el siguiente paso es volverla estacionaria.

Gráficamente, la serie original no presenta evidencia de estacionalidad, por ello al aplicar una diferencia ordinaria, obtenemos el siguiente gráfico,

Figura 42: Gráfico de la serie con una diferencia ordinaria

Una función de gran utilidad es `ndiffs()`, que se encuentra dentro del paquete forecast. Aplicarla es muy sencillo y determina cuántas diferencias ordinarias son necesarias. Se utiliza de la siguiente manera,

```
ndiffs(ImportPetroGas, alpha=0.05, test=c("kpss","adf", "pp"),
max.d=2)
```

Los parámetros que recibe son cuatro, el nombre de la serie a probar, el nivel de confianza, los test de estacionaridad, y el número máximo de diferencias aceptadas. Usualmente no son necesarias más de dos diferencias. Las pruebas que utiliza son Kwiatkowski-Phillips-Schmidt-Shin (kpss), el test de Dickey–Fuller aumentado (adf) y el test de Phillips-Perron (pp).

El resultado de aplicar `ndiffs()` es un número, indicando cuantas diferencias se deben realizar para estacionalizar la serie, como se muestra a continuación,

```
> ndiffs(ImportPetroGas, alpha=0.05, test=c("kpss","adf", "pp"), max.d=2)
[1] 1
```

La función `nsdiffs()` tiene el mismo propósito que `ndiffs()` pero en este caso, determina cuántas diferencias estacionales son necesarias. Se usa de la siguiente manera,

```
nsdiffs(ImportPetroGas, m=frequency(ImportPetroGas),
test=c("ocsb","ch"), max.D=1)
```

Las pruebas que aplica esta función son, el test de Canova-Hansen (ch) y el test de Osborn-Chui-Smith-Birchenhall (ocsb).

El resultado es el siguiente,

```
> nsdiffs(ImportPetroGas, m=frequency(ImportPetroGas), test=c("ocsb","ch"), max.D=1)
[1] 0
```

Es posible utilizar la función `auto.arima()` que se encuentra dentro del paquete forecast para obtener un modelo de manera automática.

Para el caso anterior,

```
ModeloPetroGas<-auto.arima(ImportPetroGas, seasonal = FALSE)
```

La función `auto.arima()` acepta los siguientes parámetros:

```
auto.arima(x, d=NA, D=NA, max.p=5, max.q=5,
max.P=2, max.Q=2, max.order=5, max.d=2, max.D=1,
start.p=2, start.q=2, start.P=1, start.Q=1,
stationary=FALSE, seasonal=TRUE,
ic=c("aicc","aic", "bic"), stepwise=TRUE, trace=FALSE,
approximation=(length(x)>100 | frequency(x)>12), xreg=NULL,
test=c("kpss","adf","pp"), seasonal.test=c("ocsb","ch"),
allowdrift=TRUE, lambda=NULL, parallel=FALSE, num.cores=2)
```

Para más detalles es posible consultar la documentación del paquete forecast en la siguiente dirección electrónica:

http://cran.r-project.org/web/packages/forecast/forecast.pdf

Al aplicar dicha función en la serie que se ha trabajado, se obtienen los siguientes resultados,

```
Series: ImportPetroGas
ARIMA(1,1,1) with drift

Coefficients:
         ar1      ma1   drift
      0.3789  -0.8454  2.5156
s.e.  0.0846   0.0517  0.9419

sigma^2 estimated as 3996:  log likelihood=-1602.67
AIC=3213.35   AICc=3213.49   BIC=3228.01
```

Para pronosticar, se utiliza la función `forecast()`, que se encuentra dentro del paquete forecast.

Los parámetros que acepta son los siguientes,

```
forecast(object, h = ifelse(frequency(object) > 1, 2 *
frequency(object), 10) ,

level=c(80,95), fan=FALSE, robust=FALSE, lambda=NULL,
find.frequency=FALSE, ...)
```

donde `object` es la serie de tiempo a pronosticar, `h` es el número de periodos a pronósticar y `level` es el nivel de confianza.

