UNIVERSIDAD DE GRANADA

MÁSTER DE CIENCIA DE DATOS E INGENIERÍA DE COMPUTADORES

CURSO ACADÉMICO 2019-2020

SERIES TEMPORALES Y MINERÍA DE FLUJO DE DATOS

Trabajo autónomo I: Series Temporales.

Análisis y predicción con series temporales mensuales y diarias de las temperaturas máximas registradas en la estación meteorológica de la AEMET en Montoro (Córdoba).

Nicolás Cubero

30 de Abril de 2020

Índice

Índice de figuras

1. Introducción

En el presente proyecto se pretende llevar a cabo el tratamiento de la evolución de las temperaturas máximas registradas Enero de 2013 y Febrero de 2018 en la localidad de Montoro (Córdoba) y predecir las temperaturas máximas en la primera semana de Marzo de 2018 y en los meses de Mayo y Abril de 2018.

Se basará este estudio, en los datos meteorológicos recopilados por la estación de la *AEMET* ubicada en Montoro (Córdoba) entre Enero de 2013 y Febrero de 2018 (recopilación `5361X`), a partir de los cuales, se tomará la **serie temporal diaria** y la **serie temporal mensual** sobre las temperaturas máximas registradas en esta localidad.

Se desea aclarar que se han elegido los datos registrados en esta localidad por su mayor proximidad a la localidad de residencia particular.

Se realizará un preprocesamiento, análisis y tratamiento preliminar de cada serie temporal y se elaborarán modelos predictivos a partir de cada serie para predecir, la evolución de la temperatura máxima en Montoro durante la primera semana de Marzo de 2018 (en este caso se usará la serie temporal diaria) y la evolución de la temperatura máxima en los meses de Marzo y Abril de 2018 (para lo que se usará la serie temporal mensual).

2. Fundamentos teóricos

Primeramente, se exponen los fundamentos teóricos necesarios [1] tanto para desarrollar este proyecto como para entender su resolución.

Se comienza tratando el concepto de serie temporal:

Una **serie temporal** es una serie de mediciones de una determinada magnitud a lo largo del tiempo, en un determinado periodo. La serie temporal refleja por tanto, la evolución de esta magnitud respecto al tiempo.

En el estudio de series temporales, la serie temporal se descompone en las siguientes componentes:

- **Tendencia**: Presencia de incrementos o decrementos en los valores de la serie a largo plazo.

- **Ciclo**: Aparición de crecimientos o decrecimientos de los valores de la serie en una frecuencia no fija.

  Se suele tratar conjuntamente con la Tendencia, dando lugar a una componente combinada denominada **Tendencia-Ciclo** o simplemente Tendencia.

  En lo sucesivo, se trabajará bajo esta aceptación de Tendencia, de forma que ambas componentes se estudiarán y se tratarán conjuntamente como una única componente.

- **Estacionalidad**: Consiste en la repetición de un determinado patrón en la serie en periodos de tiempo fijos.

- **Residuo**: Componente irreductible de la serie.

2.1. Preprocesamiento

Las técnicas de preprocesamiento que se aplicarán en este proyecto, se centrarán en la imputación de los valores perdidos de la serie y en el cómputo de la serie mensual a partir de los datos de registro diario de la serie.

2.1.1. Imputación de valores perdidos

La imputación de valores perdidos en serie stemporales implica la aplicación de algoritmos específicos de estimación de valores perdidos que, a diferencia de los algoritmos de imputación tradicionales, asumen la progresión de los datos respecto al tiempo, es decir, las medidas en un determinado instante guardan relación con las medidas en instantes anteriores y centran esta estimación en la información de esta progresión.

Entre las medidas de imputación para series temporales, se muestra interés por los métodos de **imputación por interpolación**.

La **imputación por interpolación**, trata de estimar cada valor perdido en base a los valores de instantes anteriores y posteriores que se encuentren definidos (interpolan dicho valor ajustando alguna función regresora con los valores de los instantes anteriores y posteriores y suponen que el valor perdido se encuentra en el valor central dicha función).

Entre los métodos de interpolación para imputación de series temporales, podemos encontrar los siguientes:

- Interporlación lineal: Asume que cada valor perdido se encuentra en la recta situada entre los valores de los instantes inmediatamente anteriores y posteriores definidos.
- Interpolación por *splines*: Cada valor perdido es estimado mediante *splines* polinómicos que se ajustan a partir de los valores de los instantes anteriores y posteriores al valor perdido que se hallen definidos.
- Interpolación *stine*: Cada valor perdido se estima a partir de la función de *Stinerman* estimada a partir de los instantes anteriores y posteriores al valor perdido que se hallen definidos.

2.1.2. Selección de características

La selección de características en preprocesamiento consiste en la seleción de un subconjunto de variables dentro del conjunto original de variables del *dataset*.

Constituye un mecanismo de reducción de datos que se emplea para desechar variables que se consideran irrelevantes y que, o bien, no aportan utilidad en los algoritmos de minería de datos o bien, introducen ruido.

La selección de características resulta de utilidad para la obtencion de los datos de la serie temporal de las temperaturas máximas diarias a partir del conjunto de variable recogidas en el *dataset* original. En este proceso, se seleccionaría de este *dataset* la columna que registra la información sobre las temperaturas máximas.

2.1.3. Resumen de características

El resumen de características se refiere al cómputo de estadísticos resumen sobre las medidas evaluadas a lo largo de cada periodo de tiempo para la obtención de una medida representativa de cada periodo de tiempo.

En este proyecto, se requeriá usar este preprocesamiento para resumir todas las mediciones de temperatura máxima registradas durante de todos los días de cada mes en una única métrica que refleje la temperatura máxima global en cada mes.

2.2. Análisis de tendencia y estacionalidad

Para **determinar la existencia de tendencia** en la serie, principalmente se hará uso de la representación gráfica de la serie y se tratará de detectarla a simple vista. En caso de que la tendencia se aproxime a una función conocida, se tratará de ajustar un modelo de regresión que estimara esta tendencia y proceder a su eliminación de la serie.

Otro método para estimar y determinar la tendencia de forma analítica consiste en el **filtrado por medias móviles**:

El **filtrado por medias móviles** trata de estimar la tendencia en cada instante de tiempo t mediante el cálculo de la media de los valores que se encuentran en el mismo periodo que el instante t.

Este filtrado logra eliminar parte de la aleatoriedad de los datos obtiendo una componente de tendencia suavizada.

Se define para un orden m siendo $m = 2k + 1$ donde k es el número de períodos de la serie, y se determina por la expresión:

$$\hat{T}_t = \frac{1}{m} * \sum_{j=-k}^{k} y_{t+j} \tag{1}$$

Para la **detección de estacionalidad**, una vez estimada y eliminada la tendencia, también se hará uso de la representación gráfica de la serie para tratar de visualizarla a simple vista y determinar su periodo.

La estacionalidad se estima para cada instante del periodo como el cómputo de las medias de todos los valores de la serie que se encuentran en el mismo periodo:

$$\forall i \in \{1, 2, ..., k\} \hat{t}_i = \frac{1}{k} * \sum_{j=0}^{k} t_{i+jk} \tag{2}$$

2.3. Estacionariedad

Una serie es **estacionaria** cuando sus medidas estadísticas no varían a lo largo del tiempo y por tanto, se puede garantizar la independencia de la serie respecto del instante de tiempo en la que fue capturada.

Para determinar la estacionariedad de la serie, principalmente se hará uso

del **test de *Dicker-Fuerier* aumentado** o **test ADF**, que constituye una prueba de raíz unitaria que, basándose en la deteción de reversiones en la media entre un valor y el valor en el instante anterior, **recoge en su hipótesis nula la no estacionalidad de la serie**. Por consiguiente, **una serie estacionaria llevará al fallo del test**.

Otra forma de determinar la estacionariedad de la serie consiste en hacer uso de su gráfico ACF y/o su gráfico PACF.

La **función de autocorrelación** o **función ACF** mide la correlación lineal entre la serie y la serie retardada t instantes de tiempo. En base a la función de autocorrelación, se construye el **gráfico ACF** o **gráfico de autocorrelaciones**, que refleja mediante líneas verticales el grado de autocorrelación existente entre el instante actual y los retardos sucesivos.

La **función de autocorrelación parcial** o **función PACF** por su parte, mide la **correlación parcial** entre la serie y la serie retardada t instantes de tiempo. En base a esta medida se construye también, el **gráfico PACF** o **gráfico de autocorrelaciones parciales**, del mismo modo que el gráfico ACF.

Si una serie es no estacionaria, la autocorrelación y autocorrelación parcial decrecen de forma próxima casi lineal al aumentar el número de retardos de forma no muy pronunciada, por lo que en los gráficos ACF y PACF se observaría un **descenso de la autocorrelación constante, progresivo y lento**. Ante estacionariedad, la autocorrelación y autocorrelación parcial decrecen rápidamente y alcanzan valores nulos y/o negativos en pocos retardos.

En el caso de no estacionariedad, se pueden aplicar diversas transformaciones a la serie para lograr la estacionariedad:

- Aplicar una **transformación logarítmica** a la serie para tratar de estabilizar su varianza a lo largo del tiempo.
- **Diferenciar la serie** con valor anterior para tratar de estabilizar la media a lo largo del tiempo:

  Se suele aplicar una primera diferenciación de la serie y si no se logra la estacionariedad de la serie, se puede tratar de diferenciar una segunda vez, o aplicar diferenciación estacional, consistente en restar a cada valor el valor del instante anterior en el periodo de la serie.

2.4. Modelado de series temporales

En este proyecto se trata el modelado de series temporales mediante el ajuste de **modelos ARIMA** (*AutoRegressive Integrated Moving Average*) no estacionales de órdenes p, d y q.

El modelo ARIMA constituye una combinación de los **modelos autorregresivos** (AR) de orden p y los **modelos de medias móviles** (MA) de orden q:

- Los **modelos autorregresivos** (AR) tratan de ajustar un polinomio de regresión múltiple que permita predecir los valores de la serie en el instante actual en base a los p valores de la serie en los instantes anteriores. El polinomio de regresión considera como variables dependientes los valores de la serie desde el instante anterior hasta p instantes anteriores.

  El modelo autorregresivo AR(p) se formula de la siguiente forma:

  $$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_t \tag{3}$$

  donde ε_t representa los errores aleatorios del modelo.

- Los **modelos de medias móviles** (MA) tratan de ajustar un polinomio de regresión múltiple considerando los q errores cometidos en la predición de los instantes anteriores para predecir el valor de la serie en el instante actual. El polinomio de regresión considera como variables dependientes los errores de la serie desde el instante anterior hasta q instantes anteriores.

  El modelo de medias móviles MA(q) se formula de la siguiente forma:

  $$y_t = c + \phi_1 \varepsilon_{t-1} + \phi_2 \varepsilon_{t-2} + \ldots + \phi_p \varepsilon_{t-q} \tag{4}$$

El modelo ARIMA combina estos modelos y, en el ajuste admite la diferenciación de los valores retardado un número de veces d, es decir, **admite que la serie sea diferenciada**.

La expresión genérica de un modelo ARIMA(p,d,q) es la siguiente:

$$y'_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q} + \varepsilon_t \tag{5}$$

Donde y'_t es la serie diferenciada d veces.

En resumen, los parámetros del modelo ARIMA son los siguientes:

- **p**: Orden del la parte autorregresiva.
- **d**: grado de diferenciación (número de diferenciaciones llevadas a cabo en la serie).
- **q**: Orden de la parte de medias móviles.

Para ajustar un modelo ARIMA no estacionario sobre una serie temporal, **se requiere que la serie no presente componentes de tendencia, ni de estacionalidad y que sea estacionaria**.

Se requiere por tanto, realizar previamente un estudio de tendencia y estacionalidad de la serie, tal y como se explicó en la sección anterior y, en el caso en el que la serie presentara alguna de estas componentes, se debe proceder a su estimación y eliminación hasta lograr una serie sin tendencia ni estacionalidad (serie únicamente con la componente residual).

Para acabar, se requiere también que la serie sea estacionaria, en caso contrario, se procedería a utilizar alguna de las técnicas explicadas en la sección anterior y, en el caso de que se requiera realizar d diferenciaciones de la serie, se entrenaría el modelo ARIMA con la serie sin diferenciar y proporcionando este número de diferenciaciones en el parámetro d del modelo.

Por último, para determinar los parámetros p y q se pueden seguir las siguientes reglas:

Se preferirá un modelo ARIMA(p,d,0) (sin componente de medias móviles) si aparecen los siguientes patrones en el gráfico ACF:

- El gráfico ACF presenta un descenso senoidal o exponencial de la autocorrelación conforme aumentan los *lags*.
- El gráfico ACF presenta una gran autocorrelación para un determinado *lag* p tras el cual no se percibe ningún otro *lag* que ofrezca una autocorrelación igual o superior.

