Master en Ciencia de Datos e Ingenieria de Computadores

Trabajo autónomo I: Series Temporales

Autor:
Nicolás Delgado Guerrero

Índice

1. Resumen. **2**

2. Parte Teórica. **2**
2.1. Preprocesamiento. . . . 2
2.2. Modelos Estadísticos. . . . 3
2.2.1. Regresión Lineal Multiple. . . . 4
2.2.2. Modelos Autorregresivos. . . . 4
2.2.3. Modelos De Medias Móviles. . . . 4
2.2.3. Modelos ARIMA. . . . 4
2.3. Modelización De La Tendencia y Estacionalidad. . . . 5
2.3.1. Tendencia. . . . 5
2.3.2. Estacionalidad. . . . 5
2.4. Modelización De La Estacionaridad. . . . 5
2.4.1. Selección de parámetros ARIMA(p,d,q). . . . 6

3. Parte Práctica. **6**
3.1. Serie Mensual. . . . 6
3.1.1. Cargar Datos. . . . 6
3.1.2. Preprocesamiento. . . . 7
3.1.3. Análisis Inicial. . . . 11
3.1.4. Construcción Conjuntos Test y Train. . . . 11
3.1.5. Modelización Estacionaridad. . . . 13
3.1.6. Validación del modelo. . . . 17
3.1.7. Predicción de nuevos valores. . . . 20
3.2. Serie Diaria. . . . 20
3.2.1. Preprocesamiento. . . . 20
3.2.2. Construcción Conjuntos Test y Train. . . . 22
3.2.3. Modelización Estacionaridad. . . . 22
3.2.4. Validación del modelo. . . . 27
3.2.5. Predicción de nuevos valores. . . . 30

1. Resumen.

El presente trabajo trata de hacer una predicción sobre las temperaturas máximas del tiempo en distintas regiones de España. Los datos se proporcionan por distintas estaciones meteorológicas esparcidas por toda el país. Escogeremos una de ellas y deberemos responder a dos preguntas. La temperatura máxima para el mes de marzo y abril y las temperaturas máximas de la primera semana de marzo. Para poder responder a estas preguntas cargaremos los datos en nuestro entorno de trabajo Rstudio. Haremos uso de distintas técnicas de preprocesamiento y modelaje de series temporales.

Es por eso que en el trabajo se disponen de dos partes bien diferenciadas. La primera se encargará de dar una introducción teórica a todas las técnicas usadas de preprocesamiento y series temporales. Finalmente en la otra parte del trabajo se verá el código en R necesario para poder llevarlas a cabo en nuestro ejemplo concreto.

2. Parte Teórica.

Comenzaremos estudiando las técnicas que posteriormente usaremos en un problema real. El orden que seguiremos es el usual al que uno hace cuando se enfrenta a un problema de inferencia sobre una serie temporal. Hablaremos de distintas técnicas de preprocesamiento para poder trabajar con la serie, imputación de valores perdidos, normalización, tranformaciones funcionales, etc. Explicaremos distintas técnicas de modelaje y nos centraremos en los modelos ARIMA. Estudiaremos como probar las hipótesis necesarias para el modelo y las heurísticas usadas para la elección de los parámetros del mismo.

Para empezar debemos de comprender que es una serie temporal, por lo que definiremos el término rigurosamente como sigue:

Definición (Serie Temporal):

Diremos que una serie temporal es un conjunto de obserbaciones $\{x_t\}$, estas observaciones son realizaciones muestrales de la distribución conjunta de un una sucesión de variables aleatorias $\{X_t : t \in T \subset \mathbb{R}\}$.

Normalmente el conjunto T representa el tiempo, por eso se le acuña el nombre de serie temporal. Formalmente podriamos decir que cualquier magnitud observada a lo largo del tiempo es una serie temporal. Querer averguiar la distribución de probabilidad subyacente de la serie temporal es demasiado ambicioso por lo que para hacer predicciones se usan modelos que simulen el comportamiento de esta distribución sin necesidad de conocerla. Hay modelos estadísticos (ARIMA), de maching learning (Neural Networks), análisis armónico, etc.

2.1. Preprocesamiento.

Una de las partes más importantes, sino la que más, es el preprocesamiento de los datos. Si no le damos a nuestro algoritmo unos buenos datos de entrada este nunca devolverá unas buenas predicciones, por muy sofisticado que sea el modelo. Datos de calidad es sinonimo de predicciones de calidad.

Uno de los mayores problemas a los que nos enfrentamos en estos datos, es la ausencia de valores, missing values. Por tanto debemos de elegir una buena técnica de imputación de valores. Debido a que los datos que manejamos son numéricos haremos uso de modelos basados en distancias, más concretamente en una imputación por knn.

Knn es un algoritmo que es útil para valorar un punto fijandonos en sus k vecinos más cercanos en un espacio miltidimensional. Se puede usar para datos continuos, discretos, ordinales y categóricos, por eso es una de las técnicas más usadas para la imputación de valores perdidos. Una de las consideraciones más importantes para este algoritmo es la distancia o índice de similitud que usaremos para computar como de similares son unos puntos respectos otros. En este caso usaremos la distancia usual, la Euclidea. La idea que hay detrás de la imputación de valores con knn es que el valor de un punto puede ser aproximado por los valores de sus vecinos más cercanos, basado en sus otras variables.

2.2. Modelos Estadísticos.

Las series temporales contemplan distintos patrones que las caracterizan. Se puede observar como con el paso del tiempo la serie tiende a aumentar ó si sus valores se repiten.

Supondremos que nuestra serie $\{X_t : t \in T\}$ puede descomponerse como suma de tres series donde cada una de ellas representa una característica de $\{X_t : t \in T\}$.

- T_t: es la serie que modela la tendencia, es decir, incrementos o decrementos a largo plazo en los datos.
- S_t: registra la periodicidad o estacionalidad. Como los datos son afectados por un patrón estacional tales como el día del año o el día de la semana.
- E_t: esta es la componente irregular y debe cumplir la propiedad de estacionaridad.

La estacionaridad es una propiedad que en resumen, nos dice que la distribución subyacente del proceso estocástico no varía con el tiempo. Como normalmente los momentos de primer y segundo orden son los que definen los parámetros de una distribución, consideraremos que estos no cambian con el tiempo. Por tanto:

Definición (Estacionaridad):

Se define una serie $\{X_t : t \in T\}$ estacionaria como aquella que verifica las siguientes afirmaciones:

i) $\mathrm{Var}(X_t) < +\infty$, $\forall t \in T$.

ii) $\mu_X(t) = \mu$, $\forall t \in T$.

iii) $\gamma_X(r, s) = \gamma_X(r + t, s + t)$, $\forall r, s, t \in T$.

Esta última propiedad implica que $\gamma_X(r, s)$ es una función de $r - s$ y conveniente definirla

$$\gamma_X(h) \stackrel{def}{=} \gamma_X(h, 0).$$

El valor h se refiere al "lag".

Definición (ACVF, ACF):

Sea $\{X_t : t \in T\}$ una serie estacionaria. La función de autocovarianza (ACVF) de $\{X_t\}$ es

$$\gamma_X(h) = \mathrm{Cov}(X_{t+h}, X_t).$$

La función de autocorrelación (ACF) es

$$\rho_X(h) \stackrel{def}{=} \frac{\gamma_X(h)}{\gamma_X(0)} = \frac{\mathrm{Cov}(X_{t+h}, X_t)}{\mathrm{Var}(X_t)} = \frac{\sum_{t=1}^{T-h}(x_{t+h} - \bar{x})(x_t - \bar{x})}{\sum_{t=1}^{T}(x_t - \bar{x})^2}.$$

La correlación nos indica la dependencia lineal que hay entre dos variables aleatorias, en este caso, la función de autocorrelación parcial nos informaría de las dependencias lineales entre un instante y los posteriores o los anteriores ($h > 0$ ó $h < 0$). Si visualizamos la gráfica de esta función de autoccorrelación podremos distinguir ciertas propiedades interesantes que nos ayuden a seleccionar el mejor modelo para la serie temporal.

Del mismo modo se define la función de autocorrelación acumulada (PACF). En el lag h, es la autocorrelación entre los valores de las series que se encuentran a h intervalos de distancia, teniendo en cuenta los valores de los intervalos intermedios.

Nosotros hemos tomado una hipótesis de descomposición aditiva:

$$X_t = T_t + S_t + E_t,$$

pero dependiendo del problema a lo mejor una descomposición en con productos o mixta puede ser más óptima para la resolución del problema.

Nuestro objetivo será modelizar la tendencia, la estacionalidad y restarlas a nuestra serie original. Con ello habremos obtenido una serie estacionaria, esta serie la modelizaremos mediantes técnicas derivadas de regresiones lineales multiples, llamados modelos ARIMA. Finalmente, si queremos obtener valores predictorios de nuestra series X_t solo habrá que sumarte la tendencia y estacionalidad a la predicción del modelo ARIMA.

Vamos a presentar brevemente la regresión lineal y posteriormente los modelos de medias móviles y autoregresivos. Finalmente entenderemos como a partir de estos modelos podemos construir un modelo ARIMA.

2.2.1. Regresión Lineal Multiple.

El proposito de la regresión lineal multiple es predecir Y a partir de un hiperplano formado por las variables $X_1, X_2, \ldots, X_k$:

$$Y = \sum_{i=1}^{k} \beta_i X_i + \beta_0 + \epsilon.$$

Los valores de ϵ definen los errores de la regresión $\epsilon = Y - \sum_{i=1}^{k} \beta_i X_i - \beta_0$. Usaremos el criterio de los mínimos cuadrados para inferir los parámteros β.

2.2.2. Modelos Autorregresivos.

Los modelos autorregresivos es una regresión lineal multiple de la serie temporal, en este caso queremos predecir un valor en el instante t y nos basaremos en los p instantes anteriores.

$$X_t = \sum_{i=1}^{p} \beta_i X_{t-i} + \beta_0 + \epsilon_t.$$

Por lo tanto usaremos las mísma técnica de minimizar el cuadrado de los residuos para obtener los parámetros β. Los procesos autoregresivos no pueden representar series de memoria muy corta, donde el valor actual de la serie sólo está correlado con un número pequeño de valores anteriores de la serie.

2.2.3. Modelos De Medias Móviles.

La idea de los modelos de medias moviles es hacer una ponderación de los valores de la serie en los q instantes anteriores y posteriores a t. Uno de los mayores problemas de este tipo de modelos es que cuando aumentamos el q perdemos información ya que nuestra serie modelada usa $2q$ valores menos de la serie original.

$$\hat{X}_t = \frac{1}{m} \sum_{k=-q}^{q} X_{t+q}.$$

2.2.3. Modelos ARIMA.

La clase de modelos ARIMA es amplia y flexible, ya que combina las estructuras AR y MA. Es útil para representar una gran variedad de series utilizando pocos parámetros. Se modelizan con la siguiente expresión:

$$X_t = \sum_{i=1}^{p} \beta_i X_{t-i} + \beta_0 + \epsilon_t - \sum_{i=1}^{q} \phi_i \hat{X}_{t-i}.$$

Unos de los grandes retos es la selección de los parámetros p y q par obtener el mejor modelo para nuestra serie estacionaria, además aparece otro parámetro correspondiente a las diferencias de la serie en caso que esta no sea estacionaria. Hablaremos más adelante de las técnicas usadas para la selección de los parámetros.

2.3. Modelización De La Tendencia y Estacionalidad.

2.3.1. Tendencia.

La tendencia debe recoger el movimiento a largo plazo de una serie, independientemente de otros componentes irregulares. Es decir, debe recoger el nivel subyacente y regular de la serie.

Los modelos más simples para la tendencia son regresiones de la variable con respecto al tiempo. Un modelo de tendencia lineal sería (cuando la misma crece o decrece):

$$T_t = \beta_0 + \beta_1 t \quad \forall t \in 1, 2, \ldots, n,$$

y un modelo de tendencia cuadrático:

$$T_t = \beta_0 + \beta_1 t + \beta_2 t^2 \quad \forall t \in 1, 2, \ldots, n,$$

Estos modelos explican la evolución pasada de una serie en función de pautas simples, pero tienen problemas y limitaciones.

Aunque son útiles para describir las pautas que sigue una serie temporal, las predicciones que proporcionan suelen ser muy malas (es decir, con un gran error asociado).

La razón de esto es que en una serie temporal la observación más reciente depende, en general, de sus valores pasados, pero esta dependencia suele ser más fuerte con los datos más recientes y más débil con los más alejados. Los modelos de tendencias deterministas proporcionan predicciones que no utilizan esta propiedad.

2.3.2. Estacionalidad.

Estacionalidad: es un cambio en la media de la serie que se repite periódicamente cada s estaciones. Si la serie es mensual, s=12; si es trimestral, s=4; si es semanal, s=52 ó 53, etc.