El resultado de aplicar la función `forecast()`, pronosticando dos periodos hacia el futuro es el siguiente,

```
> PetroPro<-forecast(ModeloPetroGas, h=2, level=c(80,95))
> PetroPro
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Mar 2014       839.4046 758.3918 920.4173 715.5063 963.3028
Apr 2014       836.9528 745.1314 928.7742 696.5240 977.3816
```

Graficando el pronóstico,

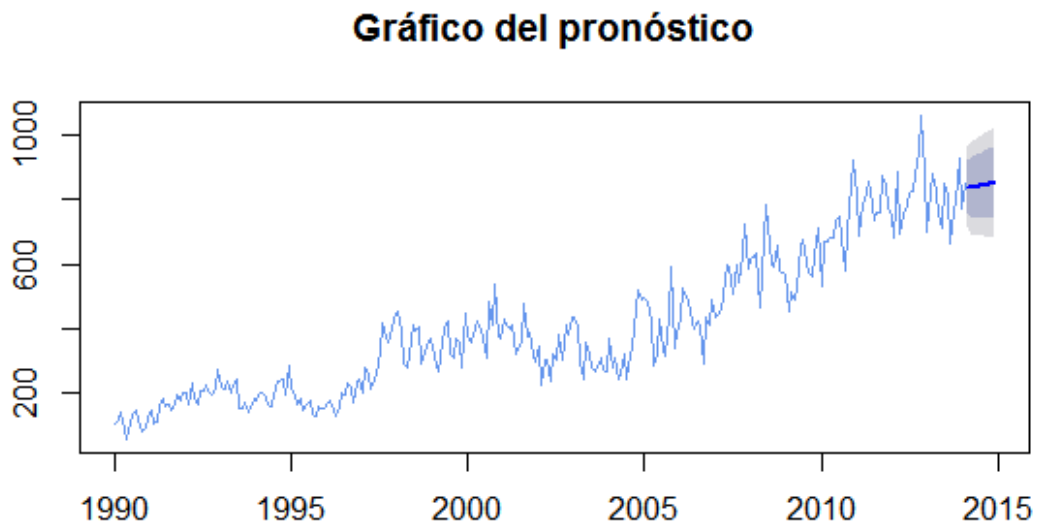


Figura 43: Gráfico del pronóstico para el volumen de importación de productos petrolíferos y gas natural

```
plot(PetroPro, main = "Gráfico del pronóstico", col =
"cornflowerblue")
```

Es posible elegir un modelo propio, y no el que sugiere la función `auto.arima()`, utilizando la función `Arima()`, de la siguiente manera,

```
ARIMA111<-Arima(ImportPetroGas, order = c(1,1,1), include.drift =
TRUE)
```

Y obtenemos el siguiente resultado

```
> ARIMA111
Series: ImportPetroGas
ARIMA(1,1,1) with drift

Coefficients:
         ar1      ma1   drift
      0.3789  -0.8454  2.5156
s.e.  0.0846   0.0517  0.9419

sigma^2 estimated as 3996:  log likelihood=-1602.67
AIC=3213.35   AICc=3213.49   BIC=3228.01
```

Observamos que el resultado al usar esta función es el mismo que al usar `auto.arima()`.

Podemos vizualizar el periodograma de los residuales, primero aplicando la función `forecast()` al modelo obtenido con `Arima()`,

```
ARIMA111Pro<-forecast(ARIMA111, h=10, level=c(80, 95))
```

Y después utilizando la función `cpgram()`,

```
cpgram(ARIMA111Pro$residuals, main="Periodograma integrado de los
              residuales", ci.col="cornflowerblue")
```