En el caso en el que se detecten estos patrones, se tomaría como parámetro p el *lag* que ofrece la mayor autocorrelación con el valor actual.

Se consideraría, por el contrario un modelo ARIMA (0,d,q) (sin componente autorregresiva) si aparecen los siguientes patrones en el gráfico PACF:

- El gráfico PACF presenta un descenso senoidal o exponencial de la autocorrelación conforme aumentan los *lags*.
- El gráfico PACF presenta una gran autocorrelación parcial para un determinado *lag* q tras el cual no se percibe ningún otro *lag* que ofrezca una autocorrelación igual o superior.

En este caso, se tomaría como parámetro q el *lag* que ofrece la mayor autocorrelación parcial con el valor actual.

Una vez estimado el modelo ARIMA (p,d,q) no estacional, se procedería a realizar prediciones en un conjunto de instantes futuros. Para obtener los valores reales de predición, se procedería a sumar a esta predición la componente de tendencia y estacionalidad estimadas y correspondientes a los instantes de tiempo futuros.

3. Predicciones temporales

Como bien se indicó con anterioridad, se parte de los registros de temperaturas máximas diarias desde Mayo de 2013 a Febrero de 2018 del registro `5361X` de la *AEMET* en la localidad de Montoro (Córdoba).

Procedemos a cargar los datos en R y a realizar un análisis preliminar de los mismos:

```
# Obtener número de registros
cat('Número de registros en el dataset: ', length(met$Tmax), fill=TRUE)

# Intervalo de registros
cat(paste('Intervalo de fechas: ', min(met$Fecha),' - ', max(met$Fecha)
    ), fill=TRUE)

# Realizar un breve resumen de los estadísticos
summary(met$Tmax)

# Calcular la desviación típica
cat('Desvicación típica de Temperaturas máximas: ', sd(met$Tmax, na.rm=
    T), fill=TRUE)
```

Script 1: Sentencias destinadas a cargar el fichero 5361X, a obtener el número de registros en el mismo y los principales estadísticos

Exponemos la información reflejada por la salida de estas sentencias en la siguiente tabla:

Medida	Valor
Intervalo de fechas	7/5/2013 - 28/2/2018
Nº de registros	1754
Valores perdidos	139
Mínimo	6
1er cuartil	18.20
Mediana	26.9
Media	26.71
3er cuartil	34.5
Máximo	47.3
Desvicación típica	9.322056

Cuadro 1: Estadísticos y datos básicos extraídos del dataset 5361X

Representamos una gráfica con la serie de las temperaturas máximas:

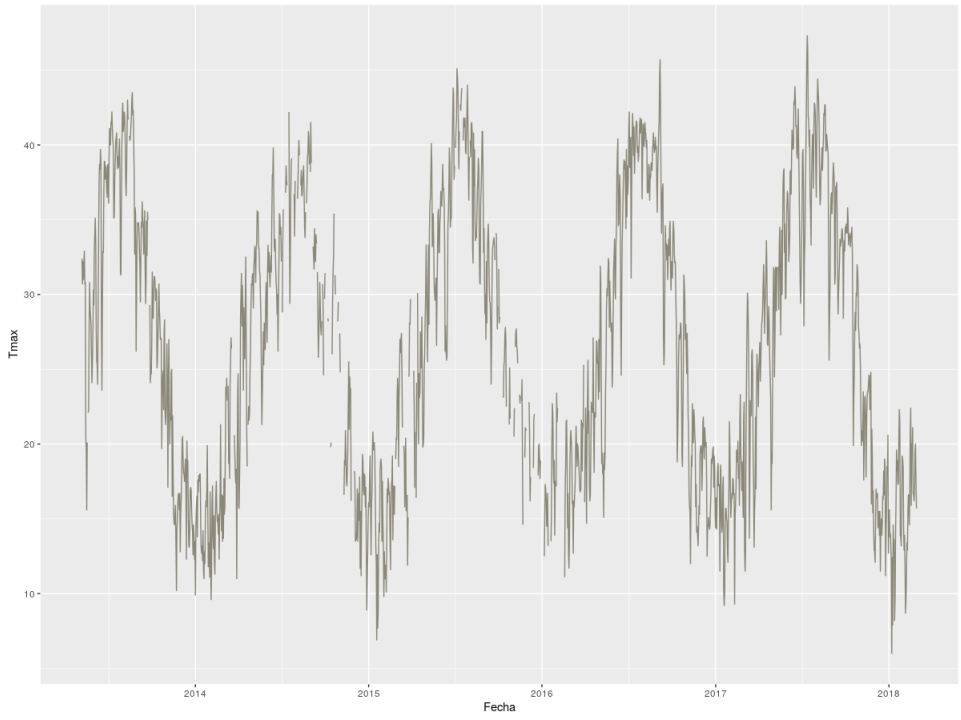

Figura 1: Representación gráfica de la evolución de las temperaturas máximas entre Mayo de 2013 y Febrero de 2018.

Esta gráfica nos muestra una serie **carente de tendencia** y con una **estacionalidad que se asemeja a una función senoidal**.

Por otra parte, para analizar y entender mejor cómo están distribuídos los valores perdidos en la serie, nos ayudamos de la siguiente representación:

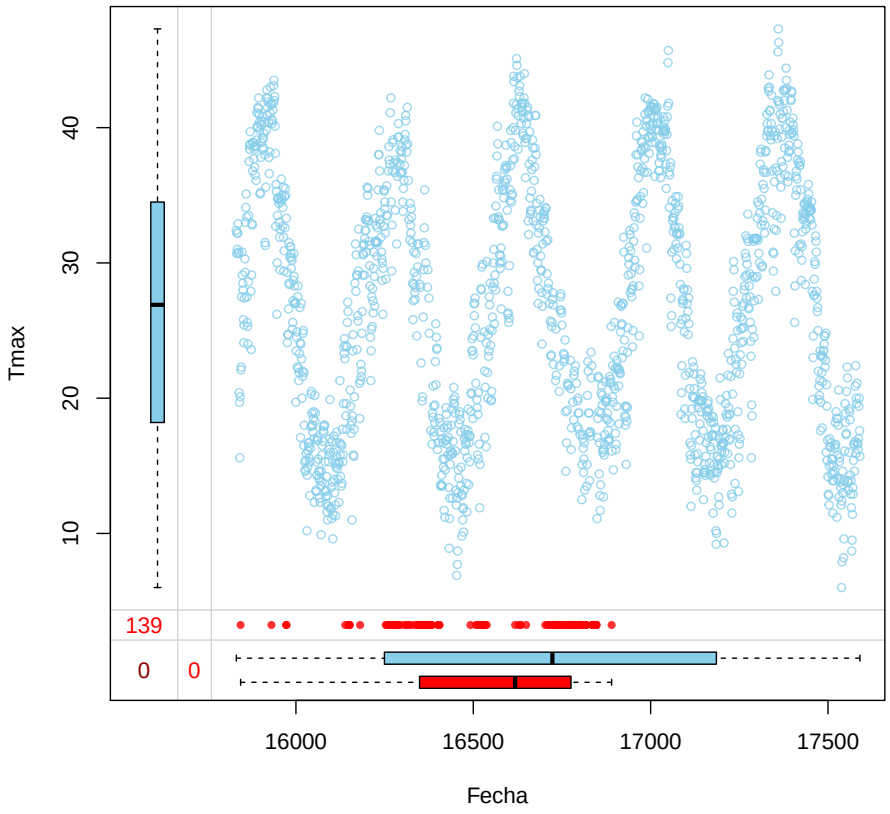

Figura 2: Representación gráfica de la nube de puntos de la serie de las temperaturas máximas frente al tiempo, donde se dibujan en color rojo los valores que se corresponden con valores pérdidos.

Observamos que, **la mayor parte de los valores perdidos se hayan situados en regiones concentradas próximas al instante central** de la serie.

Contando con esta información preliminar, ejecutamos un **preprocesamiento** con el cual se tratará de imputar los valores perdidos de la serie y a obtener

de la misma, la serie mensual y diaria que se estudiarán y se usarán para elaborar los modelos predictivos.

3.1. Preprocesamiento

A continuación trataremos algunos preprocesamientos previos sobre la serie para preparar los datos para su análisis y la elaboración de modelos:

3.1.1. Imputación de valores perdidos

Como se analizó en el análisis preliminar, los registros de las temperaturas máximas contienen 139 valores perdidos (7.924743 % de los datos), por lo que se necesitan **imputar** estos valores perdidos.

Dado la semejanza de la estacionalidad de la serie a una función conocida (función senoidal), se decide probar a utilizar métodos de imputación de valores perdidos basados en interpolación [1]

Probamos con interpolación lineal de los valores perdidos, de forma que la serie es reconstruída de la siguiente forma:

[1] Todos estos métodos se hayan implementados para R en la librería `imputeTS`

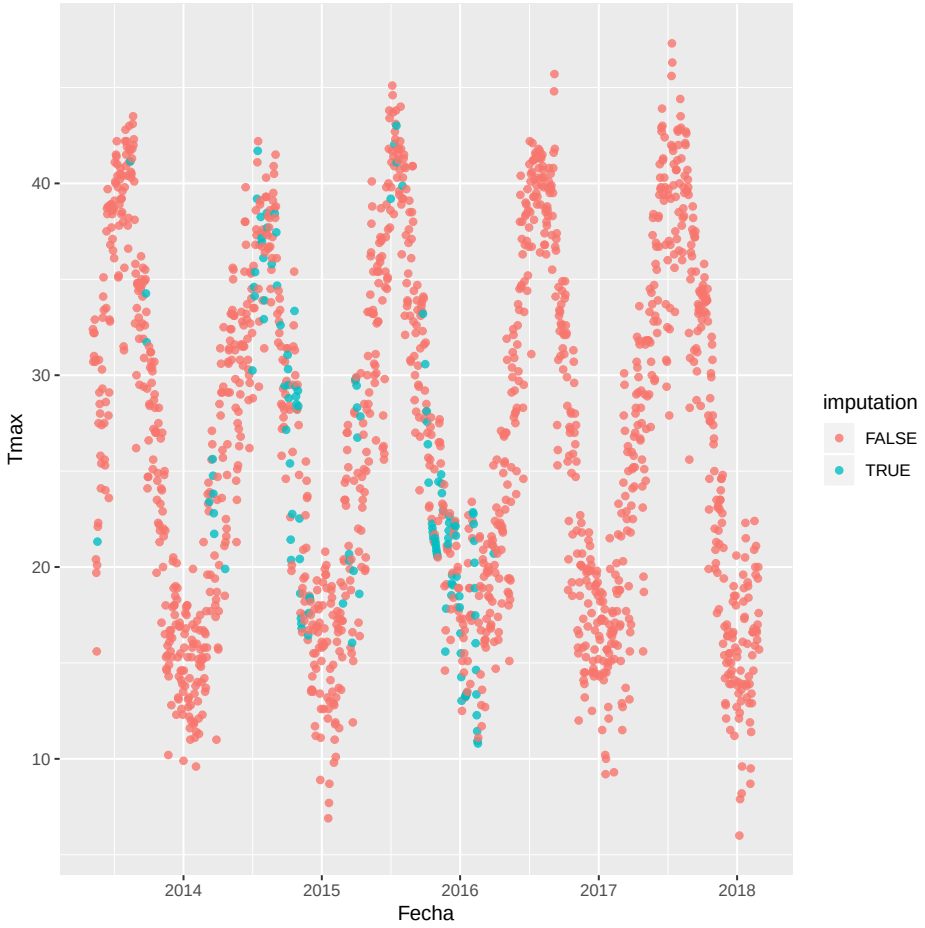

Figura 3: Representación gráfica de la nube de puntos de la serie de las temperaturas máximas frente al tiempo, en la que los valores perdidos se imputan por interpolación lineal. Se dibuja en color azul los valores imputados y en rojo los valores reales.

Observamos en este caso, que los valores imputados se ajustan con mucha precisión a la forma de la nube de puntos conocidas, lo cual nos lleva a considerar que este método de interpolación no introduciría mucho ruido a la serie.