Una forma determinista de captar la estacionalidad consiste en definir variables dummies (con valores 0 ó 1). Por ejemplo, para una serie mensual definir las doce dummies correspondientes a Enero (S1), Febrero (S2), ..., hasta Diciembre (S12). Se puede escribir como:

$$S_t = \beta_1 S1_t + \cdots + \beta_{12} S12_t.$$

2.4. Modelización De La Estacionaridad.

Una vez que ya hayamos modelado la tendencia y la estacionalidad de nuestra serie, deberiamos tener una serie E_t estacionaria. Debemos comprobar que en efecto la serie es estacionaria, para ello se realiza el est de Dickey-Fuller aumentado. Este se construye a para contrastar las siguientes hipóteisis:

$$\begin{cases} H_0 : \text{Existe una raíz unitaria,} \\ H_1 : \text{No existe una raíz unitaria,} \end{cases}$$

donde una raíz unitaria es una característica de los procesos que evolucionan a través del tiempo y que puede causar problemas en inferencia estadística en modelos de series temporales.

Por tanto en caso de obtener un p-valor muy pequeño rechazaremos la hipótesis nula y concluiremos que hay evidencias estadísticas suficientes para suponer la estacionaridad de esa serie.

En caso de que la serie E_t no fuera estacionaria podemos diferenciarla tantas veces hasta que sea estacionaria.

$$E'_t = E_t - E_{t-d},$$

donde d =instanse de tiempo de diferenciación.

2.4.1. Selección de parámetros ARIMA(p,d,q).

Una vez ya hayamos probado la estacionaridad de la serie E_t, usaremos un modelo ARIMA para modelizarla. Este modelo consiste en una combinación de medias móviles y autorregresivos. El parámetro p corresponde al número de instantes anteriores de la serie que consideraremos para construir la parte autorregresiva, la q es el parámetro correspondiente a las medias móviles y es el número de observaciones que vamos a usar para ponderación. Finalmente d será el número de veces que hemos diferenciado la serie hasta que sea estacionaria, si no fuera necesario diferencias $d = 0$.

Está claro que valor de d tomar, pero que "trucos" o métodos debemos seguir para la elección del p yq. La verdad que no hay una ciencia exacta pero si hay algunos truquillos que podemos tomar para hacer pruebas con esos parámetros. Para ello debemos de fijarnos en la función de correlación parcial y la acumulada, dependiendo de la forma de estas podremos intuir que tipo de modelos son más apropiados para nuestro problema.

Modelos AR y MA:

- Como norma general, los modelos AR tienen una función de autocorrelación (ACF) que decrece a 0, la convergencia a cero puede darse de diferentes formas: regulares, sinusoidales, alternando valores positivos y negativos, etc. El orden de p es tantos valores "distintos" de 0 como haya en la función de autocorrelación acumulada (PACF).
- Los modelos de medias móviles tienen un PACF que decrece a 0, lo puede hacer de diferentes formas: regulares, sinusoidales, alternando valores positivos y negativos, etc. El valor de q es tantos valores distintos de 0 como haya en el ACF.
- Dieremos que n valor se considera distintos de cero cuando no pertenezca al intervalo $\left[\frac{-2}{\sqrt{N}}, \frac{2}{\sqrt{N}}\right]$ con N =longitud de la serie. En la parte práctica verémos como las gráficas de las funciones ACF y PACF muestran una recta azul índicando los valores distintos de cero.

3. Parte Práctica.

Para los experimentos prácticos usaré los datos de la estación meteorlógica de San Clemente en Cuenca. Nuestro propósito es responder a las siguientes preguntas:

- ¿Qué valores de temperatura máxima, a escala mensual, se espera que tengan los meses de Marzo y de Abril de 2018?
- ¿Qué valores de temperatura máxima, a escala diaria, se espera para la primera semana de Marzo de 2018?

Para ello modelizaremos dos series temporales, una a escala mensual y otra a escala diaria. Seguiremos los pasos de preprocesamiento y modelización mediantes técnicas ARIMA descritas en la parte teórica.

Haremos todo el proceso primero para la serie a escala mensual y posteriormente la diaria.

3.1. Serie Mensual.

3.1.1. Cargar Datos.

El código de la estación de San Clemente viene dado por el código 4090y. Cargaremos los datos del archivo .csv correspondiente.

```
datos = read.csv("datos/datos/DatosEstaciones - 2018-02/4090Y.csv",
                 sep = ";")
head(datos)
```

```
##      Id      Fecha Tmax HTmax Tmin HTmin Tmed Racha HRacha Vmax HVmax TPrec
## 1 4090Y 2013-05-07   NA    NA   NA    NA   NA    NA    NA    NA    NA    NA
## 2 4090Y 2013-05-08   NA    NA   NA    NA   NA    NA    NA    NA    NA    NA
## 3 4090Y 2013-08-12 35.6 17:10 19.5 07:00 27.5    55 23:20    29 23:00     0
```

```
## 4 4090Y 2013-08-13 34.3 18:50 18.9 06:40 26.6    46  21:30   26 00:20     0
## 5 4090Y 2013-08-14 33.9 18:20 17.7 07:30 25.8    39  10:10   21 10:50     0
## 6 4090Y 2013-08-15   NA         NA         NA    NA          NA          NA
##   Prec1 Prec2 Prec3 Prec4
## 1    NA    NA    NA    NA
## 2    NA    NA    NA    NA
## 3     0     0     0     0
## 4     0     0     0     0
## 5     0     0     0     0
## 6    NA    NA    NA    NA
```

El conjunto de mediciones meteorológicas consta de las siguientes variables:

- Columna 1 : Identificador Estación
- Columna 2 : Fecha
- Columna 3 : Temperatura Máxima (ºC)
- Columna 4 : Hora Temperatura Máxima
- Columna 5 : Temperatura mínima (ºC)
- Columna 6 : Hora Temperatura mínima
- Columna 7 : Temperatura Media (ºC)
- Columna 8 : Racha máxima de viento (Km/h)
- Columna 9 : Hora de Racha Máxima
- Columna 10 : Velocidad media de Viento (Km/h)
- Columna 11 : Hora de Velocidad Máxima de viento
- Columna 12 : Precipitación Total diaria (mm)
- Columna 13 : Precipitación de 0 a 6 horas (mm)
- Columna 14 : Precipitación de 6 a 12 horas (mm)
- Columna 15 : Precipitación de 12 a 18 horas (mm)
- Columna 16 : Precipitación de 18 a 24 horas (mm)

3.1.2. Preprocesamiento.

Hay variables que pueden ser redundantes para nuestro problema, por ello descataremos las siguientes columnnas. El identificador de estación no nos provee de ninguna información ya que es el mismo en todas las instancias. Las variables horarias no nos preocupan por lo tanto nuestro conjunto de datos nuevo va a estar formado de la siguiente forma:

```
datos = datos[, c(2,3,5,7,8,10,12)]
head(datos)
```

```
##        Fecha Tmax Tmin Tmed Racha Vmax TPrec
## 1 2013-05-07   NA   NA   NA    NA   NA    NA
## 2 2013-05-08   NA   NA   NA    NA   NA    NA
## 3 2013-08-12 35.6 19.5 27.5    55   29     0
## 4 2013-08-13 34.3 18.9 26.6    46   26     0
## 5 2013-08-14 33.9 17.7 25.8    39   21     0
## 6 2013-08-15   NA   NA   NA    NA   NA    NA
```

Podemos observas como hay muchos valores perdidos en los datos. Veamos gráficamente la distribución de los datos perdidos para cada variable.

```
library(mice)
```

```
## Loading required package: lattice
```

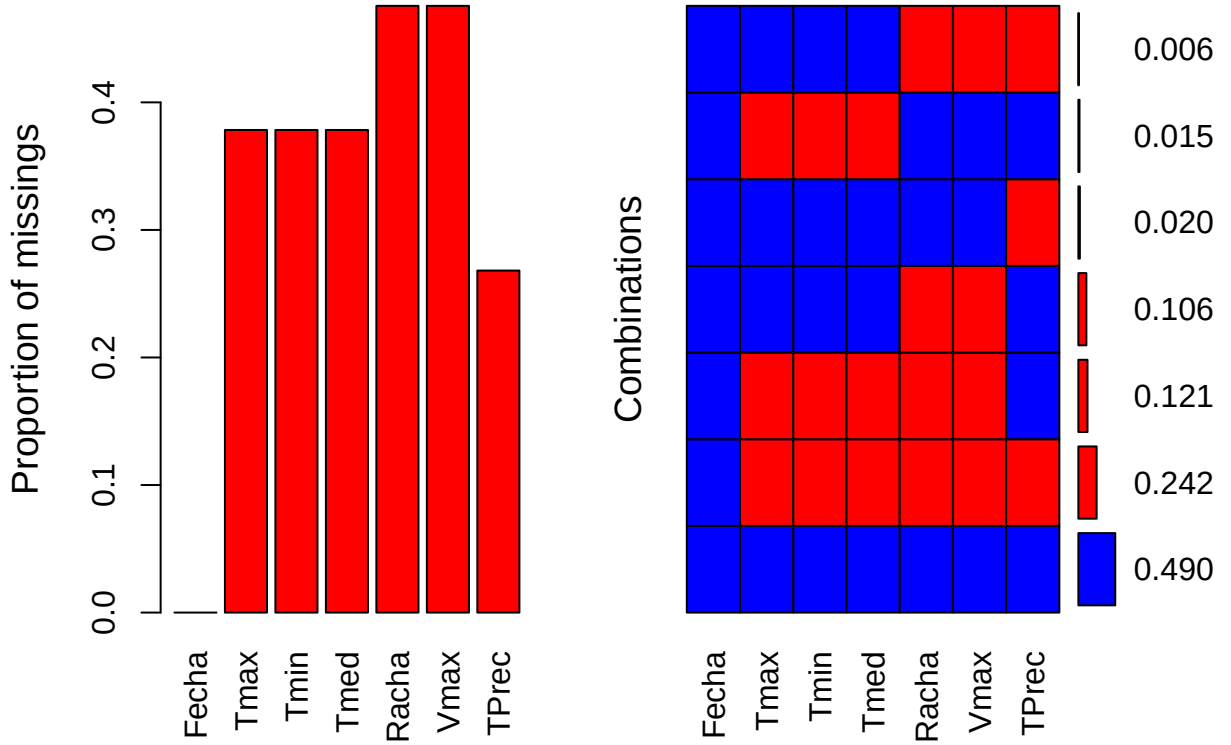
```
##
## Attaching package: 'mice'
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
VIM::aggr(datos, col = c("blue", "red"), numbers = TRUE)
```

```
## Registered S3 methods overwritten by 'car':
##   method                          from
##   influence.merMod                lme4
##   cooks.distance.influence.merMod lme4
```

```
##   dfbeta.influence.merMod         lme4
##   dfbetas.influence.merMod        lme4
```



Los valores en rojo presenta missing values, hay un alto número de ellos. Para nuestra desgracia observamos que hay un porcentaje del 24.2 % de instancias que presentan valores perdidos en todas las variables menos la fecha :(. Lo que vamos a hacer es eliminar estas observaciones ya que no proveen de ninguna información.

```
indices = NULL
# Seleccionamos las filas de nuestro dataset que no presentas NA en muchas de
# las variables
for (fila in 1:1007){
  if (sum(is.na(datos[fila,2:length(datos)])) < 4){
    indices = c(indices, fila)
  }
}
datos = datos[indices,]
```

Como todos los valores de nuestro conjunto de datos, excepto la fecha, son variables numéricas y continuas, haremos una imputación de valores perdidos fijandonos en los vecinos más cercanos.

```
library(robCompositions)
```

```
## Loading required package: ggplot2

## Loading required package: pls

##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

```
## Loading required package: data.table
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
## sROC 0.1-2 loaded
```

```
imputados = robCompositions::impKNNa(datos[,2:length(datos)], primitive = TRUE)
```

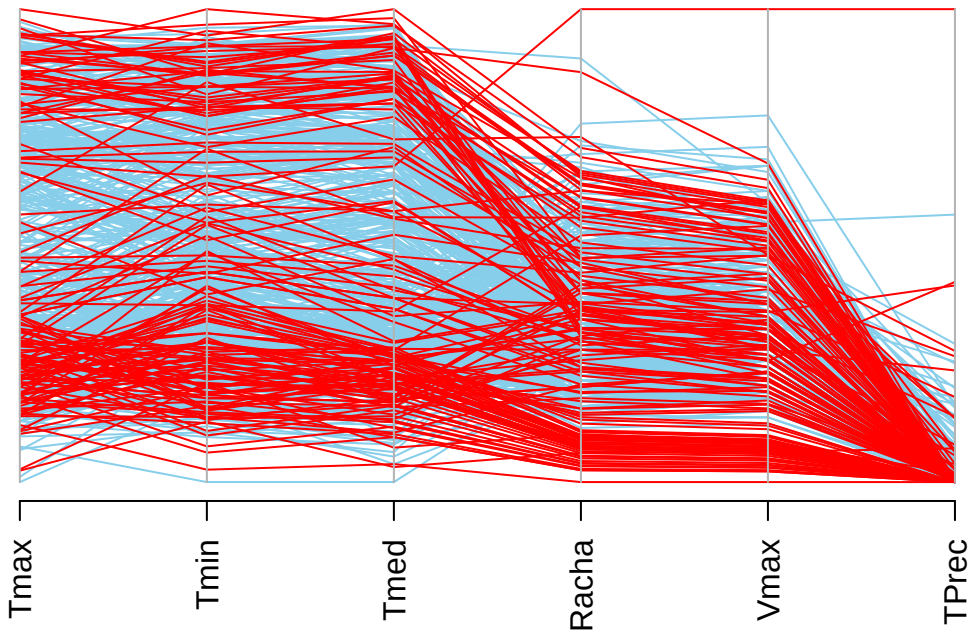
Como la variable que nos interesa es la temperatura máxima, vamos a comprobar que no tiene ningún valor perdido.