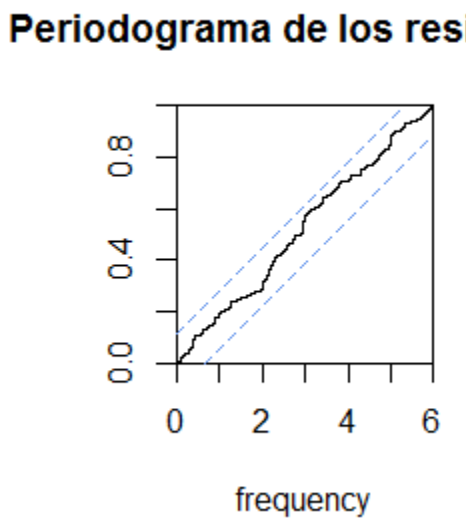


Figura 44: Periodograma de los residuales

Así, se ha podido determinar, que los valores futuros de la serie actual son los siguientes,

```
> PetroPro<-forecast(ModeloPetroGas, h=2, level=c(80,95))
> PetroPro
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Mar 2014       839.4046 758.3918 920.4173 715.5063 963.3028
Apr 2014       836.9528 745.1314 928.7742 696.5240 977.3816
```

Utilicemos la seie de tiempo referente al consumo de energía electríca de uso doméstico de enero de 2000 a febrero de 2014[6].

El gráfico es el siguiente,

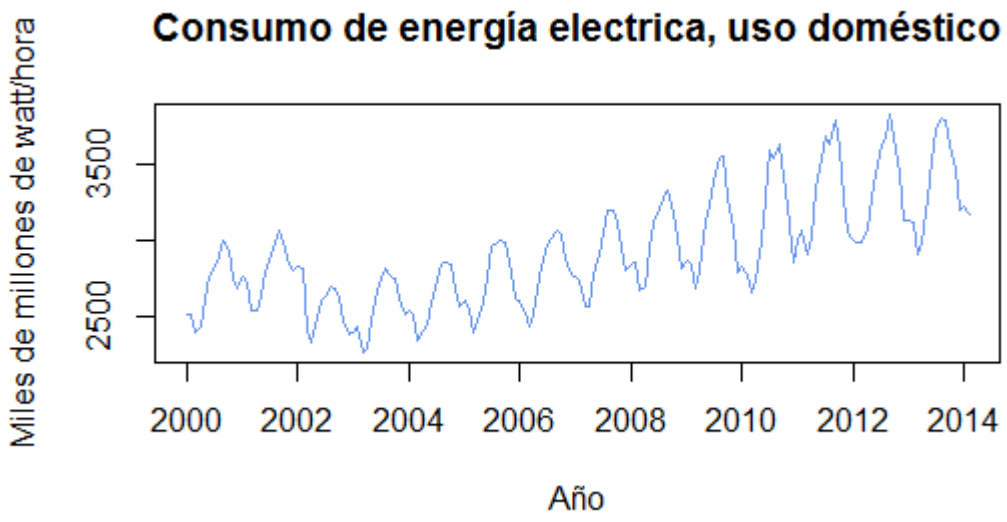


Figura 45: Consumo de energía eléctrica, uso doméstico

```
ConsumoEnergia<-
ts(scan("C:/Users/usuario/Desktop/ConsumoEnergiaDomestico.csv",
skip=4), frequency=12, start=c(2000, 1))

plot.ts(ConsumoEnergia, main="Consumo de energía electrica, uso
doméstico", xlab="Año", ylab = "Miles de millones de watt/hora",
col= "cornflowerblue", lwd=1)
```

Del gráfico es posible ver que la serie presenta tendencia creciente y estacionalidad.

Es posible utilizar la función `auto.arima()` cuando la serie presenta estacionalidad,

[6] INEGI. *Consumo de energía electríca de uso domestico de enero de 2000 a febrero de 2014.* Fuente (INSTITUTO NACIONAL DE ESTADÍSTICA Y GEOGRAFÍA, 2014a)

```
> ModeloConsumoEner<-auto.arima(ConsumoEnergia)
> ModeloConsumoEner
Series: ConsumoEnergia
ARIMA(1,1,1)(0,1,1)[12]

Coefficients:
          ar1      ma1     sma1
      0.8735  -0.9774  -0.5525
s.e.  0.0741   0.0453   0.0773

sigma^2 estimated as 4274:  log likelihood=-882.02
AIC=1772.04   AICc=1772.31   BIC=1784.27
```

Como se observa, la función propone un modelo $ARIMA(1,1,1)(0,1,1)_{12}$

La función `diff()`, nos permite realizar diferencias ordinarias y estacionales a la serie, y utilizándola junto con la función `tsdisplay()`, que se encuentra dentro del paquete forecast, permite graficar una serie de tiempo junto con su ACF y PACF.