Probando con una interpolación por "*splines*", los datos perdidos son reconstruídos de la siguiente forma:

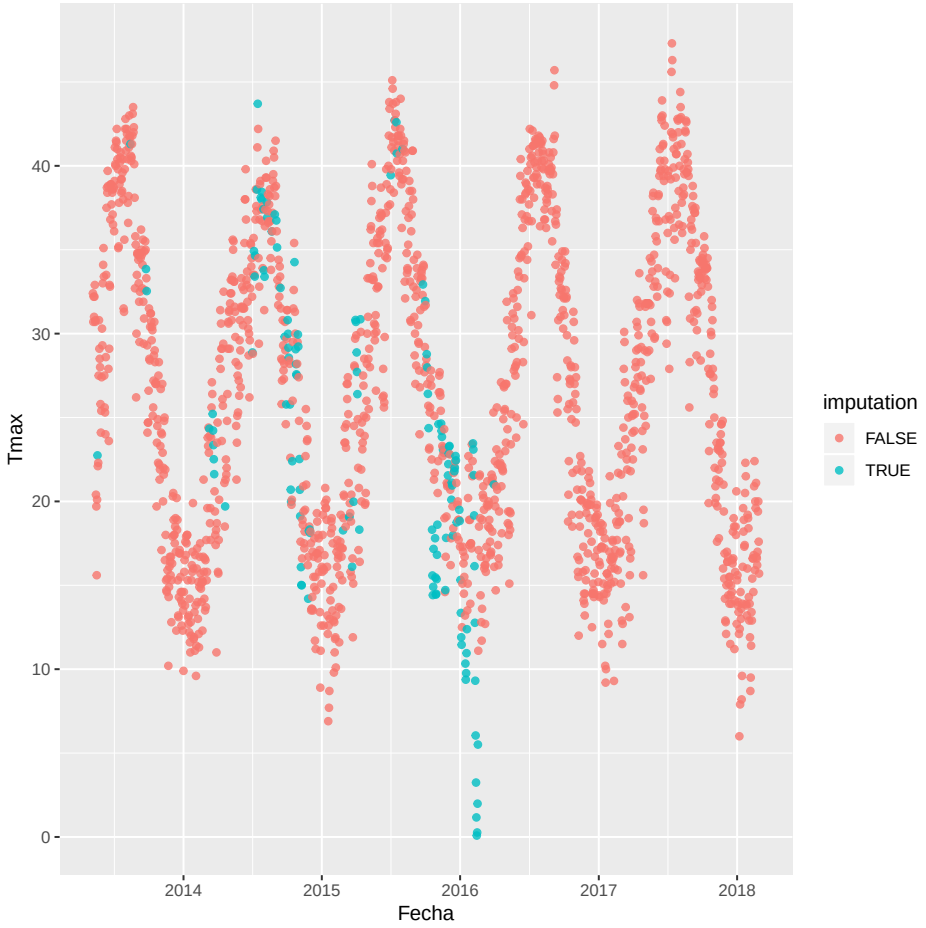

Figura 4: Representación gráfica de la nube de puntos de la serie de las temperaturas máximas frente al tiempo, en la que los valores perdidos se imputan por interpolación por "splines". Se ha dibujado en color azul los valores imputados y en rojo los valores originales.

En este caso, los valores imputados se ajustan también en gran medida a la forma de la distribución conocida, sin embargo, la reconstrucción del año 2016, difiere en gran medida de la forma de distribución conocida, lo cual lleva a desconfiar de que esta reconstrucción realizada exacta y lleve a la introducción de ruido.

Por último, se prueba también a interpolar con el método de "*Stineman*".

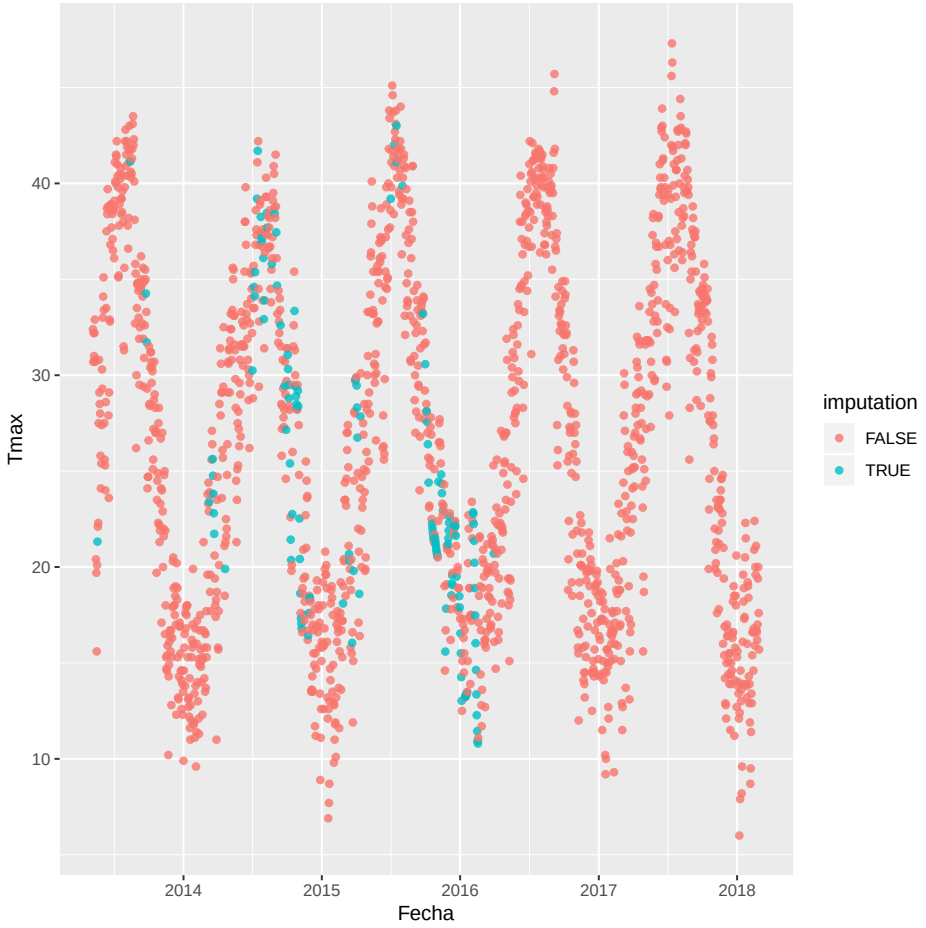

Figura 5: Representación gráfica de la nube de puntos de la serie de las temperaturas máximas frente al tiempo, en la que los valores perdidos se imputan por interpolación por el método de Stineman. Se dibujan en color azul los valores imputados mientras que los valores originales en rojo.

Apreciamos que esta reconstrución es similar a la reconstrución con interpolación lineal.

Por mayor garantía en la ausencia de ruido en los datos, se **decide trabajar con la serie reconstruída mediante interpolación lineal**.

3.1.2. Obtención de la serie diaria

Para la obtención de la serie diaria, dado que los datos recopilados en el fichero `5361X` recogen datos diarios, se construirá la serie a partir del tiempo

de registro (columna `Fecha`) y los registros de temperaturas máximas (columna `Tmax`), llevando a cabo una selección de las características columna `Fecha` y `Tmax` del *dataset* del fichero `5361X`.

La serie diaria se construye por tanto, de la selección de las columnas (`Fecha` y `Tmax`) del fichero del *dataset* original.

3.1.3. Obtención de la serie mensual

Trataremos ahora, la obtención de una serie mensual de las temperaturas máximas, con registros referidos a cada mes de la serie original en lugar de la serie actual con registros diarios.

Para lograr esta serie, usaremos un estadístico que permita resumir en un único valor las temperaturas máximas registradas durante todos los días de un mes.

Por la naturaleza de este problema, se considera más oportuno usar el valor máximo como estadístico resumen y generar una nueva serie con datos de temperatura mensuales donde la temperatura de cada mes es el valor máximo de las temperaturas de los días de dicho mes.

```
1 # Quedarnos con la temperatura más alta de cada mes
2 met.month <- met %>% mutate(Mes=format(Fecha, '%m'), Ano=format(Fecha,
    '%Y')) %>%
3   group_by(Ano, Mes) %>% summarise(Tmax=max(Tmax)) %>% ungroup() %>%
4   mutate(Mes=as.integer(Ano)*12+as.integer(Mes)) %>% select(Mes, Tmax)
     %>%
5   mutate(Mes=Mes-min(Mes)+1)
```

Script 2: Sentencias destinadas a obtener la serie temporal mensual de tempraturas máximas a partir de la serie original

3.2. Predicción de las temperaturas máximas en los meses de Marzo y Abril de 2018

Primeramente, haremos uso de la **serie temporal mensual** para elaborar, a partir, de la misma, modelos predictivos ARIMA no estacionales que permitan estimar la evolución de las temperaturas máximas durante los meses de Marzo y Abril de 2018.

La serie temporal mensual obtenida a partir del preprocesamiento es la siguiente:

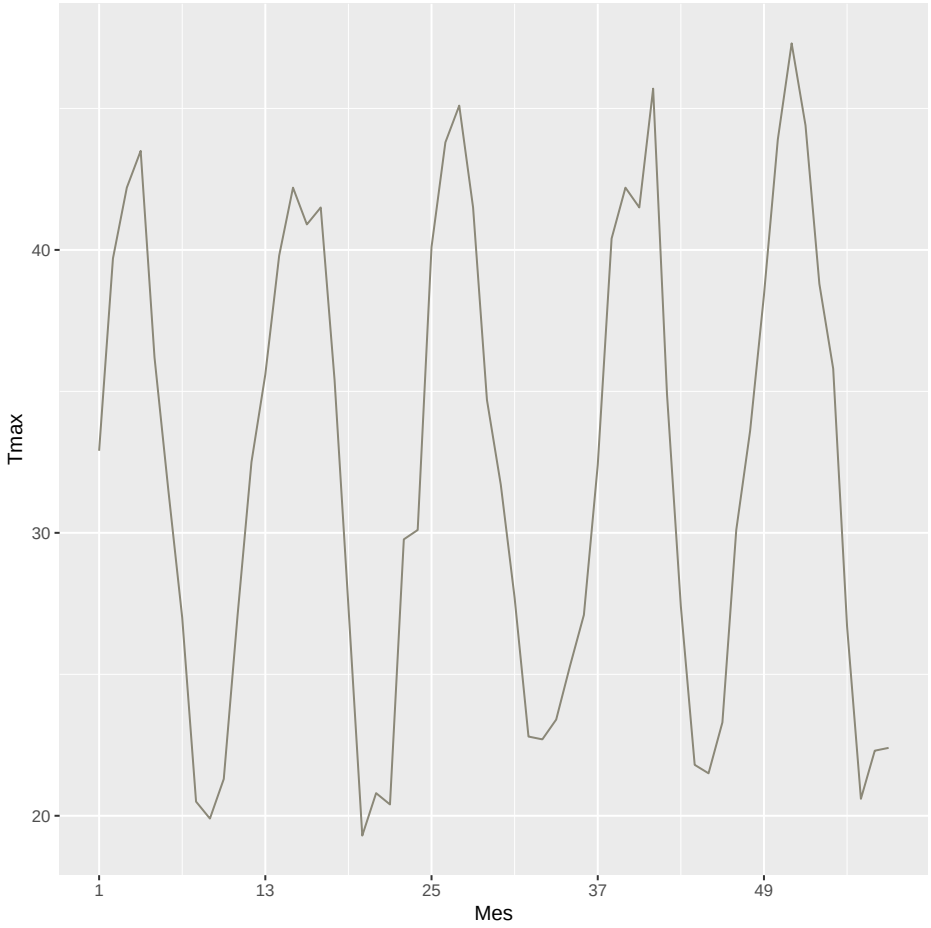

Figura 6: Representación gráfica de la serie mensual de las temperaturas máximas a lo largo de los meses de Mayo de 2013 y Febrero de 2018 (meses 24160 y 24218).

Para evaluar los modelos predictivos que se elaboren, se reservará el 10 % de los valores finales de la serie para validación dichos modelos (serie de test), usando el restante 90 % para elaborar los modelos (conjunto de *train*):

```
# Dividir el conjunto de datos en train y test,
# (se reserva el 90 % para train y 10 % para test)
met.train <- met.serie[1:as.integer(0.9*length(met.serie))]
met.train.time <- met$Mes[1:as.integer(0.9*length(met$Mes))]
met.train.serie <- ts(met.train, start = c(2013, 5),
                      deltat = 1/12, frequency = 12)
```

Script 3: Sentencias a dividir la serie en un conjunto para train y otro conjunto de test

Centrándose en el análisis, dado que la serie no presenta variaciones en

las magnitudes de las fluctuaciones estacionales, se tratará una **descomposición de tipo aditiva**. Por su parte, la serie no muestra tendencia (no se aprecian incrementos o decrementos en la temperatura máxima a largo plazo).