```
# La primera columna se refiere a la v. temperatura máxima
any(is.na(imputados$xImp[,1]))
```

```
## [1] FALSE
```

Podemos voisualizar como han sido imputados los valores.

```
plot(imputados, which = 2)
```



El siguiente gráfico muestra las instancias como líneas (en azul las completas y en rojo aquellas en que se hizo imputación). Permite comprobar de forma visual si alguna instancia imputada se sale completamente del patrón general o todas ellas se comportan de la forma esperada. Para la variable en la que estamos interesados, Tmax, no observamos una tendencia de los valores rojos muy distantes a las lineas azules.

Generemos nuestra serie temporal con los datos que tenemos sobre las temperaturas máximas. Generamos un dataframe con dos columnas, la fecha y Tmax que son los datos relevantes para generar nuestra serie mensual y poder predecir temperaturas máximas de nuevos meses.

```
datos_serie = data.frame(Fecha = datos$Fecha, Tmax = imputados$xImp[,1])
head(datos_serie)
```

```
##         Fecha Tmax
## 3  2013-08-12 35.6
## 4  2013-08-13 34.3
## 5  2013-08-14 33.9
## 8  2013-08-17 34.9
## 9  2013-08-18 35.7
## 10 2013-08-19 37.2
```

Crearemos dos columnas más mes y año para poder encontrar la temperatura máxima en los meses de cada año.

```
meses = substring(as.character(datos$Fecha),6,7)

years = substring(as.character(datos$Fecha),1,4)

datos_serie = data.frame(Fecha = datos$Fecha, Tmax = imputados$xImp[,1],
                         Mes = as.numeric(meses), Year = as.numeric(years))

head(datos_serie)
```

```
##         Fecha Tmax Mes Year
## 3  2013-08-12 35.6   8 2013
## 4  2013-08-13 34.3   8 2013
## 5  2013-08-14 33.9   8 2013
## 8  2013-08-17 34.9   8 2013
## 9  2013-08-18 35.7   8 2013
## 10 2013-08-19 37.2   8 2013
```

Tenemos muy pocos datos para los años comprendidos entre 2013 y mediados de 2016 por lo tanto descartaremos esos datos ya que pueden contribuir a una creencia de tendencia erronea para nuestra serie temporal.

```
temperaturas = NULL

for (year in 2016:2018){
  aux_year = datos_serie[which(datos_serie$Year == year),]
  for (mes in 1:12){
    if (any(aux_year$Mes == mes)){
      aux_mes = aux_year[which(aux_year$Mes == mes),]
      maxima = max(aux_mes$Tmax)
      temperaturas = c(temperaturas, maxima)
    }
  }
}
```

Como del últimos mes no tenemos pocos ejemplos, eliminamos la temperatura máxima del mes de Febrero de 2018 y comenzamos en la tercera para eliminar los meses de Enero y Febrero del año 2016.

```
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 3.6.3
```

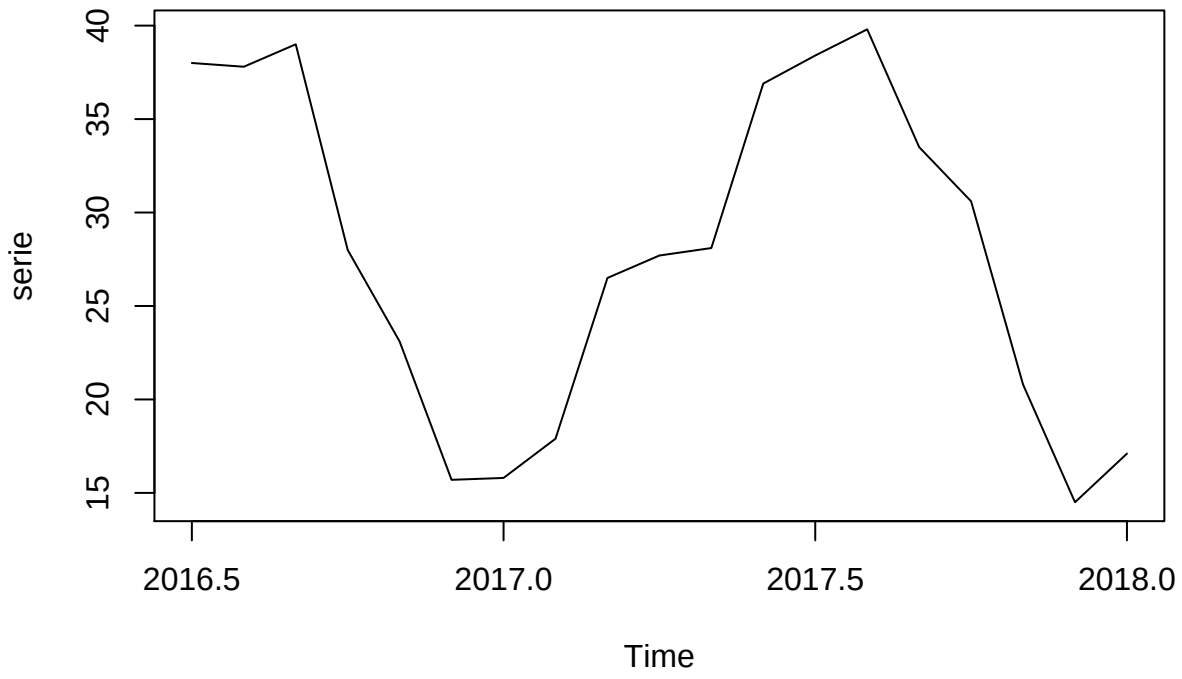
```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

```
## Registered S3 method overwritten by 'quantmod':
##   method           from
##   as.zoo.data.frame zoo
```

```
serie = ts(temperaturas[3:(length(temperaturas)-1)], start = c(2016,7),
           frequency = 12)
serie
```

```
##       Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
## 2016                              38.0 37.8 39.0 28.0 23.1 15.7
## 2017 15.8 17.9 26.5 27.7 28.1 36.9 38.4 39.8 33.5 30.6 20.8 14.5
## 2018 17.1
```

```
plot(serie)
```



Los picos representan los meses de verano y los valles los de invierno.

3.1.3. Análisis Inicial.

En una primera observación de la serie después de haber preprocesado los datos, vemos que no ocurre ninguna locura. Se comporta como se esperaría, sabemos en en verano suben las temperatura y en inverno disminuyen, además lo hacen paulativamente. Esto es algo que ocurre todos los años, cada vez menos por culpa del calentamiento global, pero podemos concluir que los datos que obtuvimos son de calidad y por ente conseguiremos buenas predicciones.

En la gráfica anterior como ya comentamos hay presencia de estacionalidad, cada pico representa un verano por lo que hay una estacionalidad anual. Parece que no hay ningún tipo de tendencia. Solo tenemos datos de un año completo, lo ideal sería tener datos de más años para ver si hubuise alguna tendencia ascendente de las temperaturas, o un rango de valores más extremos por culpa del calentamiento global como ya habiamos comentado anteriormente.

Asumiremos por tanto que no hay tendencia y que hay una estacionalidad anual. Como los datos que disponemos son solo de un año completo, tampoco asumiremos la presencia de una serie estacional.

3.1.4. Construcción Conjuntos Test y Train.

Para encontrar el mejor modelo en la inferencia de nuevos valores de la serie, dividiremos los datos en dos conjuntos disjuntos, uno de ellos de aproximadamente un 80 % de los datos será usado para entrenar nuestro

modelo. Una vez tengamos un modelo, este será evaluado con distintas métricas de bondad frente al conjunto test. Cuando seleccionemos el mejor modelo, usaremos el total de datos para entrenarlo. Con ello ya estaremos en disposición de predecir nuevos valores de la serie temporal.

Cogeremos, por ejemplo, los 3 últimos valores para el test dado que también necesitaremos predecir 3 valores de la serie.

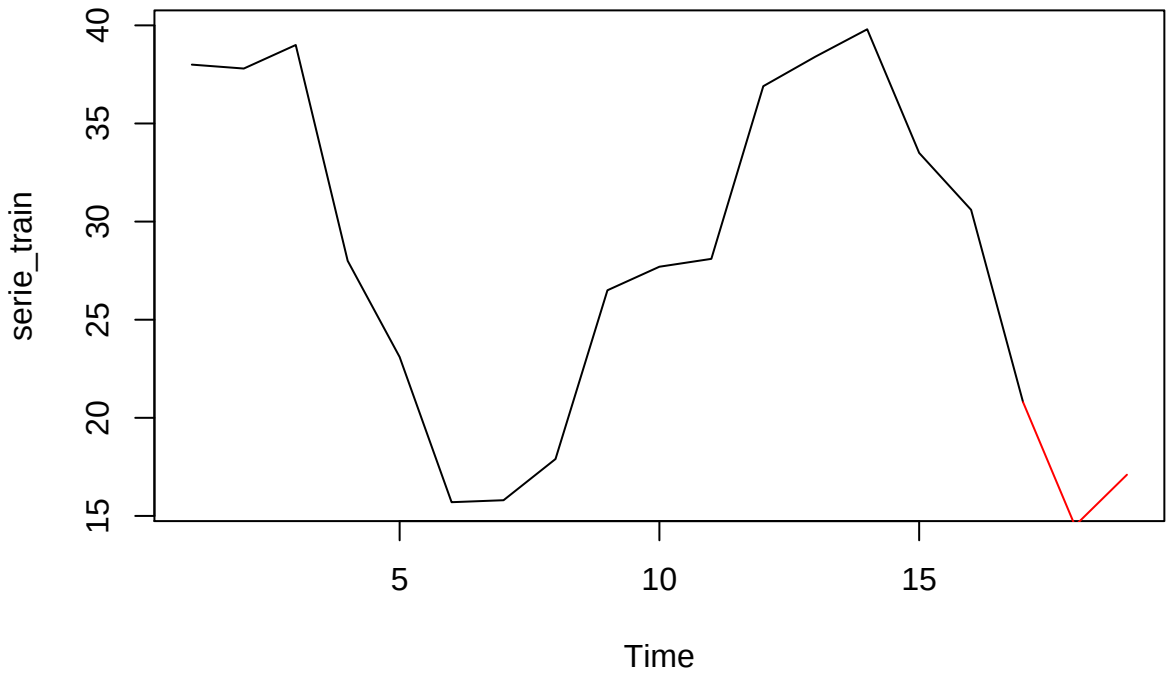
```
#Número de valores que queremos inferir
N_pred = 3
#Número de valores destinados al test
N_test = 3

#Conjunto Train
serie_train = serie[1:(length(serie) - 2)]
tiempo_train = 1:length(serie_train)

#Conjunto Test
serie_test = serie[(length(serie_train)):length(serie)]
tiempo_test = (length(tiempo_train)):(length(tiempo_train) + 2)
```

Visualizamos la serie train, en negro, y la serie test, en rojo.

```
plot.ts(serie_train, xlim = c(1, tiempo_test[length(tiempo_test)]))
lines(tiempo_test, serie_test, col = "red")
```



3.1.5. Modelización Estacionaridad.

En la descomposición aditiva, se trabaja con la hipótesis de que nuestra serie se descompone como suma de tres series: T_t, S_t y E_t. Este caso vamos a asumir que nuestra serie solo presenta la parte irregular que debe ser estacionaria para poder ser modelizada mediante técnicas ARIMA.

Con la serie sin tendencia ni estacionalidad, debemos comprobar si es estacionaria antes de hipotetizar modelos de predicción. Usamos el Test de Dickey-Fuller aumentado:

$$\begin{cases} H_0 : \text{Existe una raíz unitaria,} \\ H_1 : \text{No existe una raíz unitaria,} \end{cases}$$

donde una raíz unitaria es una característica de los procesos que evolucionan a través del tiempo y que puede causar problemas en inferencia estadística en modelos de series temporales.