La serie con una diferencia estacional de orden 12,

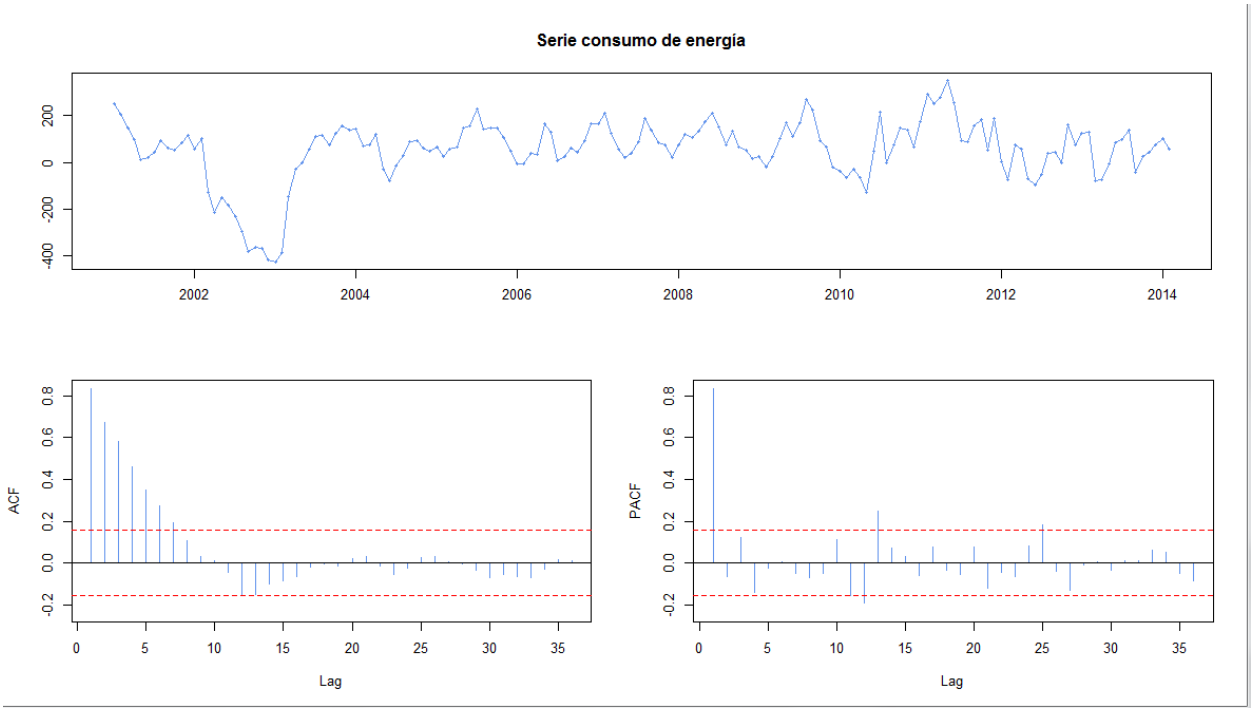


Figura 45: Serie con una diferencia estacional de orden 12, ACF y PACF

```
tsdisplay(diff(ConsumoEnergia, 12), main = "Serie consumo de
energía", col = "cornflowerblue", ci.col="red")
```