Sí **se aprecia una estacionalidad anual que se repite cada 12 meses** y que, a simple vista, **se aproxima a una función senoidal**. Por consiguiente, se considerará una frecuencia de 12 meses para esta serie.

Comparamos estas consideraciones con la descomposición aditiva basada en medias móviles implementada por defecto en el método `decompose` en R :

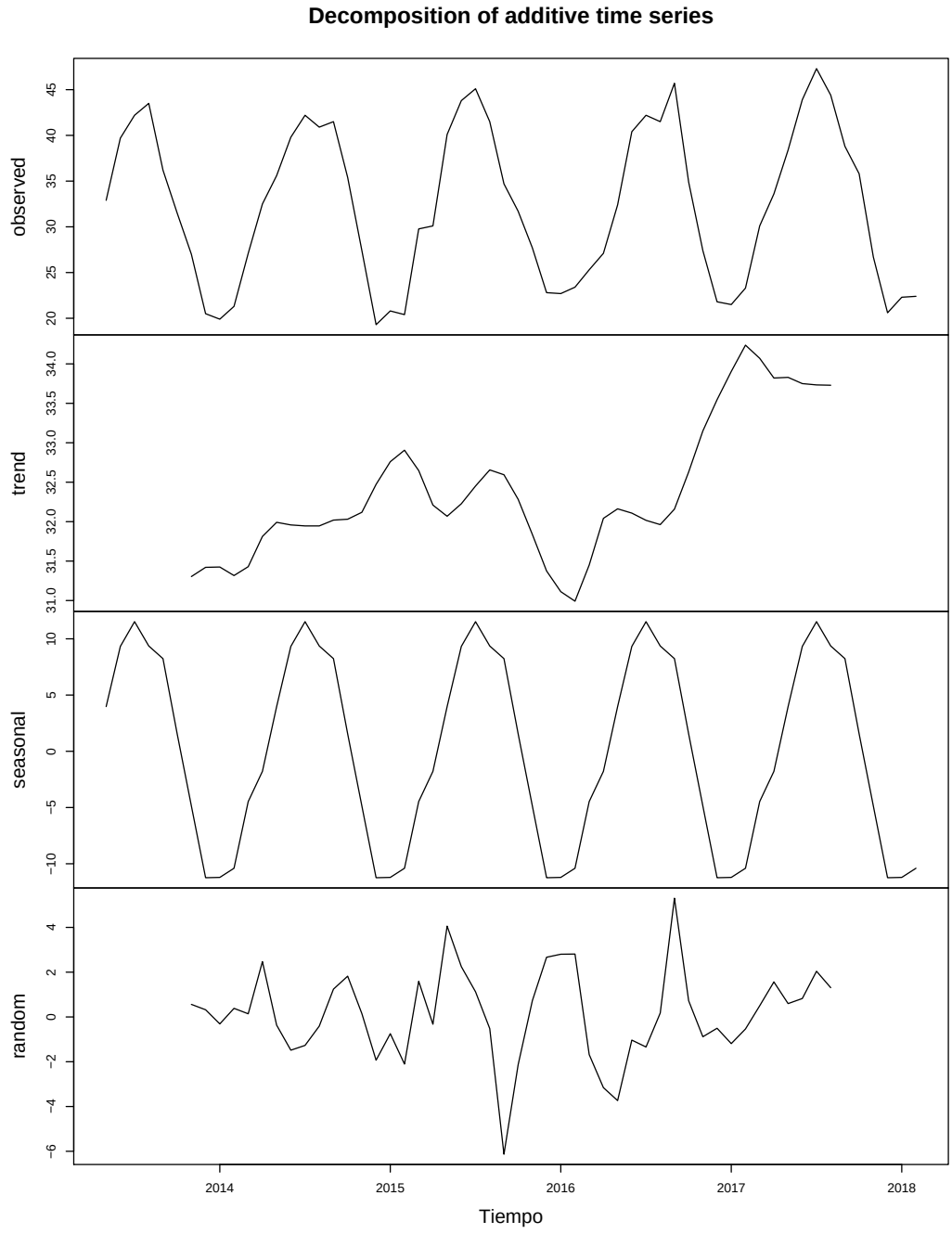

Figura 7: Representación gráfica de las serie original junto con las componentes en las que ha sido descompuestas por el método decompose.

La tendencia extraída por este método mediante medias móviles presenta

una forma irregular que no se ajusta a ninguna función conocida, lo que lleva a apoyar la ausencia de tendencia en la serie.

Respecto a la estacionalidad, tal y como se indicó anteriormente, la serie presenta una estacionalidad anual que se puede apreciar con claridad en la descomposición.

Procedemos a estimar esta estacionalidad mediante el simple cómputo de medias para cada uno de los instantes de tiempo en cada periodo y se la restamos a la serie, obteniéndo así el error aleatorio:

```
# Estimar estacionalidad y eliminarla
met.train.matrix <- matrix(data = met.train[1:((length(met.train)%/%12)
    *12)],
                           ncol = 12, byrow = TRUE)
estac.met.train <- apply(met.train.matrix, FUN=mean, MARGIN = 2)

rep.estac.met.train <- c(rep(estac.met.train,
                           length(met.serie)/length(estac.met.train)),
                         estac.met.train[1:(length(met.serie)%%length(
    estac.met.train))])
```

Script 4: Sentencias para la estimación de la estacionalidad de la serie y la eliminación de la misma para la obtencion del error irreductible

La serie sin estacionalidad queda del siguiente modo:

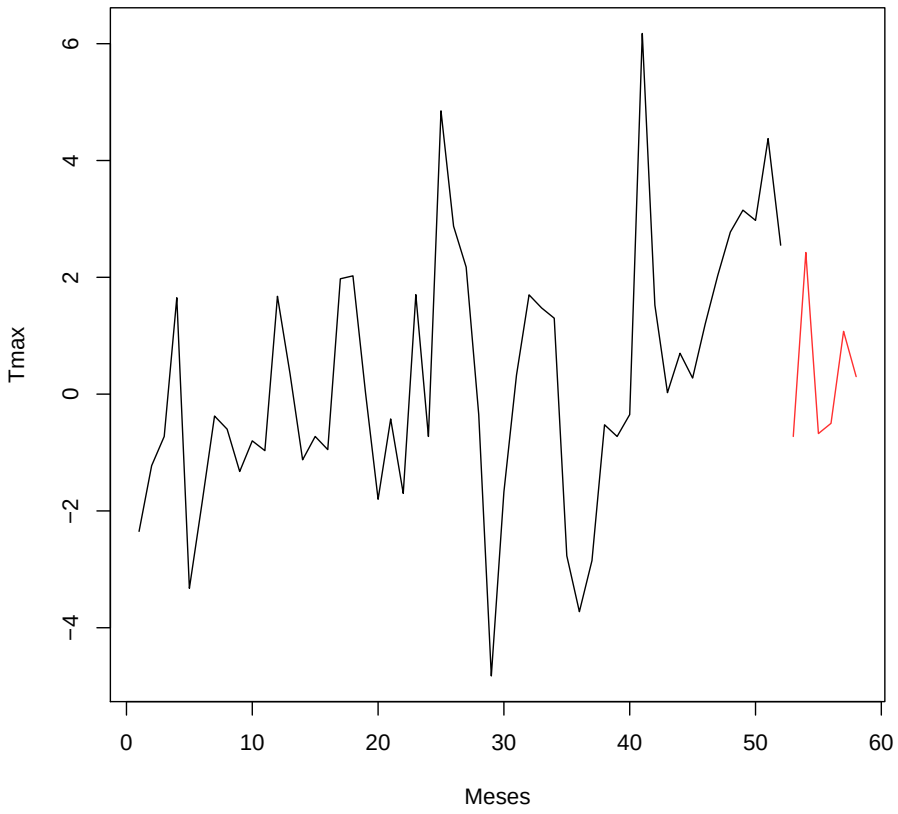

Figura 8: Representación gráfica de las serie sin estacionalidad para el conjunto de train (color negro) y el conjunto de test (color azul).

Comprobamos ahora que la serie sea **estacionaria**, es decir, que su media no varíe a lo largo del tiempo, lo cual es necesario para garantizar la independencia de la serie respecto del momento en que fue captada.

Para comprobar la **estacionaridad** de la serie, aplicamos el **test de *Dickie-Furer* aumentado** (test ADF):

```
# Test ADF para comprobar que la serie es estacionaria
adf.test(met.train.no_estac)
```

Script 5: Sentencias para la aplicación del test de Dickie-Fuller sobre los residuos del conjunto de train

```
Augmented Dickey-Fuller Test
```

```
data:  met.train.no_estac
Dickey-Fuller = -4.1286, Lag order = 3, p-value = 0.01082
alternative hypothesis: stationary
```

Con una confianza del 95 % el *p-value* del test nos lleva a rechazar la hipótesis nula y a considerar que **la serie es estacionaria**.

Corroboramos este hecho visualizando el **gráfico de autocorrelación (*ACF*)**:

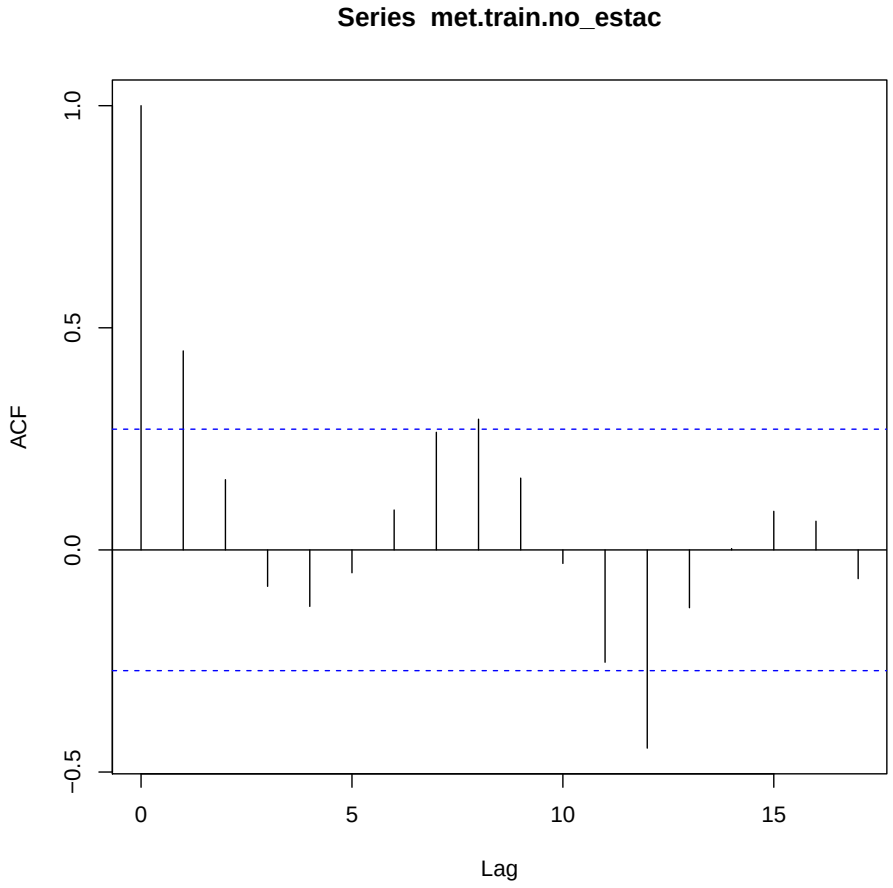

Figura 9: Gráfico ACF de la serie sobre el conjunto de train

El gráfico de autocorrelación, nos muestra un descenso rápido de la autocorrelación a 0 en pocos *lags*, lo cual nos lleva a confirmar la estacionaridad de la serie.