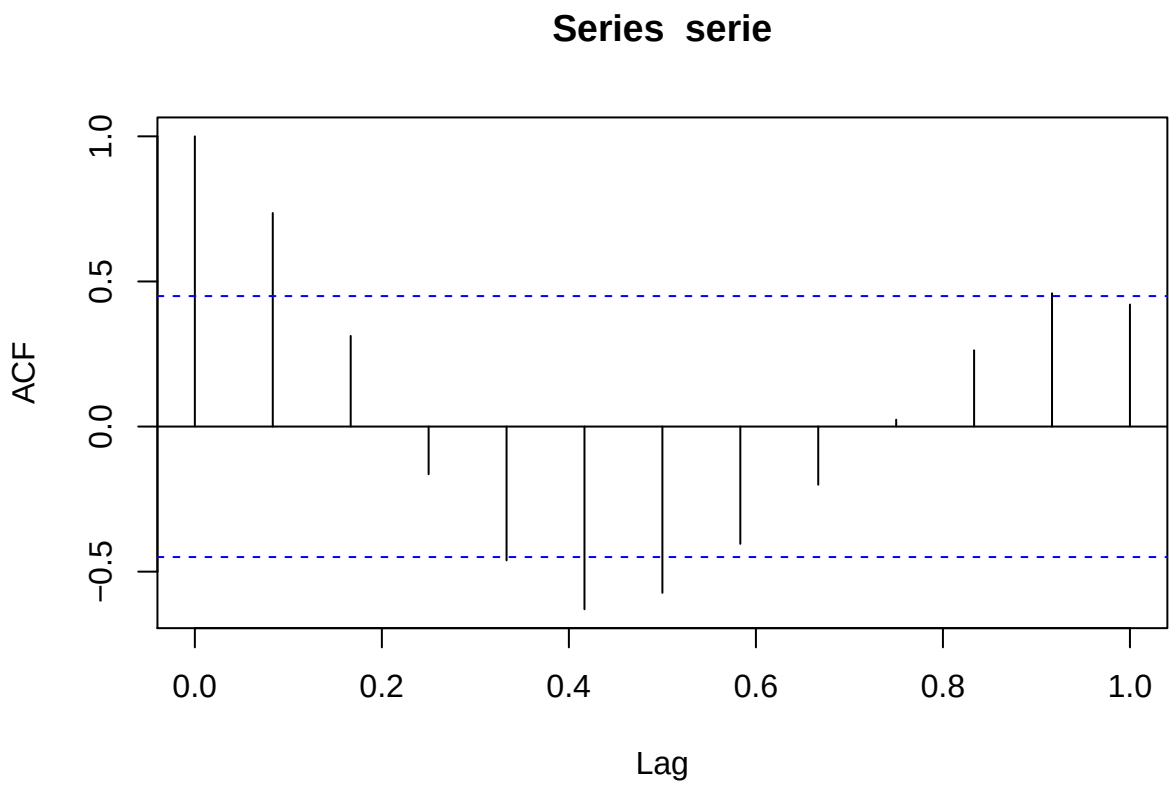
```
adftest = adf.test(serie)
adftest
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  serie
## Dickey-Fuller = -3.9604, Lag order = 2, p-value = 0.02464
## alternative hypothesis: stationary
```

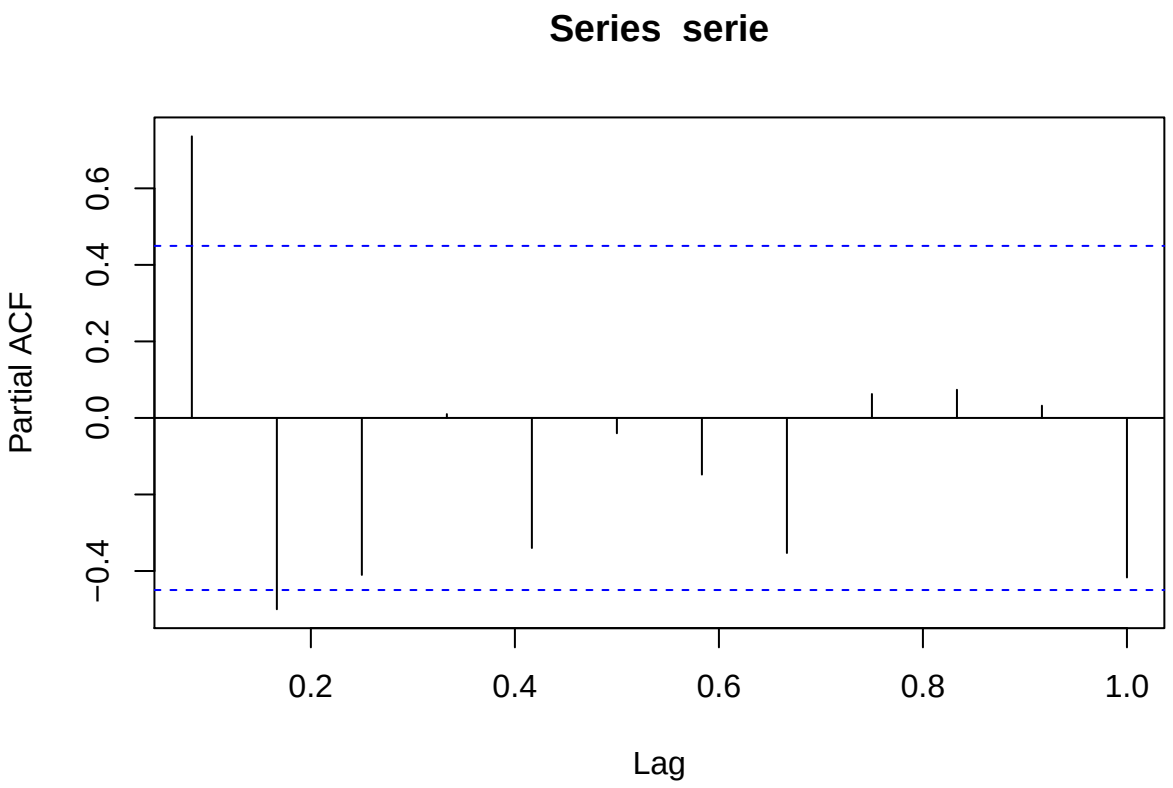
Otenemos un p-valor suficientemente pequeño para descartar la hipótesis nula y aceptar la estacionaridad de nuestra serie.

Visualizamos las gráficas de las de funciones de autocorrelación parcial y acumulada. Con ellas podremos deducir que parámetros de los modelo ARIMA pueden ser lo mejores para nuestro problema.

```
acf(serie)
```




```
pacf(serie)
```



No se ve un decrecimiento rápido de ninguna de las funciones, si es verdad que el PACF tiene menos oscilaciones que el ACF y puede ser que se trate de un modelo autoregresivo. Probaremos con un modelo de medias móviles de parámetro 3 ya que hay tres valores antes de encontrarnos lo que uno puede interpretar como cero en el ACF. Si usamos estos parámetros conseguimos las siguientes predicciones:

```
#Ajustamos modelo
modelo = arima(serie_train, order = c(3, 0, 0))
valores_ajustados = serie_train + modelo$residuals

#Predecimos Valores
predicciones = predict(modelo, n.ahead = N_pred)
valores_predichos = predicciones$pred

valores_predichos
```

```
## Time Series:
## Start = 18
## End = 20
## Frequency = 1
## [1] 18.92615 16.81585 20.99625
```

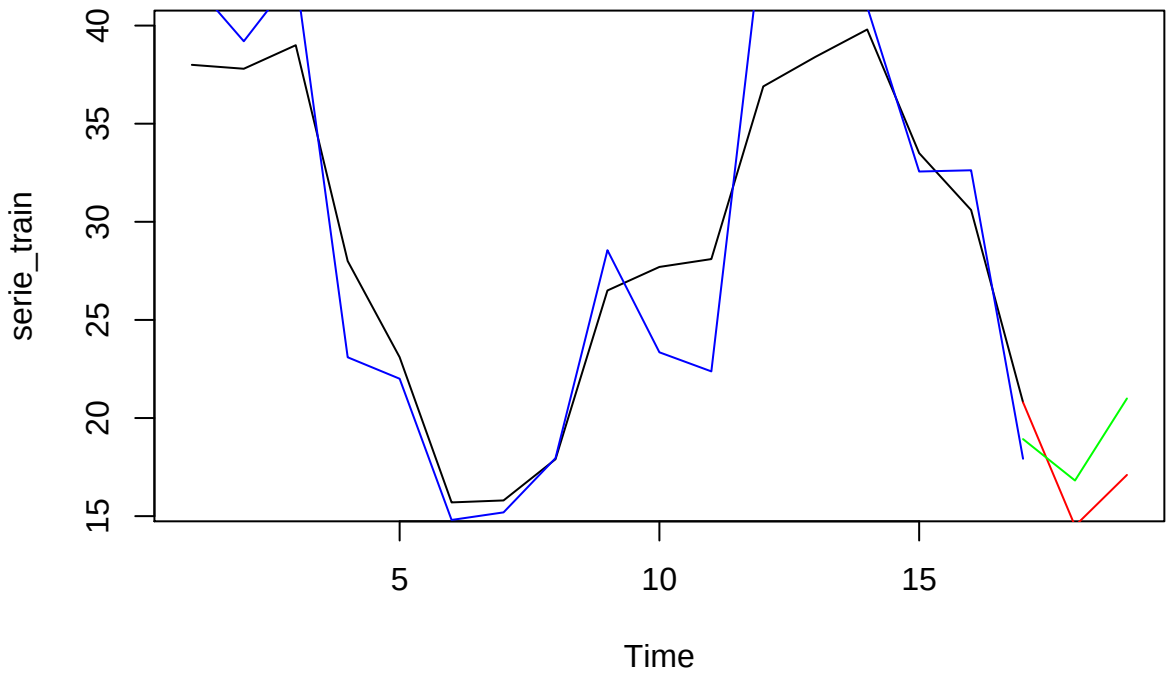
```
serie_test
```

```
## [1] 20.8 14.5 17.1
```

```
#Error cuadrático acumulado para Train y Test
error_train = sum((modelo$residuals)^2)
error_test = sum((valores_predichos - serie_test)^2)
```

Graficamente se ve que:

```
plot.ts(serie_train, xlim = c(1, tiempo_test[length(tiempo_test)]))
lines(valores_ajustados, col = "blue")
lines(tiempo_test, serie_test, col = "red")
lines(tiempo_test, valores_predichos, col = "green")
```



La curva predicha es fidedigna a la real. La serie que obtenemos presenta muchos picos, podemos incluir modelos de medias móviles para suavizar el efecto de la curva. Probemos con una mezcla de los modelos autoregresivos y medias móviles para suavizar este comportamiento.

```
#Ajustamos modelo
modelo = arima(serie_train, order = c(3, 0, 3))
```

```
## Warning in arima(serie_train, order = c(3, 0, 3)): possible convergence problem:
## optim gave code = 1
```

```
valores_ajustados = serie_train + modelo$residuals

#Predecimos Valores
predicciones = predict(modelo, n.ahead = N_pred)
valores_predichos = predicciones$pred

valores_predichos
```

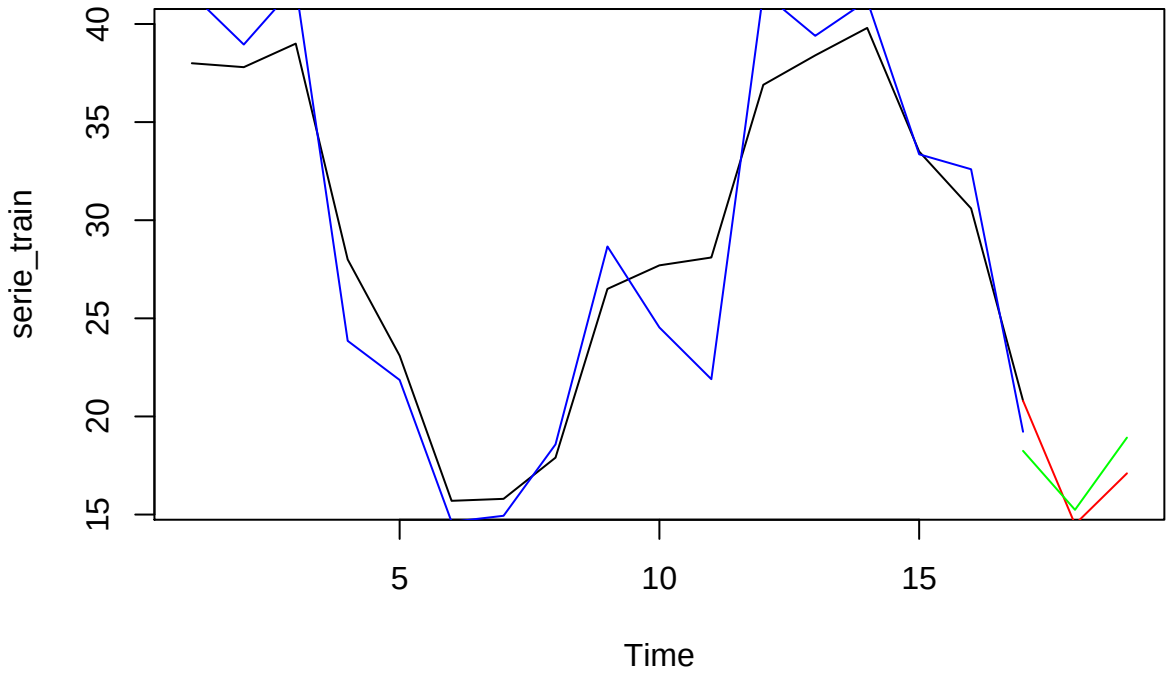
```
## Time Series:
## Start = 18
## End = 20
## Frequency = 1
## [1] 18.24446 15.24159 18.92051
```

```
serie_test
```

```
## [1] 20.8 14.5 17.1
```

```
#Error cuadrático acumulado para Train y Test
error_train = sum((modelo$residuals)^2)
error_test = sum((valores_predichos - serie_test)^2)

plot.ts(serie_train, xlim = c(1, tiempo_test[length(tiempo_test)]))
lines(valores_ajustados, col = "blue")
lines(tiempo_test, serie_test, col = "red")
lines(tiempo_test, valores_predichos, col = "green")
```



En efecto, conseguimos una curva de predicción buena y en general un modelo más suave debido a los términos de las medias móviles.