La serie con una diferencia estacional de orden 12 y una diferencia ordinaria

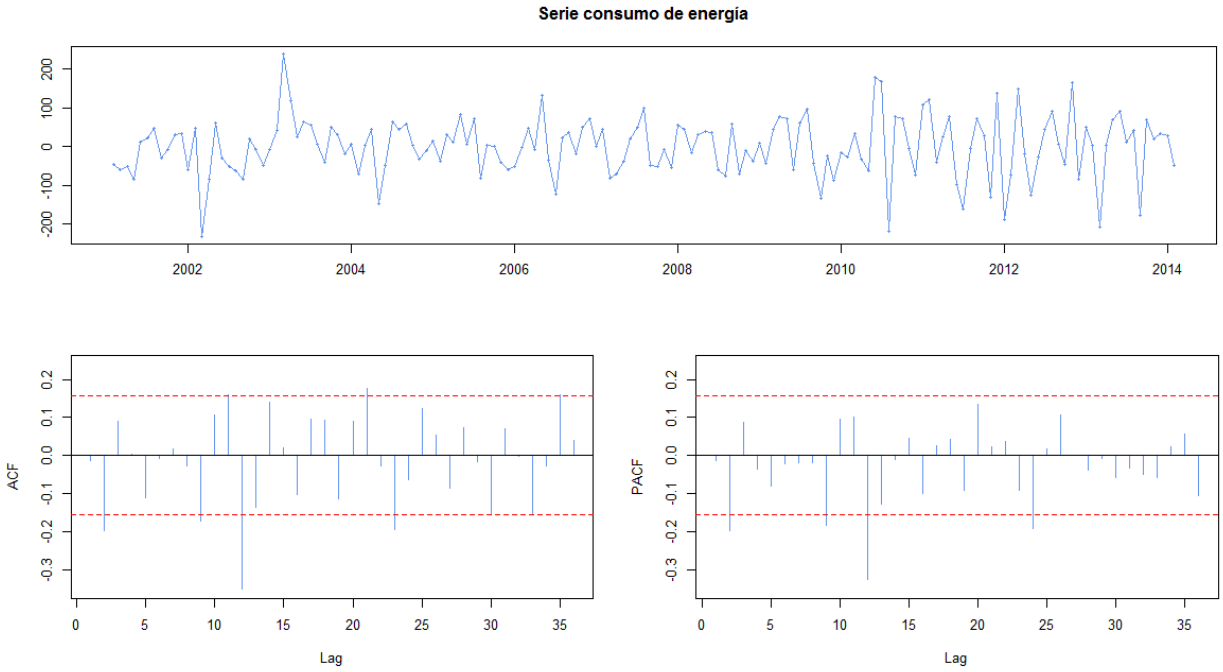


Figura 46: Serie con una diferencia estacional de orden 12 y una diferencia ordinaria, ACF y PACF

```
tsdisplay(diff(diff(ConsumoEnergia, 12)), main = "Serie consumo de
energía", col = "cornflowerblue", ci.col="red")
```

Se puede utilizar la función `Arima()`, al igual que en el ejemplo anterior, para proponer diferentes modelos y optar por el mejor,

```
> ModeloEner2<-Arima(ConsumoEnergia, order = c(1,1,1), seasonal = c(1,1,1))
> ModeloEner2
Series: ConsumoEnergia
ARIMA(1,1,1)(1,1,1)[12]

Coefficients:
          ar1     ma1    sar1     sma1
      -0.5009  0.5387  0.1305  -0.6514
s.e.   0.4458  0.4322  0.1463   0.1132

sigma^2 estimated as 4439:  log likelihood=-884.39
AIC=1778.78   AICc=1779.18   BIC=1794.06
```

El mejor modelo, lo ofrece en este caso, la función `auto.arima()`, aunque no siempre es así. Es preferible por ello trabajar en los modelos utilizando la simpleza de las funciones de R pero comprobando y mejorando el modelo de acuerdo con la teoría.