Analizamos también el gráfico de las **autocorrelaciones parciales** (test PACF):

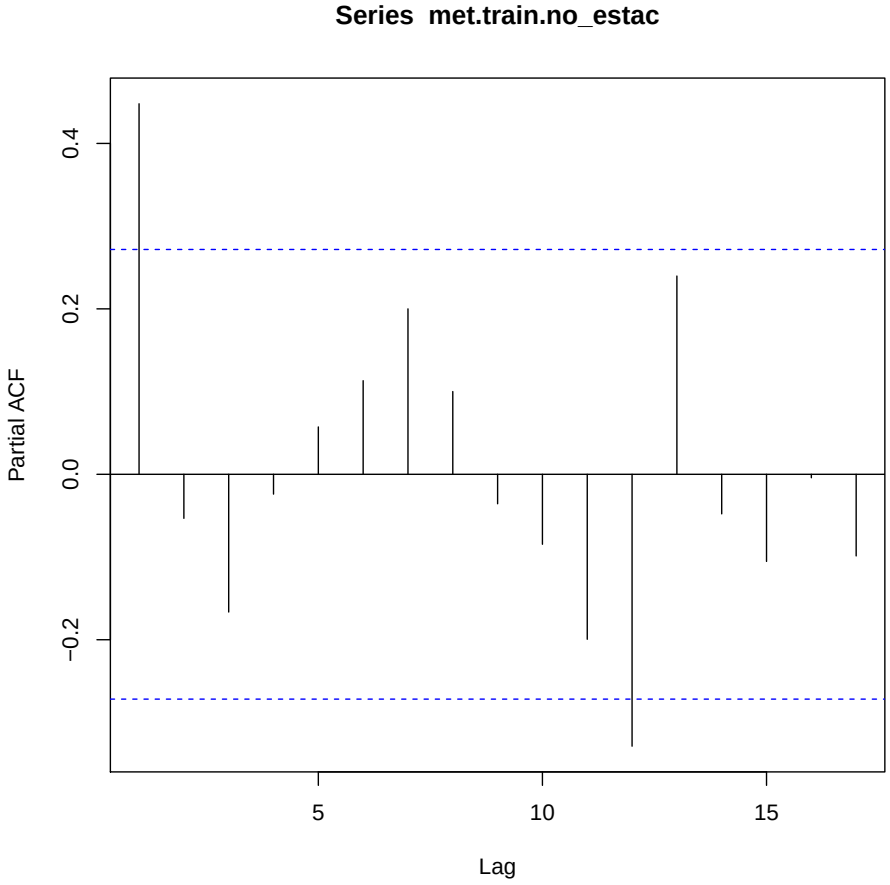

Figura 10: Gráfico PACF de la serie sobre el conjunto de train

En el gráfico PACF, también apreciamos un descenso inmediato a la autocorrelación 0, en el *lag* siguiente al instante actual, en el que la autocorrelación parcial adopta un valor negativo.

Con el residuo de la serie del conjunto de entrenamiento, procedemos a elaborar ya el modelo predictivo **ARIMA** (*AutoRegressive Integrated Moving Average*) no estacional (la estacionalidad ya ha sido estimada y eliminada) para llevar a cabo la predición.

En el gráfico ACF de la figura 9 observamos un comportamiento sinuidal

de en las autocorrelaciones conforme se incrementan los *lags*, además de la mayor autocorrelación en el *lag* inmediatamente posterior al instante actual (autocorrelación próxima a 0.5), lo cual llevaría a plantear como modelo más idóneo un modelo ARIMA(1,d,0).

Dado que no se ha realizado ninguna diferenciación, tomamos d=0 y **el modelo resultaría ARIMA(1,0,0)**, es decir, un modelo autorregresivo de orden 1:

```
# Probar modelo ARIMA(1,0,0)
model.arima <- arima(met.train.no_estac, order = c(1,0,0))
```

Script 6: Sentencias para la ejecución de un modelo ARIMA(1,0,0)

Para garantizar la validez del modelo, comprobamos que los residuos del mismo son aleatorios. Nuevamente hacemos uso del test de *Box-Pierce*, el test de *Shapiro-Wilk* y el test de *Jarque-Bera*:

```
# Comprobar que los residuos son normales
Box.test(model.arima$residuals)
shapiro.test(model.arima$residuals)
jarque.bera.test(model.arima$residuals)
```

Script 7: Sentencias para la ejecución de los test de Box-Pierce, Shapiro-Wilk y Jarque-Bera

```
Box-Pierce test

data:  model.arima$residuals
X-squared = 0.0081953, df = 1, p-value = 0.9279

Shapiro-Wilk normality test

data:  model.arima$residuals
W = 0.96177, p-value = 0.09337

Jarque Bera Test

data:  model.arima$residuals
X-squared = 7.2268, df = 2, p-value = 0.02696
```

Con una confianza del 95 %, el test de *Box-Pierce* y el test de *Shapiro-Wilk* nos lleva a afirmar la aleatoriedad de los errores residuales del modelo ARIMA. No sucede del mismo modo con el test de *Jarque-Bera*, en el cual el *p-value* obtenido nos lleva a rechazar la hipótesis nula y a considerar que la asimetría y la curtosis de la distribución de los residuos no se ajustan a los

de la distribución normal.

Analizamos más detalladamente la distribución de los errores, visualizandola en un histograma:

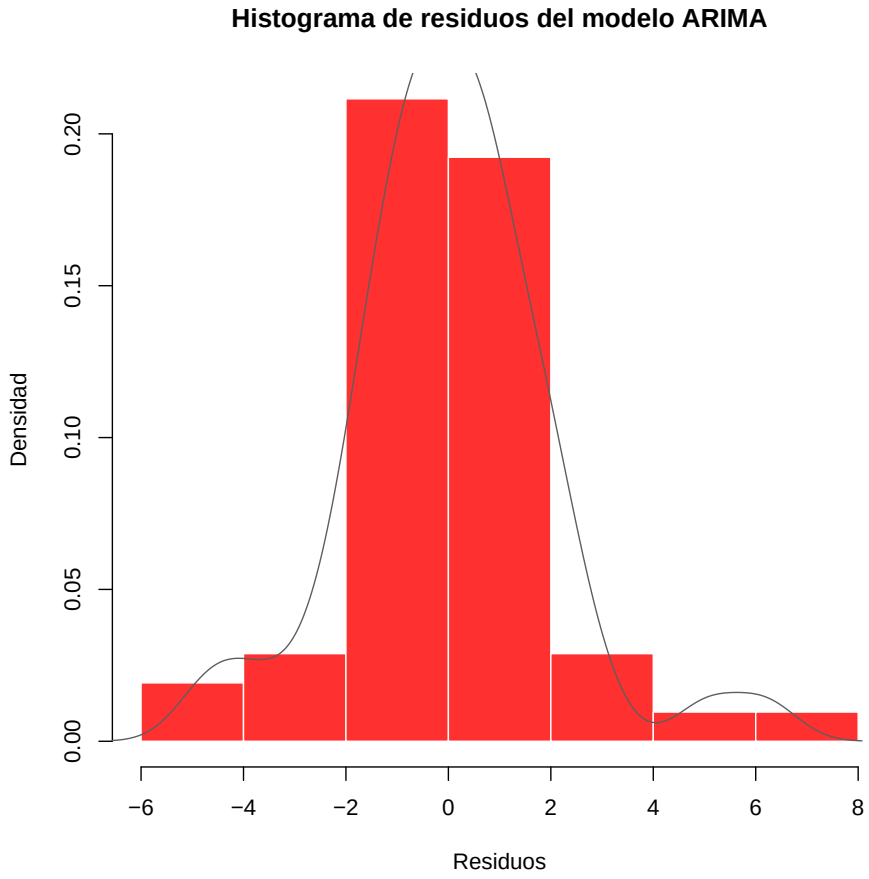

Figura 11: Gráfico de la distribución de los errores del modelo ARIMA(1,0,0)

Gráficamente, observamos que la distribución de los errores se asemeja a una distribución Normal. Sin embargo, se observa una mayor densidad en la zona izquierda de la distribución que en la zona derecha, lo cual provocaría el fallo del test de *Jarque-Bera*.

Hacemos uso de nuestro modelo para predecir la evolución de la temperaturas máximas del conjunto de test y evaluamos su rendimiento tanto en el conjunto de *test* como en el conjunto de *train* mediante el **la suma de los errores cuadráticos** (*Sum of Squared Errors* o *SSE*):

```
# Predecir los conjuntos de train y test y evaluar el modelo
train.pred <- met.train.no_estac - model.arima$residuals # Train

arima.pred.test <- predict(model.arima, n.ahead = length(met.test.no_
    estac))
test.pred <- arima.pred.test$pred # Test

# Calcular los errores sobre el conjunto de train y test
sse.arima.train <- sum((model.arima$residuals)^2)
sse.arima.test <- sum((test.pred-met.test.no_estac)^2)

cat('Error SSE cometido sobre el conjunto de train: ', sse.arima.train,
     fill=T)
cat('Error SSE cometido sobre el conjunto de test: ', sse.arima.test,
    fill=T)
```

Script 8: Sentencias para la predición de los datos sobre los conjuntos de train y test y el cálculo de los errores SSE cometidos

Conjunto	SSE
Train	195.4268
Test	9.639616

Cuadro 2: Errores SSE cometidos por el modelo ARIMA (1,0,0) sobre el conjunto de train y test

Visualizamos gráficamente la predición realizada junto con la serie real:

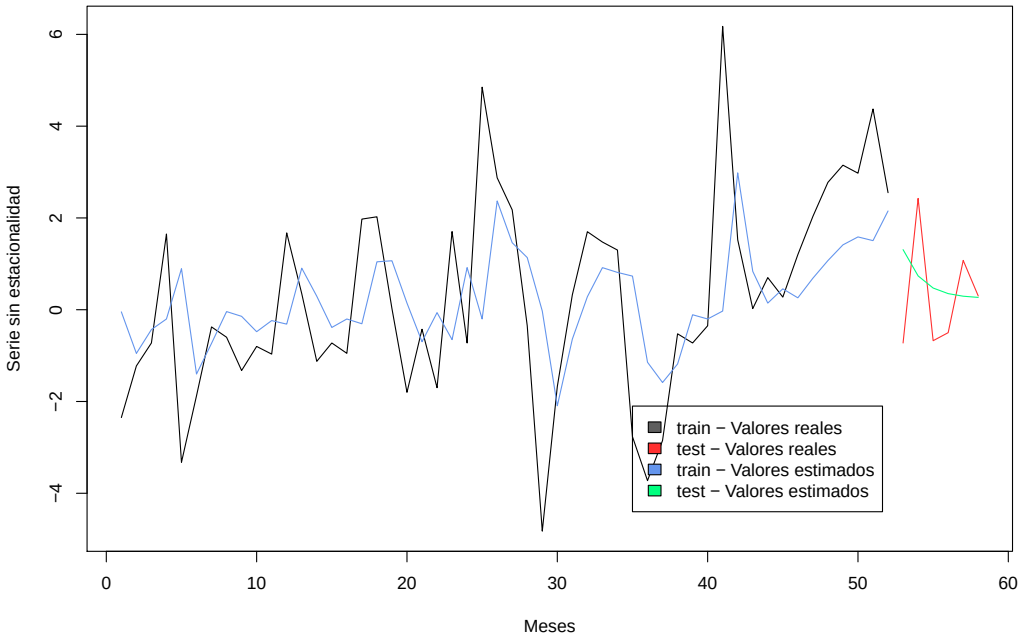

Figura 12: Gráfico de la serie sin estacionalidad dividida en conjunto de train (color negro) y conjunto de test (color rojo) sobre la que se ha dibujado la predición del modelo ARIMA (1,0,0) sobre el conjunto de train (color azul) y sobre el conjunto de test (color verde)

Añadiendo la componente de estacionalidad eliinada anteriormente tanto a la serie original como a la predición resulta:

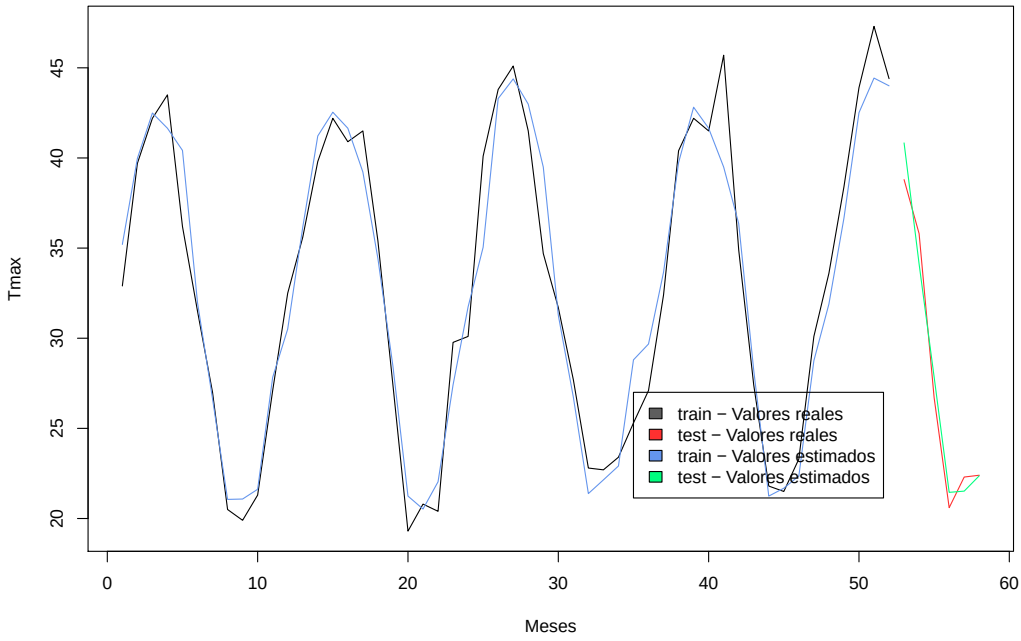

Figura 13: Gráfico de la serie dividida en conjunto de train (color negro) y conjunto de test (color rojo) sobre la que se ha dibujado la predición del modelo ARIMA (1,0,0) sobre el conjunto de train (color azul) y sobre el conjunto de test (color verde)

En este gráfico podemos apreciar con claridad como la predición realizada por el modelo ARIMA (1,0,0) tanto sobre el conjunto de *train* como de test se aproxima a los valores reales de la serie con mucha precisión.