3.1.6. Validación del modelo.

Cualquier buen modelo de predicción debe verificar las siguientes propiedades para sus residuos:

- No están correlados, independencia lineal.
- La media de los residuos es cero.

Un modelo que no verifique las anteriores afirmaciones siempre puede ser mejorado. Además estamos interesados que se cumpla que:

- Los residuos tienen varianza constante.
- Los residuos se distribuyen según una normal.

```r
#Media aproximadamente cero
mean(modelo$residuals)
```

```
## [1] 0.0668222
```

Probemos ahora la normalidad de los residuos. Primero los someteremos al test de Box-Pierce para aleatoriedad de los residuos. Finalmente los tests de Jarque Bera y Shapiro-Wilk para normalidad de residuos.

```r
#Test de aleatoridad
Box.test(modelo$residuals)
```

```
##
##  Box-Pierce test
##
## data:  modelo$residuals
## X-squared = 0.10999, df = 1, p-value = 0.7402
```

El Test de Box-Pierce se formula con las siguientes hipótesis nula y alternativa:

$$\begin{cases} H_0 : \text{Los datos se distribuyen de forma independiente.}, \\ H_1 : \text{Los datos no se distribuyen de forma independiente}, \end{cases}$$

Como hemos obtenido un valor muy grande de p-valor, no podemos rechazar la hipótesis nula y todo apunta a que los residuos son independientes.

Ahora veamos la normalidad de los residuos:

```r
jarque.bera.test(modelo$residuals)
```

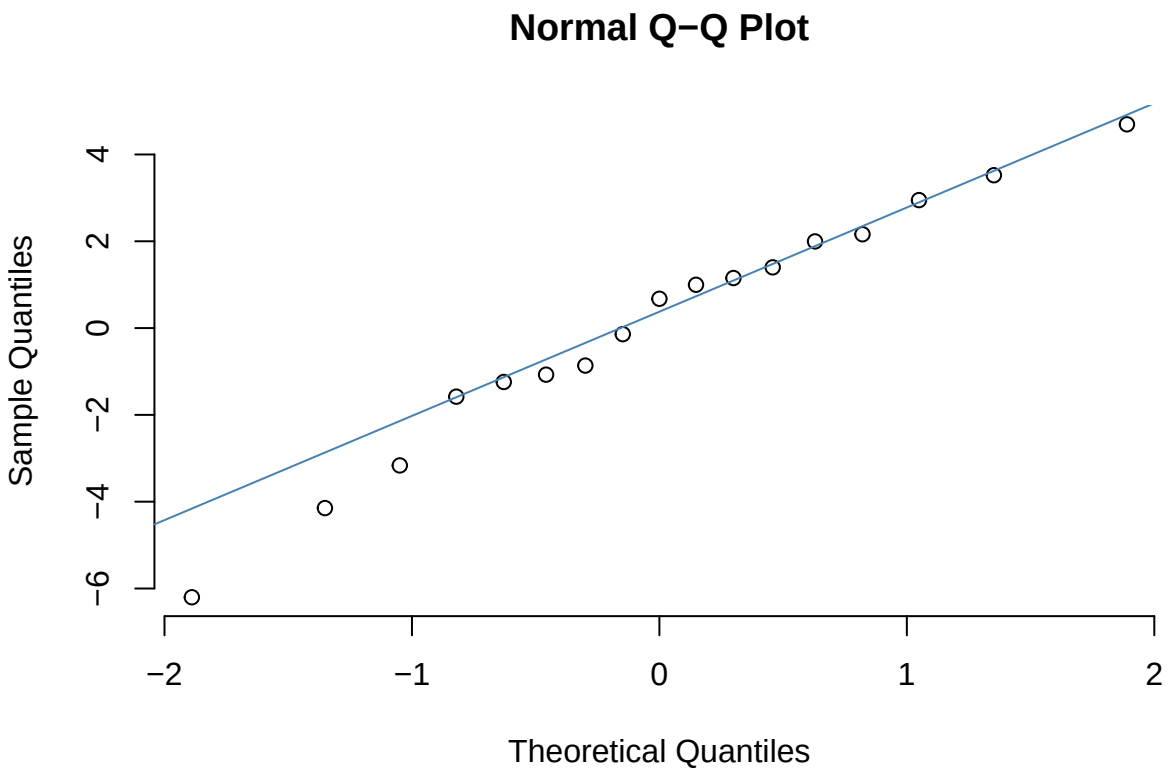
```
##
##  Jarque Bera Test
##
## data:  modelo$residuals
## X-squared = 0.76314, df = 2, p-value = 0.6828
```

```r
shapiro.test(modelo$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  modelo$residuals
## W = 0.9748, p-value = 0.8957
```

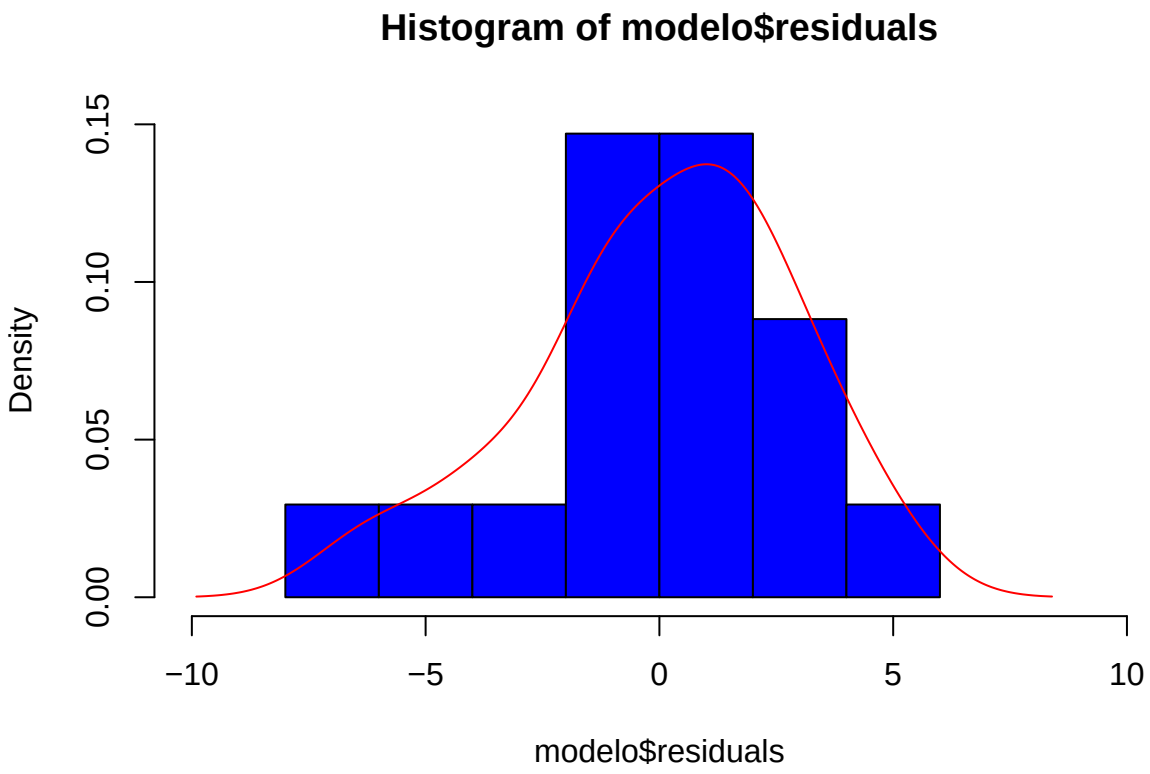
En ambos no hay evidencias estadísticas significativas para rechar la hipótesis de normalidad, por lo que podremos suponer la normalidad de los residuos. Se puede observar mediante un qqplot como los valores se apróximan a los de una normal.

```r
qqnorm(modelo$residuals, pch = 1, frame = FALSE)
qqline(modelo$residuals, col = "steelblue", lwd = 1)
```



También podemos plotear la función de densidad muestral mediante un histograma para ver como se asemaja a una Gaussiana.

```
hist(modelo$residuals, col = "blue", probability = TRUE,
     xlim = c(-10, 10), ylim = c(0, 0.15))
lines(density(modelo$residuals), col = "red")
```



3.1.7. Predicción de nuevos valores.

Una vez ya tengamos ciertas evidencias estadísticas de que nuestro modelo es bueno, usaremos el mismo modelo para en este caso para el total de los datos.

```
tiempo = 1:length(serie)

modelo = arima(serie, order = c(3, 0, 3))
valores_ajustados = modelo$residuals + serie
predicciones = predict(modelo, n.ahead = N_pred)
valores_predichos = predicciones$pred

valores_predichos
```

```
##          Feb      Mar      Apr
## 2018 18.15558 22.86994 29.15156
```

Observamos como los datos van aumentando, es una buena señal ya que en los meses se produce un cambio de estación de invierno a primavera.

3.2. Serie Diaria.

3.2.1. Preprocesamiento.

Usaremos nuestra dataframe datos_serie donde ya habíamos realizado distinttas técnicas de preprocesamiento, eliminación de instancias, selección de variables e imputación por knn. En esta ocasión queremos una serie diaría por lo que es combeniente añadir una nueva columna que codifique el día del mes en el que se tomó esa observación.

```
dias = substring(as.character(datos$Fecha),9,10)


datos_serie = data.frame(Fecha = datos$Fecha, Tmax = imputados$xImp[,1],
                         Mes = as.numeric(meses), Year = as.numeric(years),
                         Dia = as.numeric(dias))

head(datos_serie)
```

```
##         Fecha Tmax Mes Year Dia
## 3  2013-08-12 35.6   8 2013  12
## 4  2013-08-13 34.3   8 2013  13
## 5  2013-08-14 33.9   8 2013  14
## 8  2013-08-17 34.9   8 2013  17
## 9  2013-08-18 35.7   8 2013  18
## 10 2013-08-19 37.2   8 2013  19
```

Como ya comentamos anteriormente hay muchos años y meses donde no se disponen de datos, sin embargo a partir de julio de 2016 se encuentran todos los registros. Tomaremos por tanto los datos a partir de esa fecha.

```
datos_serie = datos_serie[which(datos_serie$Year >= 2016),]
datos_serie = datos_serie[21:507,]
head(datos_serie)
```