El pronóstico se muestra a continuación,

```
> EnerPro<-forecast(ModeloConsumoEner, h=10, level=c(80,95))
> EnerPro
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Mar 2014       3034.637 2950.852 3118.423 2906.498 3162.776
Apr 2014       3122.716 3010.210 3235.222 2950.654 3294.779
May 2014       3391.868 3260.676 3523.061 3191.227 3592.510
Jun 2014       3604.816 3460.202 3749.431 3383.648 3825.985
Jul 2014       3803.077 3648.342 3957.813 3566.429 4039.725
Aug 2014       3845.661 3683.054 4008.267 3596.976 4094.346
Sep 2014       3911.515 3742.648 4080.381 3653.256 4169.774
Oct 2014       3725.026 3551.091 3898.962 3459.015 3991.038
Nov 2014       3538.966 3360.862 3717.071 3266.579 3811.354
Dec 2014       3259.768 3078.187 3441.348 2982.064 3537.471
```

Y enseguida, el gráfico del pronóstico,

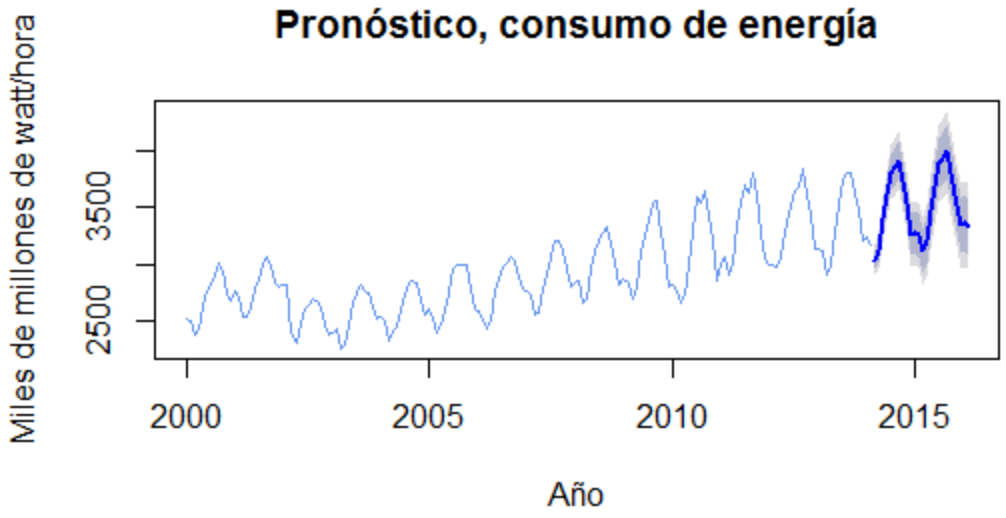


Figura 47: Gráfico del pronóstico del consumo de energía electríca de uso domestico

```
plot(forecast(ModeloConsumoEner), main = "Pronóstico, consumo de
energía", ylab = "Miles de millones de watt/hora", xlab = "Año",
col = "cornflowerblue")
```

Conclusiones

Muchas veces nos encontramos en situaciones en las que debemos hacer uso de herramientas de las cuales tan sólo hemos escuchado por mera referencia. Y sucede que a la hora de utilizarlas, no sabemos por dónde comenzar.

R es una aplicación basta, en el sentido de que existen numeros usos y constantemente, se podría decir que a diario, se desarrollan nuevas funciones, nuevos paquetes que extienden su aplicación, la mayoría de las ocaciones en situaciones que son ajenas a nuestra área del conocimiento.

Por lo anterior, este trabajo cumple con su objetivo al centrarce en un campo específico de acción haciendo uso de una herramienta libre. Sin embargo, el presente no es más que la punta de un iceberg en lo concerniente a R, así como de los métodos estadísticos tocados por él.

Otro propósio que cumple es el de alentar el uso de una herramienta que en principio de cuentas resulta desagradable si la comparamos con las aplicaciones en las que su funcionalidad está ligada a interfaces muy gráficas y llenas de botones.

Este trabajo muestra que más allá de ser de uso sencillo, las herramientas libres son de propósito general.

Un punto que no se debe pasar por alto es la capacidad del software libre de incentivar la cooperación, de tener acceso a lo que ya está hecho, de tener la oportunidad de entender cómo se hizo y utilizar aquello para lograr los fines que se persigan con toda libertad.