Por consiguiente, aceptamos que este modelo es válido y nos permitiría predecir la evolución de las temperaturas máximas en los meses de Marzo y Abril de 2018.

Para la predición de las temperaturas máximas en los meses de Marzo y Abril de 2018, ajustamos un modelo ARIMA de la misma configuración que el anterior, esta vez, usando todos los datos de la serie mensual de la siguiente forma:

Se determina la estacionalidad de toda la serie mensual y se elimina de la misma para obtener el residuo aleatorio. A continuación, se ajusta el modelo ARIMA con la configuración (1,0,0) y se predice con el mismo las tempera-

turas máximas en los meses de Marzo y Abril de 2018 (meses 59 y 60).

```
### Predecir toda la serie con el modelo ARIMA

# Eliminar estacionalidad de la serie
met.matrix <- matrix(data = met$Tmax[1:((length(met$Tmax)%/%12)*12)],
                     ncol = 12, byrow = TRUE)
estac.met <- apply(met.matrix, FUN=mean, MARGIN = 2)

rep.estac.met <- c(rep(estac.met,
                       length(met.serie)/length(estac.met)),
                       estac.met[1:(length(met.serie)%%length(estac.
    met))])

met.no_estac <- met$Tmax - rep.estac.met

# Ajustar modelo ARIMA (1,0,0)
model.arima.full <- arima(met.no_estac, order = c(1,0,0))
model.arima.full

# Predecir los meses 59 y 60 (los dos meses siguientes)
pred.arima <- predict(model.arima.full, n.ahead = 2)
pred <- pred.arima$pred

# Mostrar el error residual
cat('Error SSE cometido sobre los datos de entrenamiento: ',
    sum(pred.arima$se**2), fill = T)

# Añadir estacionalidad para el tramo predicho
pred.estac.met <- estac.met[(((59:60)-met$Mes[1])%%12)+1]
pred.estac <- pred + pred.estac.met

# Mostrar predición realizada
print(pred.estac)

# Mostrar gráfica original junto con la predición
pdf('Tmax_predicion_marzo_abril.pdf', width = 10)
plot(x = met$Mes, y = met$Tmax,
     xlim=c(met$Mes[1], met$Mes[length(met$Mes)]+2),
     xlab='Año', ylab='Tmax', type = "l", col = "gray37", xaxt="n")
axis(1, at = (2013:2018)*12, labels = 2013:2018)
lines((met$Mes[length(met$Mes)]):(met$Mes[length(met$Mes)]+1),
      c(met$Tmax[length(met$Tmax)], pred.estac[1]), col='firebrick1')
lines((met$Mes[length(met$Mes)]+1):(met$Mes[length(met$Mes)]+2),
      pred.estac, col='firebrick1')
legend(x=4, y=24, legend = c('Serie original', 'Predición'),
       fill = c('gray37', 'firebrick1'))
dev.off()
```

Script 9: Sentencias para el ajuste y evaluación de un modelo ARIMA (1,0,0) no estacionario con todos los datos de la serie y la predición con el mismo de la evolución de la temperaturas máximas en los meses de Marzo y Abril de 2018

El modelo ajustado comete un error SSE de 7.912342 sobre los datos de la serie usados en el entrenamiento y la predición del mismo sobre las temperaturas máximas en los meses de Marzo y Abril de 2018 es la siguiente:

Mes	Año	Temperatura máxima (ºC) predicha
Marzo	2018	28.32496
Abril	2018	31.06473

Cuadro 3: Predición de la temperatura máxima para los meses de Marzo y Abril por el modelo ARIMA (1,0,0) no estacionario ajustado con la serie mensual comprendida entre Mayo de 2013 y Febrero de 2018

En la siguiente gráfica, se muestra la predición realizada junto a la serie original:

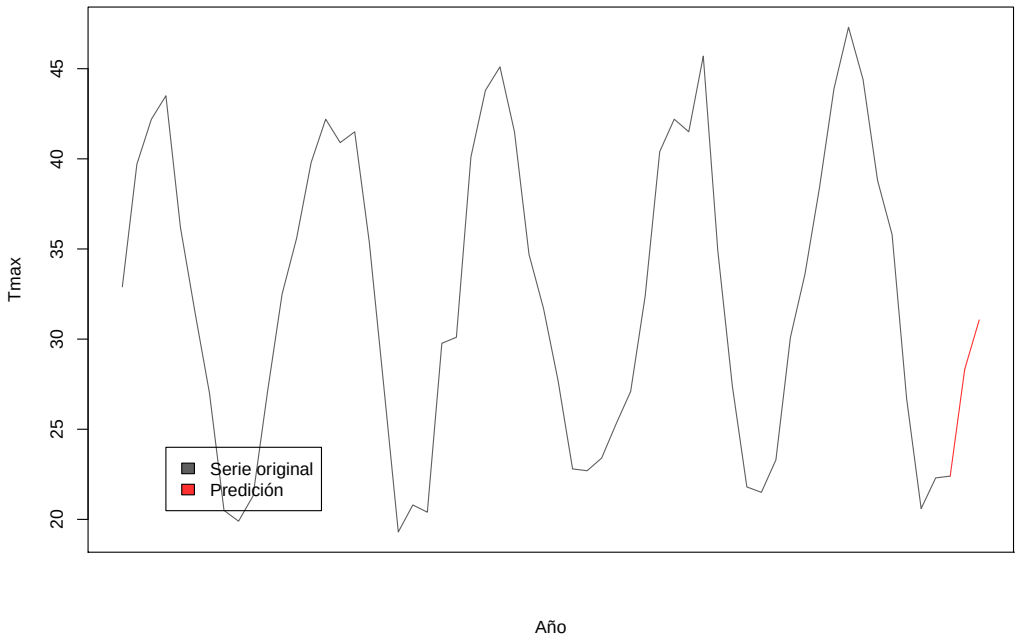

Figura 14: Gráfico de la serie mensual original (color negro) junto con la predición realizada en los meses de Marzo y Abril de 2018 por el modelo ARIMA(1,0,0) no estacional

3.3. Predicción de las temperaturas máximas en la primera semana de Marzo

A continuación, trataremos la elaboración de otro modelo predictivo, para estimar la evolución de las temperaturas máximas durante durante la primera semana de Marzo de 2018, para lo que en esta ocasión, haremos uso de la

serie temporal diaria.

La serie temporal diaria obtenida a partir del preprocesamiento es la siguiente:

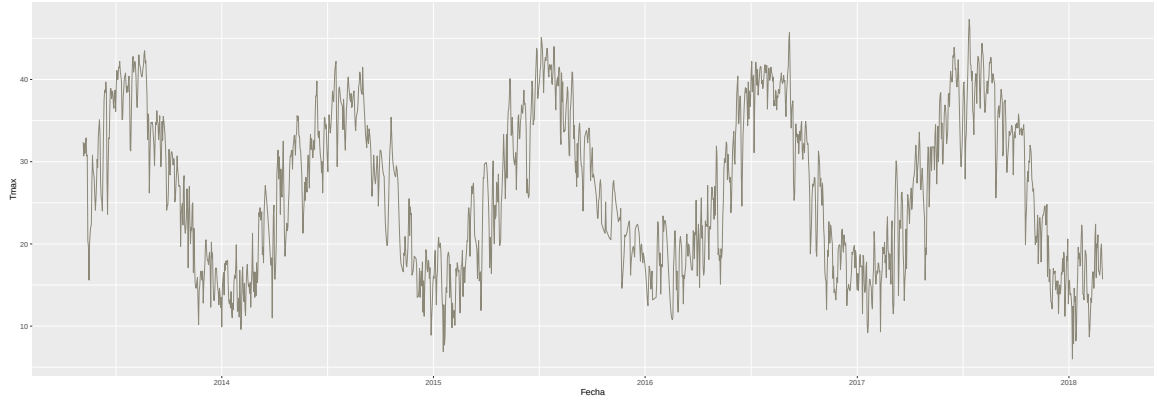

Figura 15: Representación gráfica de la serie de las temperaturas máximas diarias a lo largo de los días entre los meses de Mayo de 2013 y Febrero de 2018.

Al igual que con la anterior serie, dividimos la serie diaria en dos subconjuntos de *train* y *test*, reservando nuevamente, el 90 % de los instantes iniciales de la serie para el subconjunto de *train* y el 10 % restante para el subconjunto de *test*:

```
# Dividir el conjunto de datos en train y test,
# (se reserva el 90 % para train y 10 % para test)
met.train <- met$Tmax[1:as.integer(0.9*length(met$Tmax))]
met.train.time <- met$Fecha[1:as.integer(0.9*length(met$Fecha))]

met.test <- met$Tmax[as.integer(0.9*length(met$Tmax)+1):length(met$Tmax
    )]
met.test.time <- met$Fecha[as.integer(0.9*length(met$Fecha)+1):length(
    met$Fecha)]
```

Script 10: Sentencias para la división de la serie en un subconjunto de train y otro subconjunto de test

A simple vista, al igual que con la anterior serie, la serie tampoco presenta variaciones en las fluctuaciones de las magnitudes, por lo que se decide aplicar también **descomposición aditiva**.

La serie tampoco muestra tendencia (no se aprecian incrementos o decrementos en los valores de las temperaturas a largo plazo).

Respecto a la estacionalidad, **se percibe una estacionalidad anual** al igual que con la anterior serie. Observando con mayor detalle los días de un mes, se observan oscilaciones en los valores de temperaturas máximas, pero debido a la estacionalidad anual de la serie, resulta difícil distinguir a simple vista patrones estacionarios en esta escala temporal, por lo que primeramente, se necesita modelar y eliminar esta estacionalidad anual.

Analizamos primeramente la descomposición por defecto implementada por el método `decompose` considerando una frecuencia anual de la serie:

```
# Estudiar estacionalidad anual
met.anual <- ts(met[,2], frequency = 365)

# Descomponer la serie anualmente
desc.met.anual <- decompose(met.anual)caption
```

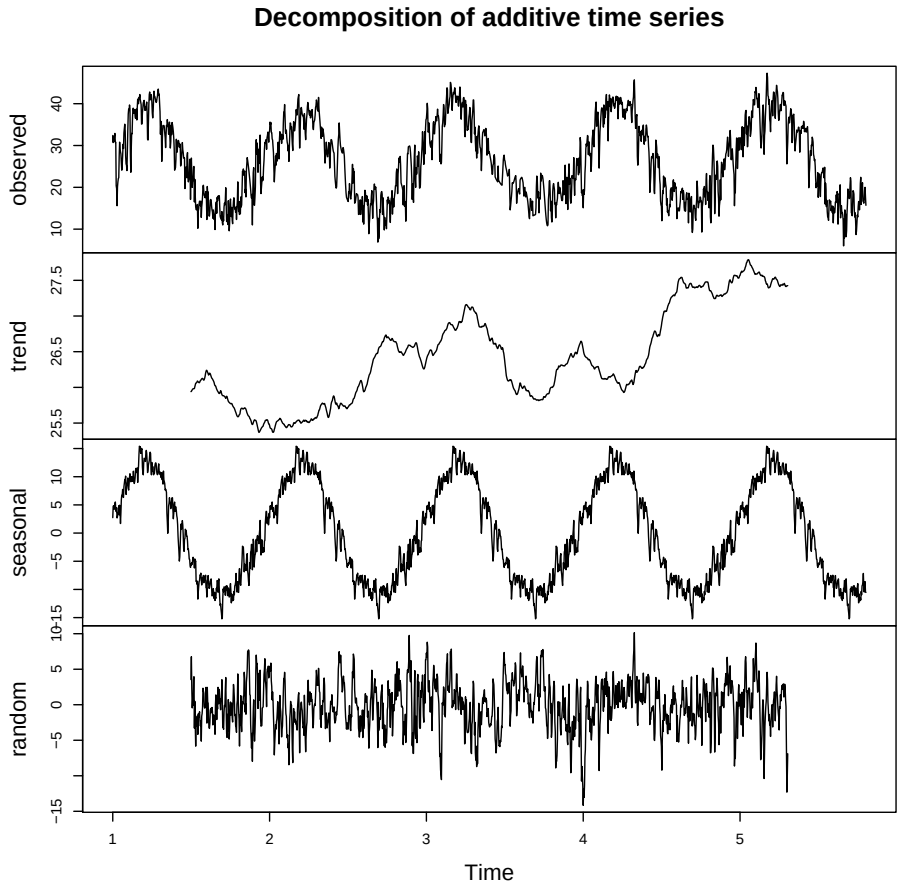

Figura 16: Serie diaria con periodo anual junto a sus componentes de tendencia, estacionalidad y error residual por el método decompose.