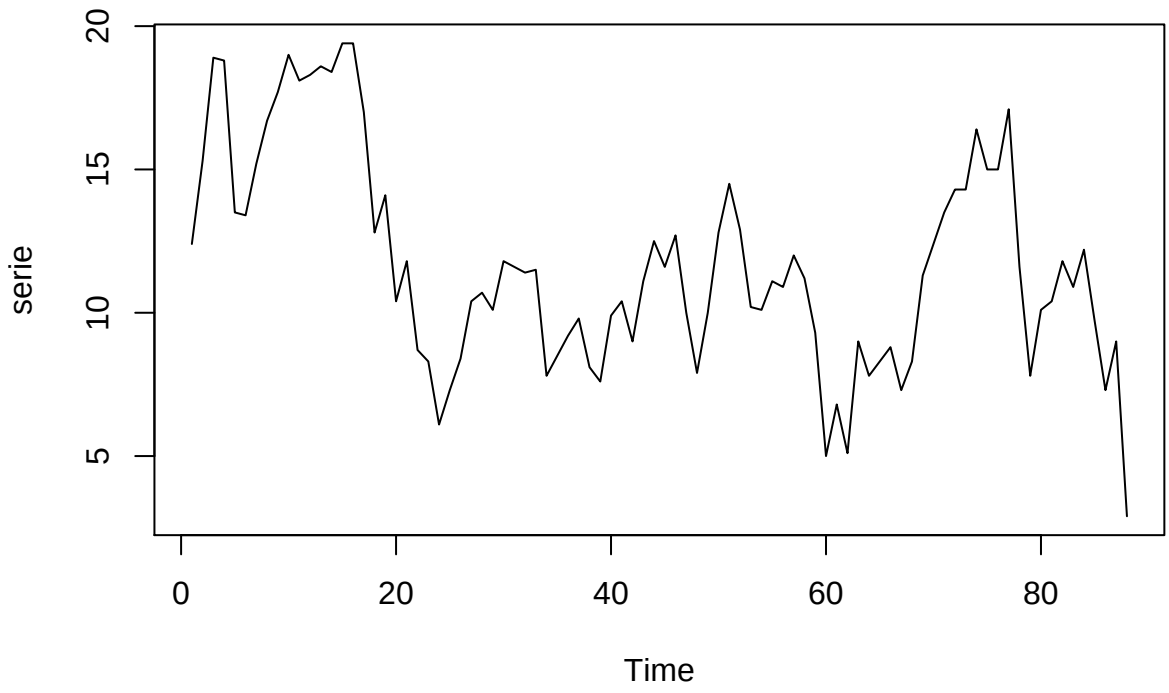
```
##          Fecha Tmax Mes Year Dia
## 422 2016-07-12 34.7   7 2016  12
## 423 2016-07-14 31.6   7 2016  14
## 424 2016-07-15 30.9   7 2016  15
## 425 2016-07-16 31.6   7 2016  16
## 426 2016-07-17 33.9   7 2016  17
## 427 2016-07-18 35.0   7 2016  18
```

```
tail(datos_serie)
```

```
##          Fecha Tmax Mes Year Dia
## 978 2018-01-30 10.9   1 2018  30
## 979 2018-01-31 12.2   1 2018  31
## 980 2018-02-01  9.7   2 2018   1
## 981 2018-02-02  7.3   2 2018   2
## 982 2018-02-03  9.0   2 2018   3
## 983 2018-02-04  2.9   2 2018   4
```

Como tenemos muchos datos para esta serie y nos centraremos en los últimos 100 valores, serán más que suficientes para conseguir una buena predicción. Podemos echar un vistazo a la forma funcional de nuestra serie.

```
serie = ts(datos_serie$Tmax[400:length(datos_serie$Tmax)])
plot(serie)
```



3.2.2. Construcción Conjuntos Test y Train.

Al igual que hicimos para la otra serie construiremos conjuntos de entrenamiento y test para poder tener algunas evidencias de si nuestro modelo es bueno y se asemeja a la realidad. En este caso dejaremos 31 valores para test ya que de esa forma los últimos 7 valores corresponden a la primera semana de marzo que son las temperaturas que queremos inferir.

```
#Número de valores que queremos inferir
N_pred = 31
#Número de valores destinados al test
N_test = 31

#Conjunto Train
serie_train = serie[1:(length(serie) - 30)]
tiempo_train = 1:length(serie_train)

#Conjunto Test
serie_test = serie[(length(serie_train)):length(serie)]
tiempo_test = (length(tiempo_train)):(length(tiempo_train) + 30)
```

3.2.3. Modelización Estacionaridad.

En la descomposición aditiva, se trabaja con la hipótesis de que nuestra serie se descompone como suma de tres series: T_t, S_t y E_t. Si nos fijamos en nuestra serie solo tenemos datos de un año y medio, se puede intuir una estacionalidad anual pero disponemos de pocos datos para poder hacer esa descomposición de la serie estacional, ya que no vemos un patrón de repeteción de al menos dos veces. Respecto a la tendencia se podría suponer que una tendencia a aumentar los valores extremos de las temperaturas, teniendo inviernos más frios

y veranos más calurosos, todo ello debido a circustancias del cambio climático. Sin embargo tenemos pocos datos para poder observar algún tipo de tendencia en nuestra serie. Por tanto para este caso vamos a asumir que nuestra serie solo presenta la parte irregular que debe ser estacionaria para poder ser modelizada mediante técnicas ARIMA. Hay diferencias significativas respecto la serie mensual, la principal es la irregularidad de la serie. En un mismo mes puede haber muchos cambios de temperatura y por eso se observan tantos picos, esto no quiere decir que haya un patrón de estacionalidad.

Primero someteremos la serie al test de Dick Fuller aumentado.

```
adftest = adf.test(serie)
adftest
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  serie
## Dickey-Fuller = -2.1941, Lag order = 4, p-value = 0.4963
## alternative hypothesis: stationary
```

Otenemos un p-valor grande y no podemos rechazar la hipótesis nula y aceptar la estacionaridad de nuestra serie. Diferenciaremos la serie hasta que esta sea estacionaria.

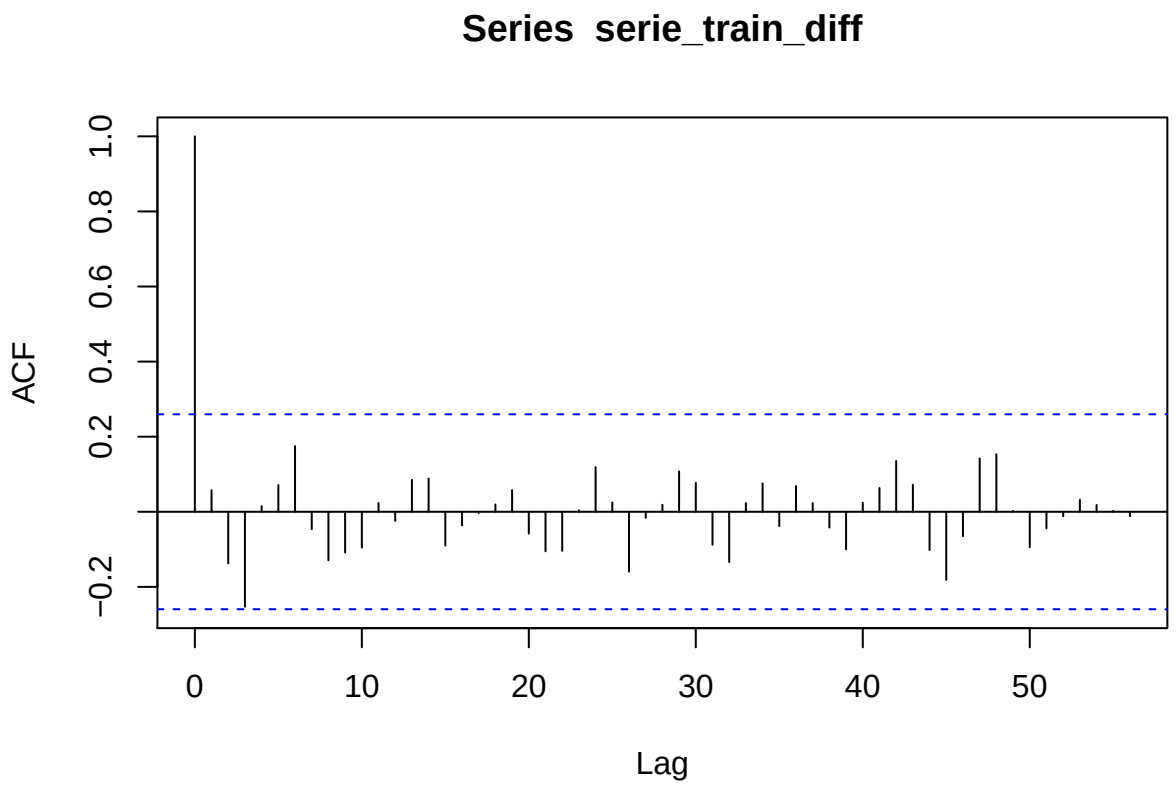
```
serie_train_diff = diff(serie_train)
serie_test_diff = diff(serie_test)

#Aplicamos de nuevo el test
adftest = adf.test(serie_train_diff)
adftest
```

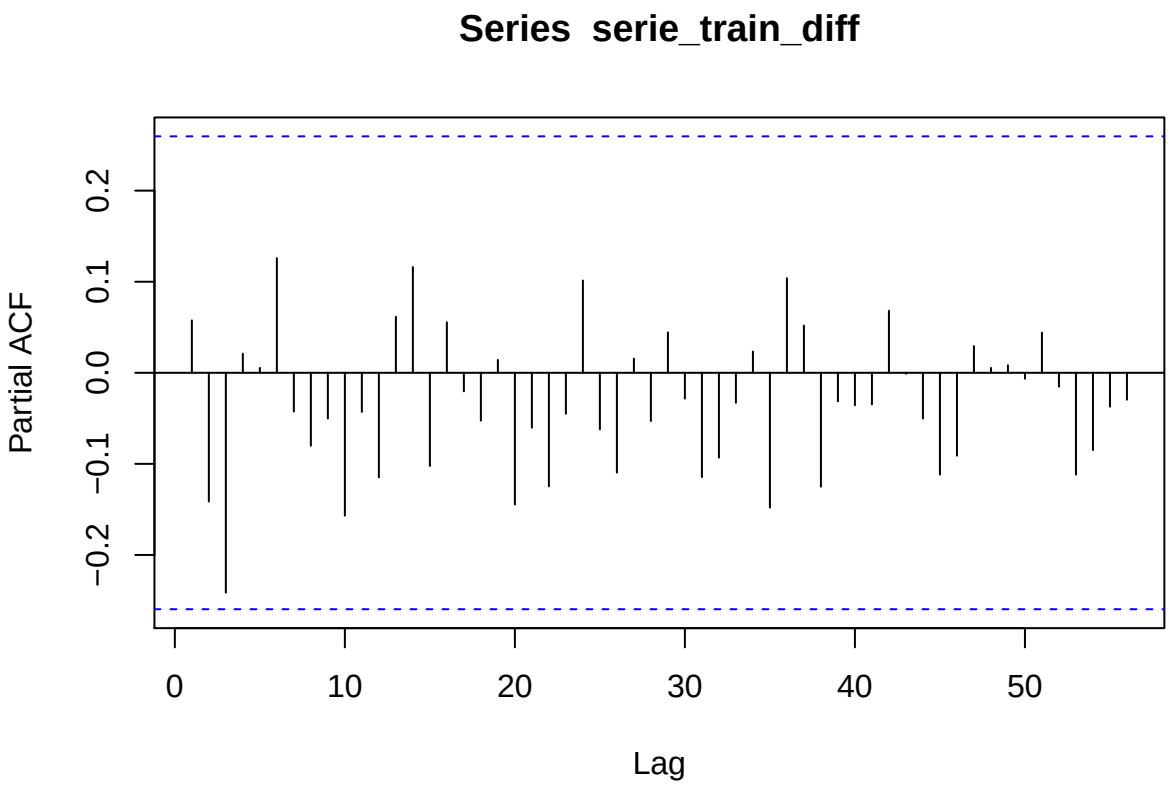
```
##
##  Augmented Dickey-Fuller Test
##
## data:  serie_train_diff
## Dickey-Fuller = -4.1204, Lag order = 3, p-value = 0.01071
## alternative hypothesis: stationary
```

Ahora si tenemos una serie estacionaria. Visualizamos las gráficas de las de funciones de autocorrelación parcial y acumulada. Con ellas podremos deducir que parámetros de los modelo ARIMA pueden ser lo mejores para nuestro problema.

```
acf(serie_train_diff, lag.max = 100)
```




```
pacf(serie_train_diff, lag.max = 100)
```



Ambas gráficas tienen valores significativamente cercanos a cero. Esto significa que los valores de la serie no están correlacionados con los valores en instantes anteriores. Teniendo en cuenta que la forma funcional que generan los modelos autoregresivos y el de medias móviles, haremos distintos experimentos para encontrar unos buenos parámetros para nuestro problema.