Por último mencionemos el gran mérito que tiene la formación que brinda la Licenciatura en Matemáticas Aplicadas y Computación (MAC), ya que encaja perfectamente en proyectos de esta naturaleza, en donde es posible aplicar las ramas de la matemática y en donde la computación juega un papel decisivo.

Bibliografía

Box, G. E. P., & Jenkins, G. M. (1970). *Time series analysis : forecasting and control*. San Francisco: Holden-Day.

Cooray, T. M. J. A. (2008). *Applied Time Series : Analysis and Forecasting*. Oxford, UK.: Alpha Science International.

Ericsson. (2014). *Connected Tree*, Fecha de consulta: 02/04/2014. http://www.ericsson.com/thinkingahead/visionary-ideas/connected-tree

Free Software Foundation. (2014). *Free Software Foundation* Fecha de consulta: 13/02/2014. www.fsf.org

Gardener, M. (2012). *Beginning R : the statistical programming language*. Indianapolis: John Wiley & Sons.

Gimp. (2013). *Feature Overview*, Fecha de consulta: 12/03/2014. http://www.gimp.org/features/

GNU. (2014). *¿Qué es el software libre?* Fecha de consulta: 12/03/2014. https://www.gnu.org/philosophy/free-sw.es.html

González Videgaray, M. (2011). *Pronósticos: Metodología de Box-Jenkins*. México: UNAM FES Acatlán.

INSTITUTO NACIONAL DE ESTADÍSTICA Y GEOGRAFÍA. (2014a). *Consumo de energía electríca de uso domestico en México,* Fecha de consulta: 05/04/2014. www.inegi.org.mx

INSTITUTO NACIONAL DE ESTADÍSTICA Y GEOGRAFÍA. (2014b). *Número de vehículos particulares de motor registrados en circulación en México de enero de 2008 a enero de 2014,* Fecha de consulta: 05/04/2014. www.inegi.org.mx

INSTITUTO NACIONAL DE ESTADÍSTICA Y GEOGRAFÍA. (2014c). *Tasa de desocupación mensual en México*, Fecha de consulta: 05/04/2014 www.inegi.org.mx

INSTITUTO NACIONAL DE ESTADÍSTICA Y GEOGRAFÍA. (2014d). *Venta al público del total de automoviles en México*, Fecha de consulta: 05/04/2014. www.inegi.org.mx

INSTITUTO NACIONAL DE ESTADÍSTICA Y GEOGRAFÍA. (2014e). *Volumen de importación de productos petrolíferos y gas natural en México*, Fecha de consulta: 05/04/2014. www.inegi.org.mx

Joyanes Aguilar, L., & Zahonero, I. (2010). *Programación en C, C++, Java y UML*. México etc: McGraw-Hill.

LibreOffice. (2014). *What is LibreOffice,* Fecha de consulta: 12/03/2014. http://www.libreoffice.org/discover/libreoffice/

Mozilla. (2014). *Get to know Mozilla,* Fecha de consulta: 12/03/2014. https://www.mozilla.org/en-US/about/?icn=tabz

National Geographic Society. (2014a). *Cambio climático, sequías e inundaciones,* Fecha de consulta: 23/02/2014. http://www.nationalgeographic.es/medio-ambiente/aguas-dulces/climage-change

National Geographic Society. (2014b). *¿Qué es el Calentamiento Global?* Fecha de consulta: 23/02/2014. http://nationalgeographic.es/medio-ambiente/calentamiento-global/calentamiento-global-definicion

Oracle. (2014). *Why MySQL?*, Fecha de consulta: 12/03/2014. http://www.mysql.com/why-mysql/

OSI. (2014). *The Open Source Definition,* Fecha de consulta: 23/02/2014. http://opensource.org/osd

Php. (2014). *php,* Fecha de consulta: 12/03/2014. www.php.net

The R Project. (2014). *What is R?,* Fecha de consulta: 01/20/2014. http://www.r-project.org/

Wei, W. W. S. (1990). *Time series analysis : univariate and multivariate methods*. Redwood City, Calif.: Addison-Wesley Pub.