La tendencia calculada por el método mediante filtro de medias móviles, muestra nuevamente una forma irregular que no se ajusta a ninguna función conocida, por lo cual se considera que no es aprovechable.

Respecto a la estacionalidad, el método `decompose` permite identificar claramente la componente de estacionalidad anual de la serie y que deseamos estimar y eliminar.

Procedemos a estimar la estacionalidad anual de la serie y su eliminación de la serie:

```
# Estimar estacionalidad
met.train.matrix <- matrix(data = met.train[1:((length(met.train)%/%
    365)*365)],
```

```
                                ncol = 365, byrow = TRUE)
estac.met.train <- apply(met.train.matrix, FUN=mean, MARGIN = 2)

rep.estac.met.train <- c(rep(estac.met.train,
                             length(met$Tmax)%/%length(estac.met.train)
    ),
                         estac.met.train[1:(length(met$Tmax)%%length(
    estac.met.train))])

met.train.no_estac <- met.train - rep.estac.met.train[1:as.integer(0.9*
    length(rep.estac.met.train))]
met.test.no_estac <- met.test - rep.estac.met.train[as.integer(0.9*
    length(rep.estac.met.train)+1):length(rep.estac.met.train)]
```

Script 11: Sentencias para la estimación y eliminación de la estacionalidad anual

La serie resultante tras eliminar esta estacionalidad es la siguiente:

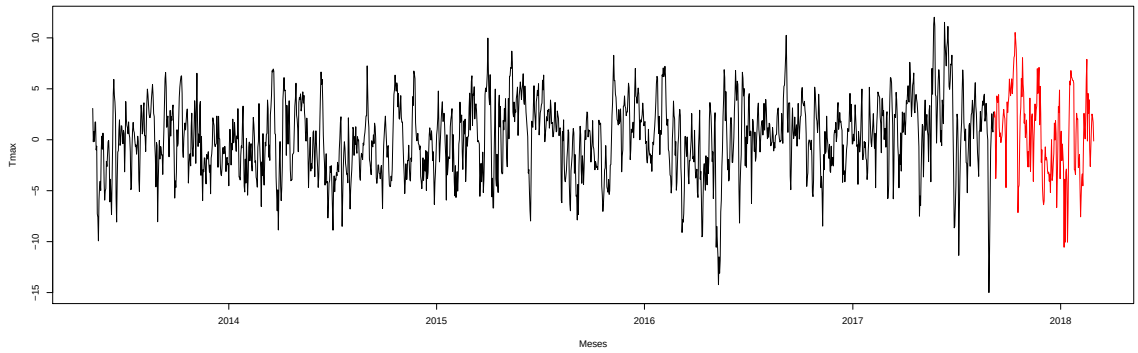

Figura 17: Serie diaria sin estacionalidad anual.

Esta serie resultante tampoco muestra tendencia y, a simple vista, sigue resultando difícil identificar más componentes estacionarias, por lo que representamos otra gráfica con el la evolución de las temperaturas máximas en el primer año:

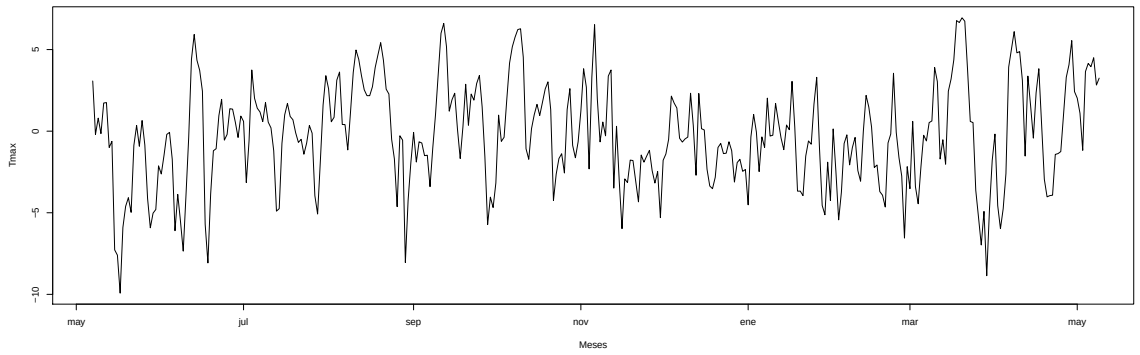

Figura 18: Serie diaria sin estacionalidad anual donde se muestra únicamente el primer año de la serie.

Acotando la representación a un año, tampoco se distinguen patrones estacionarios mensuales ni de menor orden, por lo que se considera que **la serie no presenta más componentes de estacionalidad**.

Procedemos a comprobar que la serie es estacionaria con el **test de *Dickie-Furier***:

```
# Test ADF para comprobar la estacionariedad de la serie
adf.test(met.train.no_estac)
```

Script 12: Sentencias para la aplicación del test ADF sobre los errores residuales de la serie sobre el conjunto de train y test

```
Augmented Dickey-Fuller Test

data:  met.train.no_estac
Dickey-Fuller = -9.7431, Lag order = 11, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(met.train.no_estac) : p-value smaller than printed p-value

Augmented Dickey-Fuller Test

data:  met.test.no_estac
Dickey-Fuller = -4.3644, Lag order = 5, p-value = 0.01
alternative hypothesis: stationary
```

```
Warning message:
In adf.test(met.test.no_estac) : p-value smaller than printed p-value
```

Considerando una confianza del 95 %, el *p-value* de los test realizados sobre los conjuntos de *train* y *test* nos lleva, en ambos casos, a rechazar la hipótesis nula del test y a afirmar que los residuos son estacionarios.

Analizamos las gráfica de autocorrelación (*ACF*) y autocorrelación parcial (*PACF*) para corroborar esta afirmación:

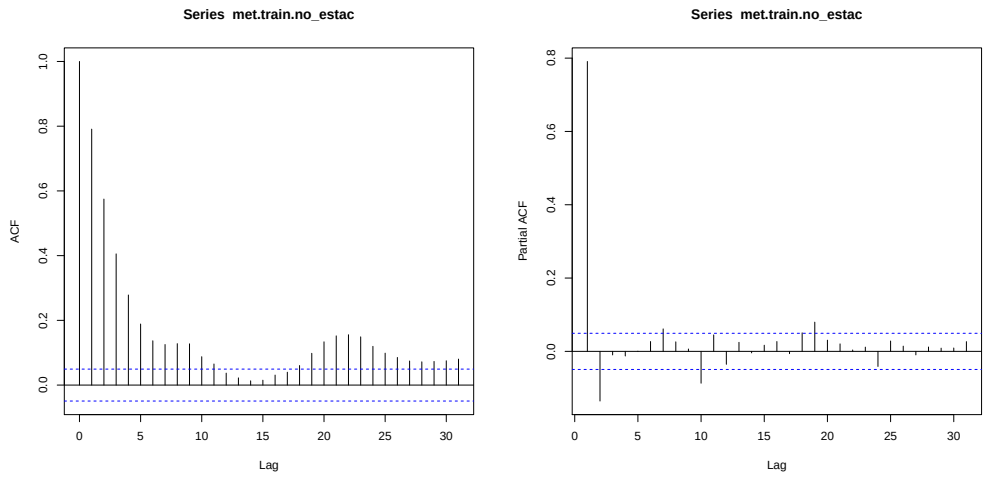

Figura 19: Gráficas ACF (a la izquierda) y PACF (a la derecha) de la serie con los errores residuales.

En el gráfico ACF podemos observar un descenso de las autocorrelaciones que se asemeja a un decaimiento exponencial conforme aumenta el número de *lags* hasta llegar a 0. En el gráfico PACF, tampoco se observa ningún indicio de estacionaridad, pues se observa que las autocorrelaciones parciales tienden rápidamente a 0.

Procedemos a ajustar el modelo ARIMA: Considerando la información mostrada en los anteriores gráficos ACF y PACF, dado que el gráfico ACF muestra un descenso que se asemeja a un descenso exponencial y, que tras el primer *lag*, que ofrece una autocorrelación de 0.8, no aparecen en el gráfico otros *lags* que ofrezcan una autocorrelación mayor o igual, se plantea un modelo ARIMA no estacional de parámetros (1,0,0):

```
# Elaborar modelo ARIMA(1,0,0)
model.arima <- arima(met.train.no_estac, order = c(1,0,0))
```

```
model.arima
```

Script 13: Sentencias para la elaboración de un modelo ARIMA(1,0,0) con la serie de entrenamiento no estacionaria

Comprobamos que los residuos del modelo se distribuyen normalmente, haciendo uso nuevamente de los test de *Box-Pierce*, *Shapiro-Wilk* y *Jarque-Bera*:

```
# Comprobar la normalidad de los residuos del modelo
Box.test(model.arima$residuals)
shapiro.test(model.arima$residuals)
jarque.bera.test(model.arima$residuals)caption
```

```
Box-Pierce test

data:  model.arima$residuals
X-squared = 18.475, df = 1, p-value = 1.721e-05

Shapiro-Wilk normality test

data:  model.arima$residuals
W = 0.9853, p-value = 1.335e-11

Jarque Bera Test

data:  model.arima$residuals
X-squared = 137.92, df = 2, p-value < 2.2e-16
```

Con una confianza del 95 %, todos los test fallan, nos llevan a rechazar la hipótesis nula y a afirmar que los residuos del modelo ARIMA no son aleatorios.

Visualizamos de forma gráfica la distribución de los residuos del modelo:

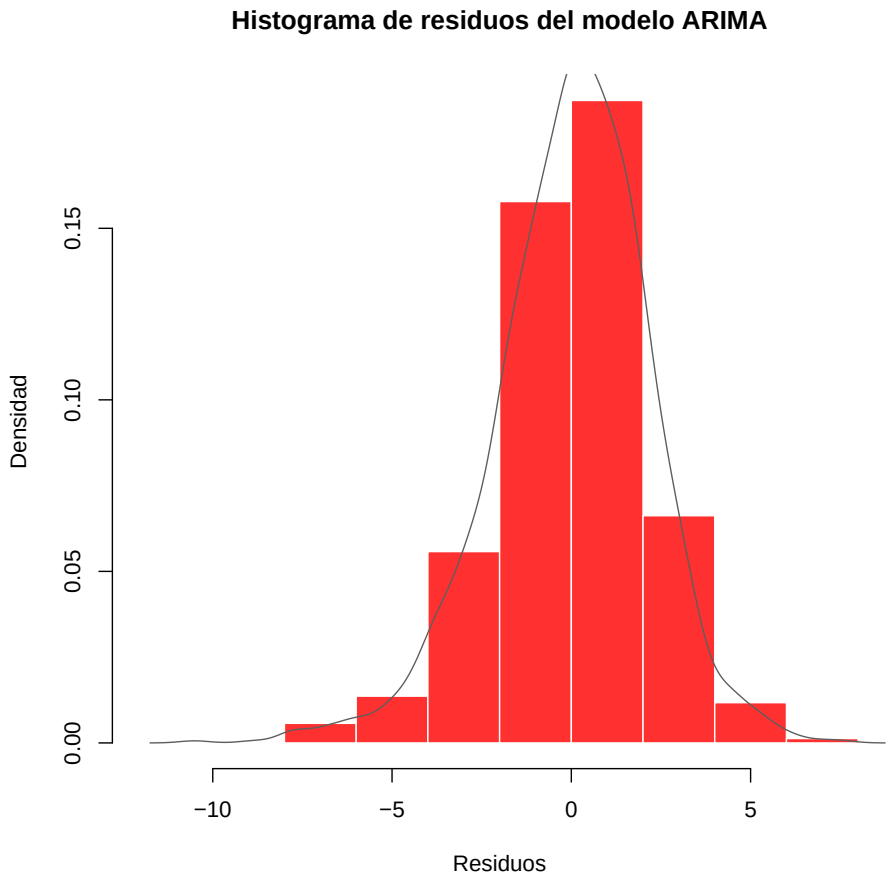

Figura 20: Distribución de los residuos del modelo ARIMA (1,0,0).