Debemos de tener en cuenta un parámetro de modelo autoregresivo lo suficientemente alto para que se consideren las fluctiaciones ascendentes y descendentes de la serie.

```
#Ajustamos modelo
modelo = arima(serie_train, order = c(20, 1, 0))
valores_ajustados = serie_train + modelo$residuals

#Predecimos Valores
predicciones = predict(modelo, n.ahead = N_pred)
valores_predichos = predicciones$pred

valores_predichos
```

```
## Time Series:
## Start = 59
## End = 89
## Frequency = 1
##  [1]  9.676611  8.665638  7.881150  8.762190 10.172738 10.598431 10.206130
##  [8]  8.686947  8.459727 10.154657 11.093113 10.823746 10.276146  9.394499
## [15]  9.511628  9.989402 10.087848 10.472763 10.266389  9.997516  9.918986
## [22]  9.905838 10.410025 11.009066 10.723644 10.208385  9.885676  9.992259
## [29] 10.475977 10.418709 10.052416
```

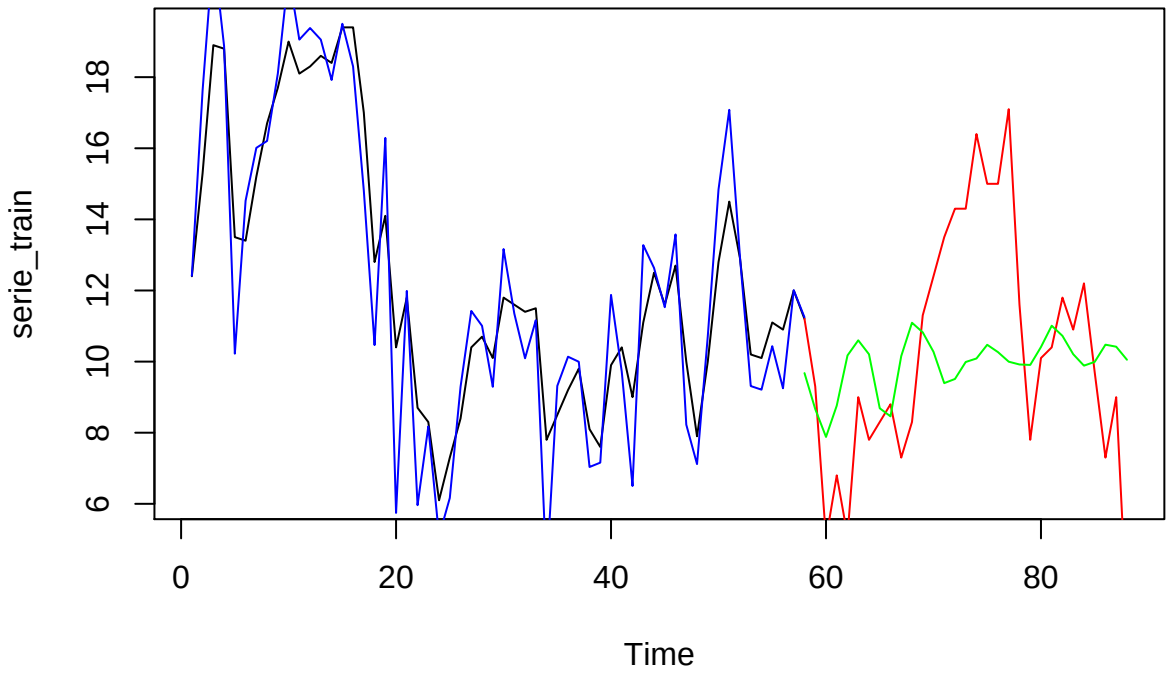
```
serie_test
```

```
##  [1] 11.2  9.3  5.0  6.8  5.1  9.0  7.8  8.3  8.8  7.3  8.3 11.3 12.4 13.5 14.3
## [16] 14.3 16.4 15.0 15.0 17.1 11.6  7.8 10.1 10.4 11.8 10.9 12.2  9.7  7.3  9.0
```

```
## [31]  2.9
```

```
#Error cuadrático acumulado para Train y Test
error_train = sum((modelo$residuals)^2)
error_test = sum((valores_predichos - serie_test)^2)

plot.ts(serie_train, xlim = c(1, tiempo_test[length(tiempo_test)]))
lines(valores_ajustados, col = "blue")
lines(tiempo_test, serie_test, col = "red")
lines(tiempo_test, valores_predichos, col = "green")
```



Vemos en color verde que nuestras predicciones han conseguido obtener ese comportamiento no monotónico. Queremos que esas oscilaciones de monotocidad aumente el rango de sus valores, para ello haremos uso de los modelos de medias móviles. No aumentamos demasiado el parámetro q para evitar sobreaprendizaje.

```
#Ajustamos modelo
modelo = arima(serie_train, order = c(20, 1, 10))
valores_ajustados = serie_train + modelo$residuals

#Predecimos Valores
predicciones = predict(modelo, n.ahead = N_pred)
valores_predichos = predicciones$pred

valores_predichos
```

```
## Time Series:
## Start = 59
## End = 89
## Frequency = 1
```

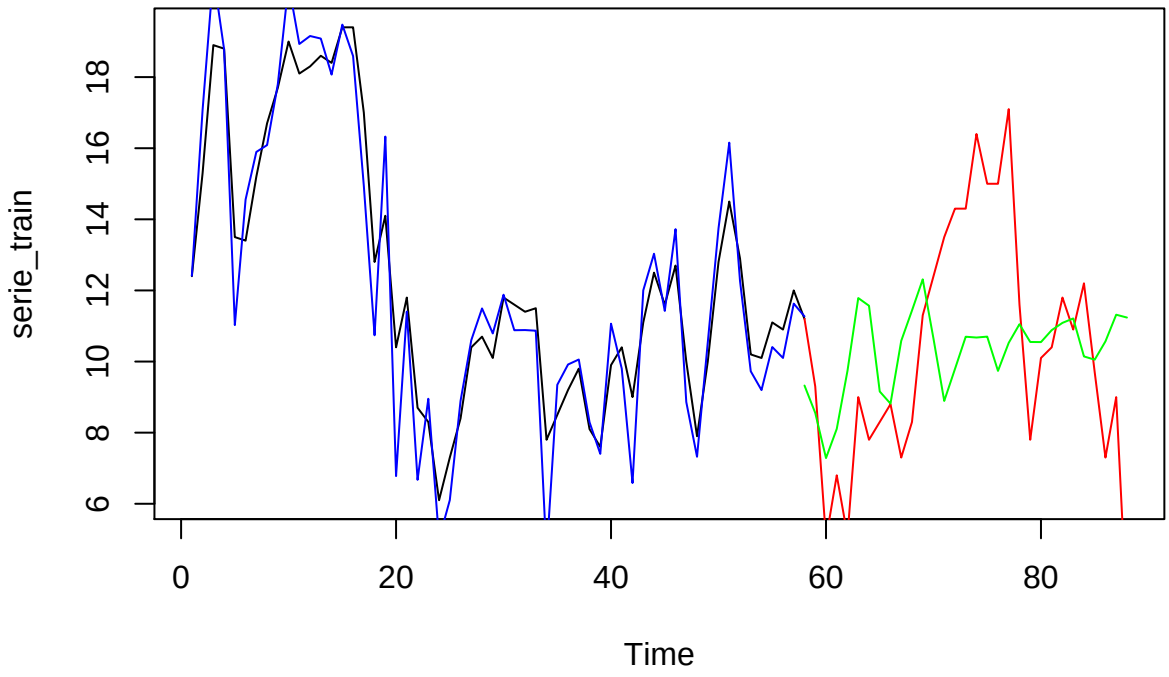
```
##  [1]  9.325315  8.555887  7.285555  8.106968  9.742429 11.786190 11.569386
##  [8]  9.157640  8.816114 10.583378 11.435454 12.312065 10.644816  8.891708
## [15]  9.791188 10.698244 10.675718 10.700745  9.737529 10.526955 11.050526
## [22] 10.550398 10.547003 10.884054 11.087952 11.212061 10.143667 10.051968
## [29] 10.567315 11.317321 11.237335
```

```
serie_test
```

```
##  [1] 11.2  9.3  5.0  6.8  5.1  9.0  7.8  8.3  8.8  7.3  8.3 11.3 12.4 13.5 14.3
## [16] 14.3 16.4 15.0 15.0 17.1 11.6  7.8 10.1 10.4 11.8 10.9 12.2  9.7  7.3  9.0
## [31]  2.9
```

```
#Error cuadrático acumulado para Train y Test
error_train = sum((modelo$residuals)^2)
error_test = sum((valores_predichos - serie_test)^2)

plot.ts(serie_train, xlim = c(1, tiempo_test[length(tiempo_test)]))
lines(valores_ajustados, col = "blue")
lines(tiempo_test, serie_test, col = "red")
lines(tiempo_test, valores_predichos, col = "green")
```



3.2.4. Validación del modelo.

Veamos si los residuos verifican las siguientes propiedades:

- No están correlados, independencia lineal.
- La media de los residuos es cero.

Un modelo que no verifique las anteriores afirmaciones siempre puede ser mejorado. Además estamos interesados que se cumpla que:

- Los residuos tienen varianza constante.
- Los residuos se distribuyen según una normal.

```
#Media aproximadamente cero
mean(modelo$residuals)
```

```
## [1] -0.1242034
```

```
#Test de aleatoridad
Box.test(modelo$residuals)
```

```
##
##  Box-Pierce test
##
## data:  modelo$residuals
## X-squared = 0.0050445, df = 1, p-value = 0.9434
```

Como hemos obtenido un valor muy grande de p-valor, no podemos rechazar la hipótesis nula y todo apunta a que los residuos son independientes.

Ahora veamos la normalidad de los residuos:

```
jarque.bera.test(modelo$residuals)
```

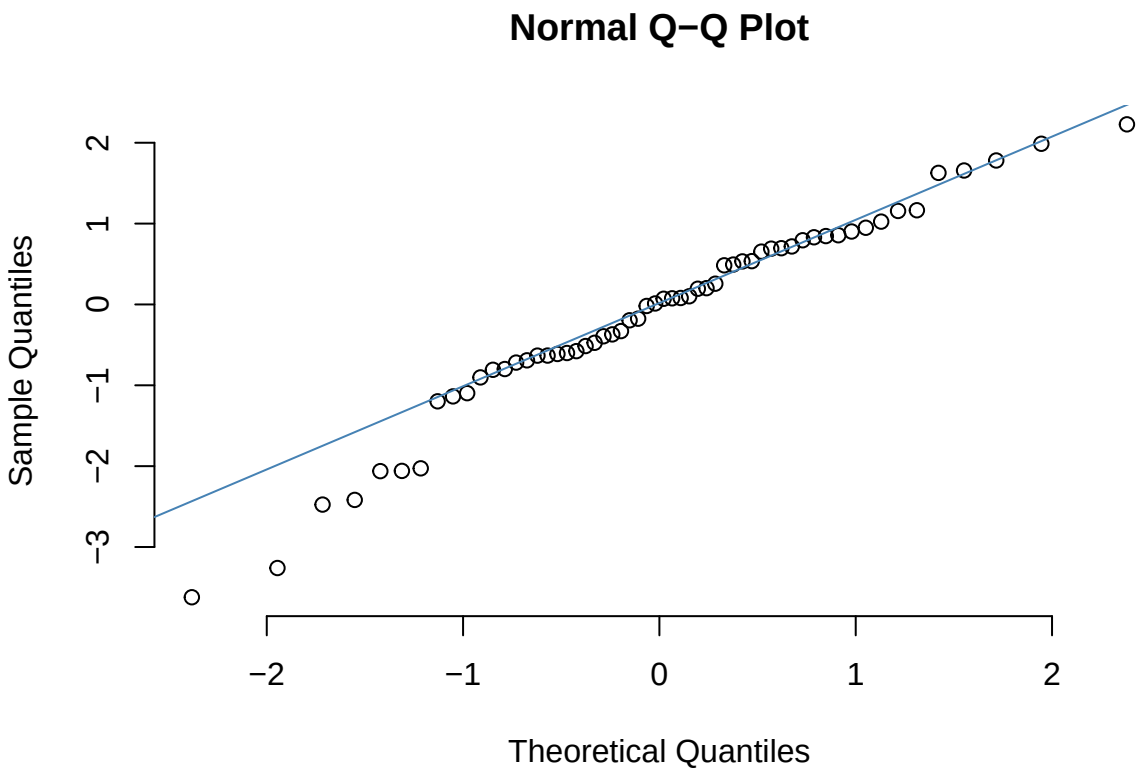
```
##
##  Jarque Bera Test
##
## data:  modelo$residuals
## X-squared = 4.9596, df = 2, p-value = 0.08376
```

```
shapiro.test(modelo$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  modelo$residuals
## W = 0.96146, p-value = 0.06269
```

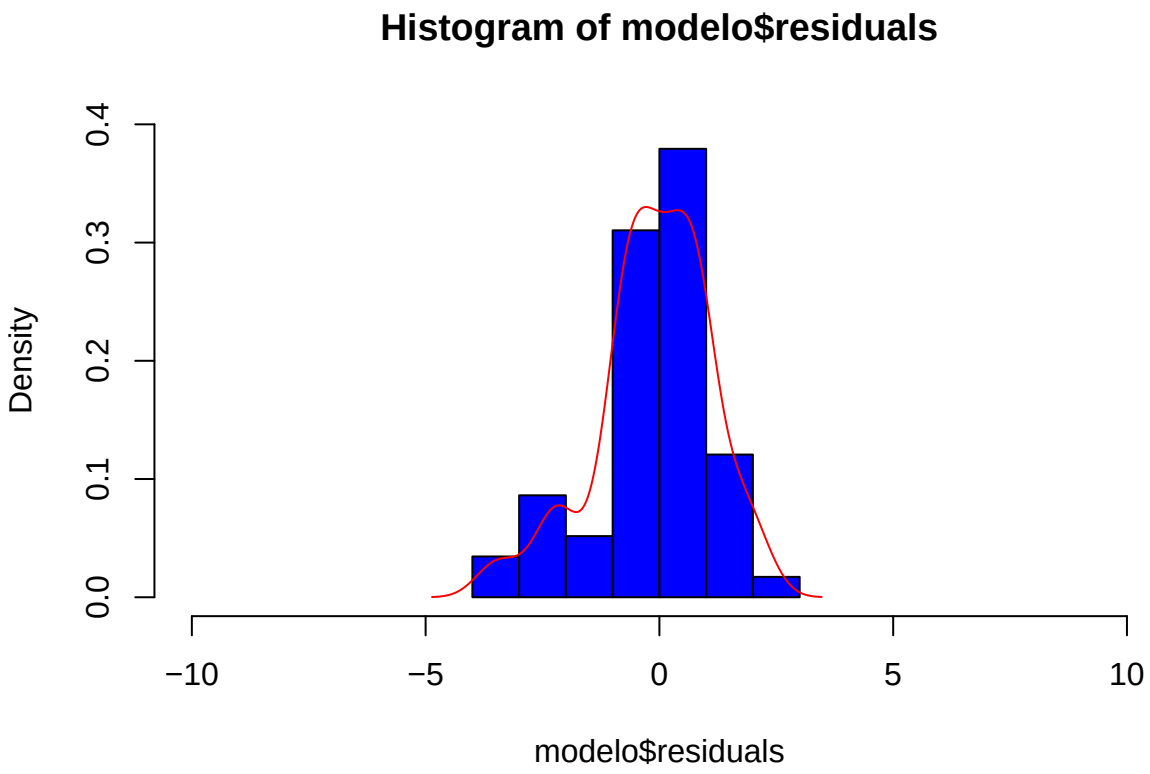
En ambos casos no podemos aceptar la hipótesis de normalidad. Podemos ver si el comportamiento se asemeja a un mediante qqplots y su histograma.

```
qqnorm(modelo$residuals, pch = 1, frame = FALSE)
qqline(modelo$residuals, col = "steelblue", lwd = 1)
```



También podemos plotear la función de densidad muestral, la distribución es uni modal por lo que no sería demasiado locura hacer uso de técnicas paramétricas.

```
hist(modelo$residuals, col = "blue", probability = TRUE,
     xlim = c(-10, 10), ylim = c(0, 0.4))
lines(density(modelo$residuals), col = "red")
```



3.2.5. Predicción de nuevos valores.

Una vez ya tengamos ciertas evidencias estadísticas de que nuestro modelo es bueno, usaremos el mismo modelo para en este caso para el total de los datos.

```
tiempo = 1:length(serie)

modelo = arima(serie, order = c(20, 1, 10))
```

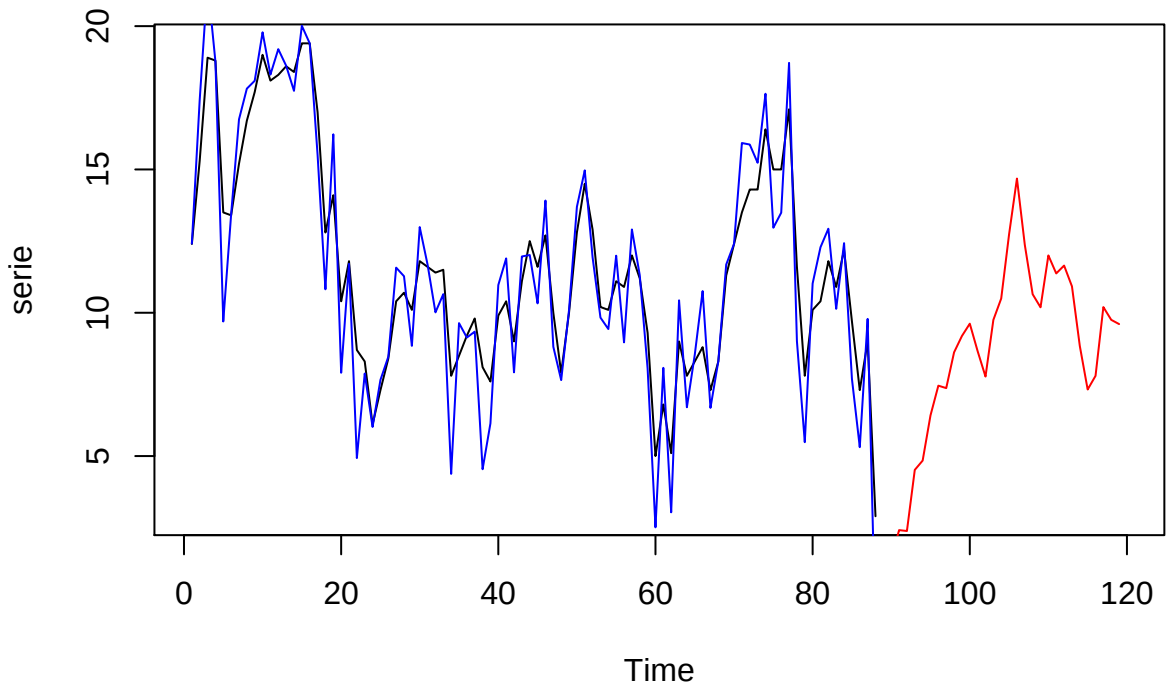
```
## Warning in arima(serie, order = c(20, 1, 10)): possible convergence problem:
## optim gave code = 1
```

```
valores_ajustados = modelo$residuals + serie
predicciones = predict(modelo, n.ahead = N_pred)
valores_predichos = predicciones$pred

valores_predichos
```

```
## Time Series:
## Start = 89
## End = 119
## Frequency = 1
##  [1]  1.056237  1.378115  2.416069  2.385521  4.516651  4.841159  6.418579
##  [8]  7.456539  7.370763  8.618031  9.189785  9.619814  8.659529  7.773048
## [15]  9.751431 10.496434 12.737042 14.683119 12.367652 10.646330 10.189591
## [22] 12.000155 11.371401 11.647021 10.921284  8.843738  7.321956  7.798778
## [29] 10.198201  9.752067  9.604414
```

```
plot.ts(serie,  xlim = c(1, 120))
lines(valores_ajustados, col = "blue")
lines(valores_predichos, col = "red")
```



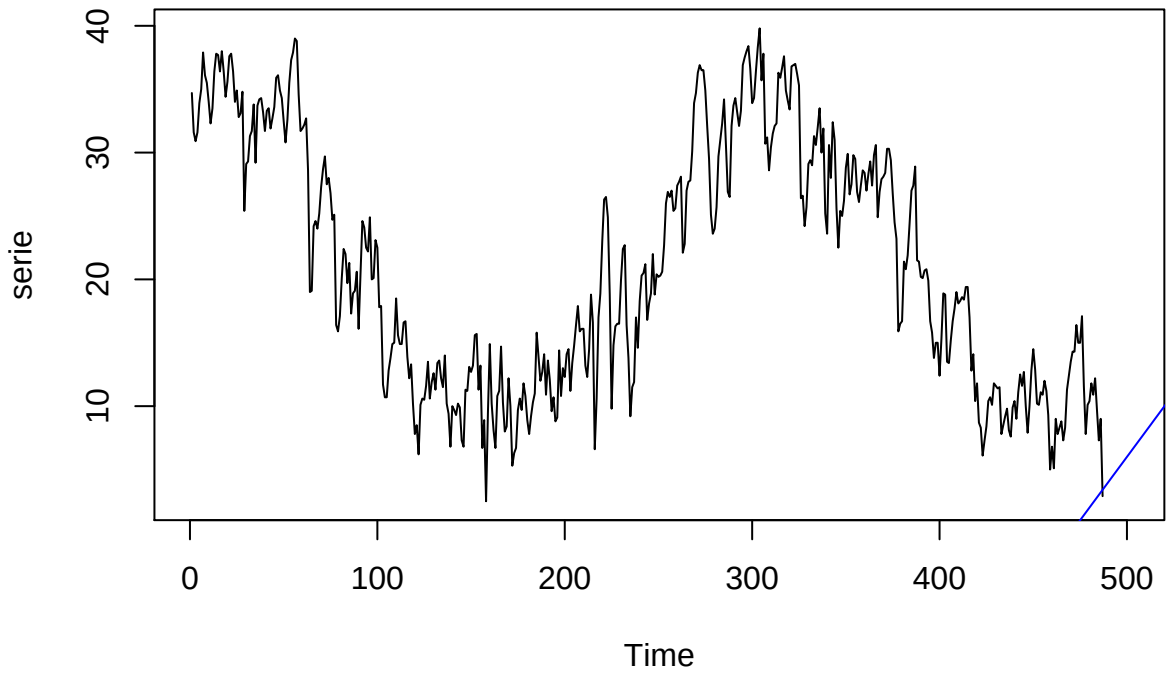
Observamos que los datos van aumentando, buena señal ya que en los días que inferimos se deja atrás el invierno. el mayor problema es que después de un tiempo el modelo vuelve a decrecer, no hemos conseguido obtener esa periodicidad anual. Como hemos comentado en los datos solo tenemos registros de un año y medio, no lo suficiente para poder controlar la estacionalidad anual. Por ello nuestro modelo no es capaz de detectar cuando se produce un cambio de temperatura debido a la estación del año.

Como podemos esperar que la tendencia de las temperaturas sea ascendente, añadiremos artificialmente una tendencia nosotros mismos. Cosideraremos la siguiente recta para modelizar la tendencia.

```
serie = ts(datos_serie$Tmax)
tiempo = 1:length(serie)

# Serie temporal tendencia: modelo lineal
tendencia_recta = tiempo * 0.2

plot.ts(serie, xlim = c(1, 500))
lines(tiempo + 470, tendencia_recta, col = "blue")
```

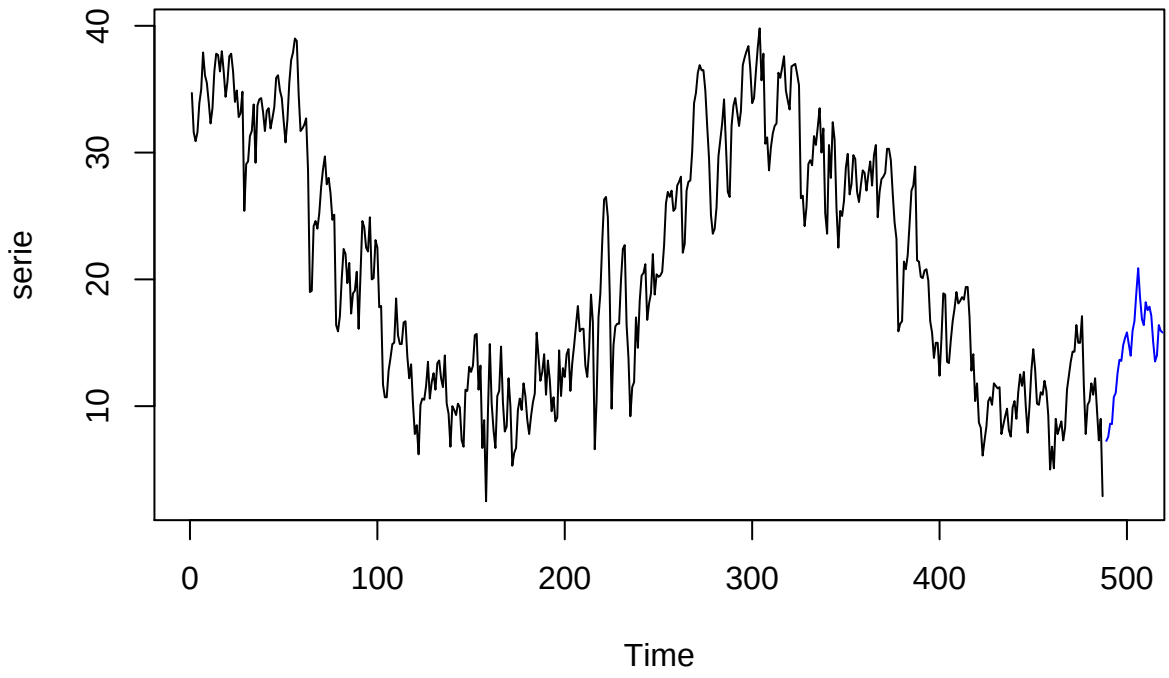


Se observa que añadiendo esta componente a nuestras predicciones obtendremos ese comportamiento ascendente que queríamos, además es lo esperable viendo los datos de toda la serie.

```
valores_finales = valores_predichos + length(valores_predichos) * 0.2
valores_finales
```

```
## Time Series:
## Start = 89
## End = 119
## Frequency = 1
##  [1]  7.256237  7.578115  8.616069  8.585521 10.716651 11.041159 12.618579
##  [8] 13.656539 13.570763 14.818031 15.389785 15.819814 14.859529 13.973048
## [15] 15.951431 16.696434 18.937042 20.883119 18.567652 16.846330 16.389591
## [22] 18.200155 17.571401 17.847021 17.121284 15.043738 13.521956 13.998778
## [29] 16.398201 15.952067 15.804414
```

Finalmente los valores para la primera semana de marzo serían los siguientes: 17.121284 15.043738 13.521956 13.998778 16.398201 15.952067 15.804414.


serie
10
20
30
40
0
100
200
300
400
500
Time