La gráfica de la distribución muestra cierta semejanza a la distribución Normal.

Se decice continuar con este modelo, por lo que predecimos en conjunto de test y evaluamos el error SSE cometido tanto sobre el conjunto de *train* como de *test*:

```
# Predecir los conjuntos de train y test y evaluar el modelo
train.pred <- met.train.no_estac - model.arima$residuals # Train

arima.pred.test <- predict(model.arima, n.ahead = length(met.test.no_
    estac))
test.pred <- arima.pred.test$pred # Test

# Calcular los errores sobre el conjunto de train y test
sse.arima.train <- sum((model.arima$residuals)^2)
sse.arima.test <- sum((test.pred-met.test.no_estac)^2)
```

```
10
11 cat('Error SSE cometido sobre el conjunto de train: ', sse.arima.train,
       fill=T)
12 cat('Error SSE cometido sobre el conjunto de test: ', sse.arima.test,
       fill=T)
```

Script 14: Sentencias para la predición del conjunto de test y cálculo de los errores SSE tanto sobre el conjunto de train como de test

Conjunto	SSE
Train	7450.733
Test	2965.2

Cuadro 4: Errores SSE cometidos por el modelo ARIMA (1,0,0) sobre el conjunto de train y test

Visualizamos la predición del modelo sobre ambos conjuntos de *train* y *test* frente a los valores reales:

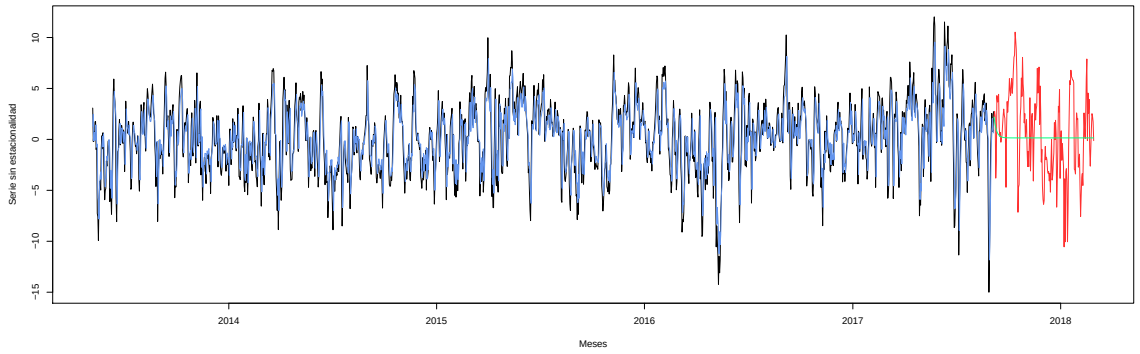

Figura 21: Serie real sin estacionalidad dividida en conjunto de train (color negro) y conjunto de test (color rojo) sobre la que se ha dibujado la predición del modelo sobre el conjunto de train (color azul) y sobre el conjunto de test (color verde). La predición sobre el conjunto de train está solapada con los valores reales, por lo que puede resultar un poco complicado diferenciarlas.

Añadiendo la estacionalidad eliminada anteriormente, la serie predicha y la serie real quedarían de la siguiente forma:

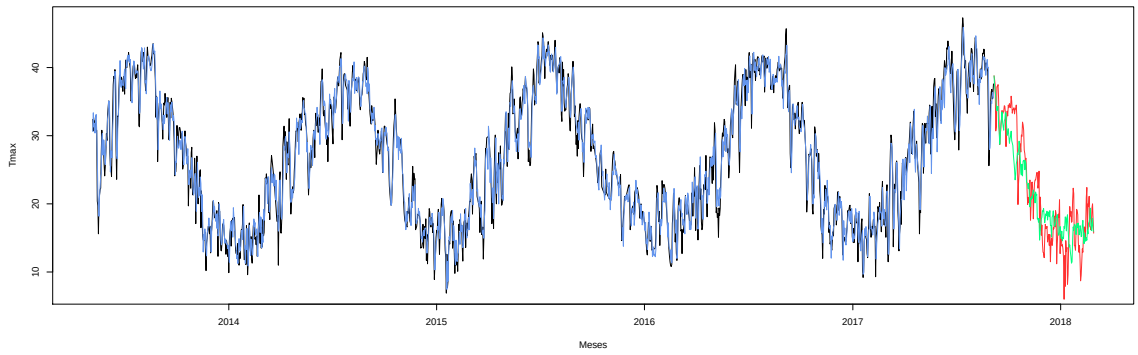

Figura 22: Serie real dividida en conjunto de train (color negro) y conjunto de test (color rojo) sobre la que se ha dibujado la predición del modelo sobre el conjunto de train (color azul) y sobre el conjunto de test (color verde). La predición sobre el conjunto de train está solapada con los valores reales, por lo que puede resultar un poco complicado diferenciarlas.

Se observa que el modelo obtenido presenta alguna dificultades para ajustarse a los datos reales sobre el conjunto de test. No obstante, a grandes rasgos logra predecir con exactitud la evolución de las temperaturas máximas sobre el conjunto de test, por lo que aceptamos el modelo elaborado.

Una vez encontrado un buen modelo para predecir a partir de esta serie temporal, al igual que se hizo anteriormente, se construirá un modelo ARIMA no estacional (1,0,0) ajustando con todos los datos de la serie y predeciremos con el mismo la evolución de las temperaturas máximas durante la primera semana de Marzo de 2018.

Del mismo modo a como se realizó con la serie anterior, se estimará la estacionalidad anual de toda la serie y se eliminará a la misma. A continuación, se estimará el modelo ARIMA con la configuración indicada y se usará el mismo para predecir los 7 días siguientes (primera semana de Marzo de 2018).

Por último, se sumará a la predición realizada la estacionalidad correspondiente para obtener la predición real.

Todo este proceso se implementa en el siguiente conjunto de sentencias:

```
### Predecir toda la serie con el modelo ARIMA

# Eliminar estacionalidad anual de la serie completa
met.matrix <- matrix(data = met$Tmax[1:((length(met$Tmax)%/%365)*365)],
                     ncol = 365, byrow = TRUE)
```

```r
estac.met <- apply(met.matrix, FUN=mean, MARGIN = 2)

rep.estac.met <- c(rep(estac.met,
                       length(met.anual)/length(estac.met)),
                   estac.met[1:(length(met.anual)%%length(estac.met))])

met.no_estac <- met$Tmax - rep.estac.met

# Ajustar modelo ARIMA (1,0,0)
model.arima.full <- arima(met.no_estac, order = c(1,0,0))
model.arima.full

# Predecir la siguiente semana
pred.arima <- predict(model.arima.full, n.ahead = 7)
pred <- pred.arima$pred

# Mostrar el error residual
cat('Error SSE cometido sobre los datos de entrenamiento: ',
    sum(pred.arima$se**2), fill = T)

# Añadir estacionalidad para el tramo predicho
pred.estac.met <- estac.met[(((1:7)+length(met$Tmax))%%length(estac.met
    )) + 1]
pred.estac <- pred + pred.estac.met

# Mostrar predición realizada
print(pred.estac)

# Mostrar gráfica con la predición
pdf('Tmax_predicion_primera_semana_marzo.pdf', width = 20)
plot(as.numeric(pred.estac), xlab='Días de Marzo de 2018',
     ylab='Tmax', type = "l", col = "firebrick1")
dev.off()

# Mostrar gráfica original junto con la predición
pdf('Tmax_predicion_primera_semana_marzo_serie_completa.pdf', width =
    20)
plot(x = met$Fecha, y = met$Tmax,
     xlim=c(met$Fecha[1], met$Fecha[length(met$Fecha)]+2),
     xlab='Año', ylab='Tmax', type = "l", col = "gray37")
lines((met$Fecha[length(met$Fecha)]):(met$Fecha[length(met$Fecha)]+1),
      c(met$Tmax[length(met$Tmax)], pred.estac[1]), col='firebrick1',
    lty = "dashed")
lines((met$Fecha[length(met$Fecha)]+1):(met$Fecha[length(met$Fecha)]+7)
    ,
      pred.estac, col='firebrick1')
legend(x=241, y=27, legend = c('Serie original', 'Predición'),
       fill = c('gray37', 'firebrick1'))
dev.off()
```

Script 15: Sentencias para la elaboración de un modelo ARIMA no estacional (1,0,0) sobre el conjunto de datos de la serie diaria de temperaturas máximas, la estimación del error residual SSE en su ajuste y la predición con el mismo de las temperaturas máximas diarias en la primera semana de Marzo de 2018

El error residual cometido por el modelo en el ajuste sobre los datos de la serie es de 71.08954.

Por su parte, la predición del modelo sobre la evolución de las temperaturas durante la primera semana de Marzo es la siguiente:

Día	Temperatura máxima diaria (ºC)
1 de Marzo de 2018	19.99339
2 de Marzo de 2018	18.67206
3 de Marzo de 2018	17.26416
4 de Marzo de 2018	17.17219
5 de Marzo de 2018	18.79811
6 de Marzo de 2018	20.36844
7 de Marzo de 2018	21.93440

Cuadro 5: Predición de las temperaturas máximas en la primera semana de Marzo de 2018

Visualizando esta predición de forma gráfica:

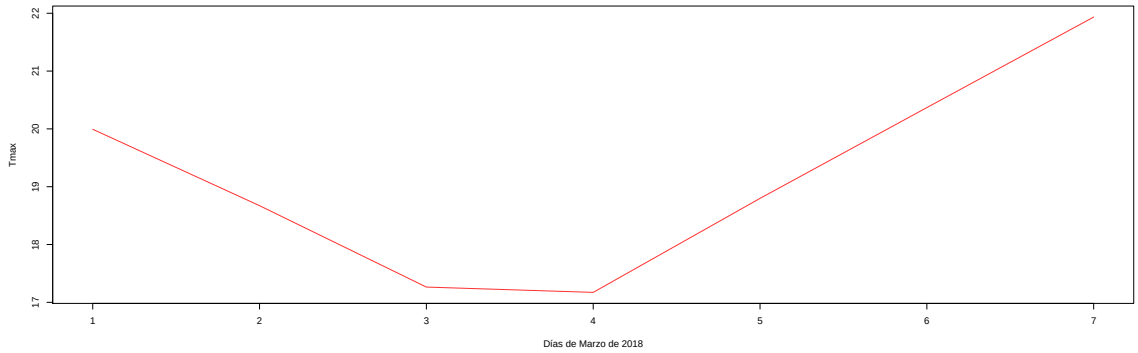

Figura 23: Predición del modelo ARIMA sobre la evolución de las temperaturas máximas durante la primera semana de Marzo de 2018.

Ahora, la serie junto con la predición:

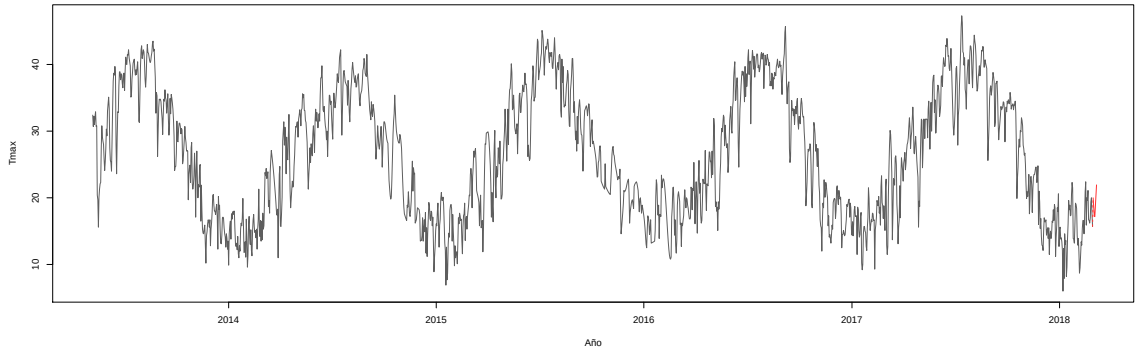

Figura 24: Predición del modelo ARIMA sobre la evolución de las temperaturas máximas durante la primera semana de Marzo de 2018 (color rojo) junto a la serie real original (color negro).

Referencias

[1] R.J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts: Melbourne, Australia, 2nd edition edition, Abril 2018.