Proyecto Final

Series Temporales

Francisco Martínez

22/12/2022

Contents

1 Descripcion de la serie temporal con tendencia: Máximo precio diario del Bitcoin en USD desde 2017 hasta 2022 3
1.1 Carga de los datos 3
1.2 Representación gráfica de los datos 3
1.3 Análisis descriptivo de la serie temporal 3

2 Descripcion de la serie temporal con tendencia + estacionalidad: Clima diario de la India desde 2013 hasta 2017 5
2.1 Carga de los datos 5
2.2 Representación gráfica de los datos 5
2.3 Análisis descriptivo de la serie temporal 5
2.4 Descomposición de la serie temporal 6

3 Elección del modelo para la serie temporal con tendencia 8
3.1 División del conjunto de datos 8
3.2 Suavizado exponencial simple vs Suavizado exponencial doble 8
3.3 RMSE 8
3.4 RMSE utilizando la suma de errores al cuadrado 8
3.5 MAPE 8
3.6 Ecuación del modelo ajustado 9
3.7 Representamos la serie real frente a la serie ajustada 9
3.8 Calculamos la predicción para h=2 instantes temporales futuros 10
3.9 Representación gráfica de la serie junto a la predicción obtenida 10

4 Elección del modelo para la serie temporal con tendencia + estacionalidad 12
4.1 División del conjunto de datos 12
4.2 Suavizado exponencial doble vs Suavizado exponencial triple 12
4.3 RMSE 12
4.4 RMSE utilizando la suma de errores al cuadrado 12
4.5 MAPE 13
4.6 Ecuación del modelo ajustado 13
4.7 Representamos la serie real frente a la serie ajustada 13
4.8 Calculamos la predicción para h=c instantes temporales futuros 14
4.9 Representación gráfica de la serie junto a la predicción obtenida 14

5 Modelo ARIMA en serie con tendencia 17
5.1 Transformar la serie a un modelo estacionario 17
5.2 Función de autocorrelación y de autocorrelación parcial 18
5.3 Encontrar el modelo ARIMA(p,d,q) 20

5.4 Calculo de la bondad del ajuste 22
5.5 Representación de la serie real vs la ajustada 24
5.6 Predicción 25

6 Modelo ARIMA en serie con tendencia + estacionalidad 27
6.1 Transformar la serie a un modelo estacionario 27
6.2 Función de autocorrelación y de autocorrelación parcial 29
6.3 Encontrar el modelo sARIMA 34
6.4 Calculo de la bondad del ajuste 35
6.5 Representación de la serie real vs la ajustada 36
6.6 Predicción 37

7 Modelo NAR en serie con tendencia 38
7.1 Encontrar el modelo NAR 38
7.2 Cálculo de la bondad de ajuste 39
7.3 Representación de la serie real vs. ajustada 40
7.4 Calculo de la predicción para h=60 instantes temporales futuros 40
7.5 Representación gráfica de la serie junto a la predicción obtenida 41
7.6 Comparación con el mejor modelo ARIMA 43

8 Modelo NAR en serie con tendencia + estacionalidad 44
8.1 Encontrar el modelo NAR 44
8.2 Cálculo de la bondad de ajuste 45
8.3 Representación de la serie real vs. ajustada 46
8.4 Calculo de la predicción para h=c instantes temporales futuros 46
8.5 Representación gráfica de la serie junto a la predicción obtenida 47
8.6 Comparación con el mejor modelo ARIMA 47

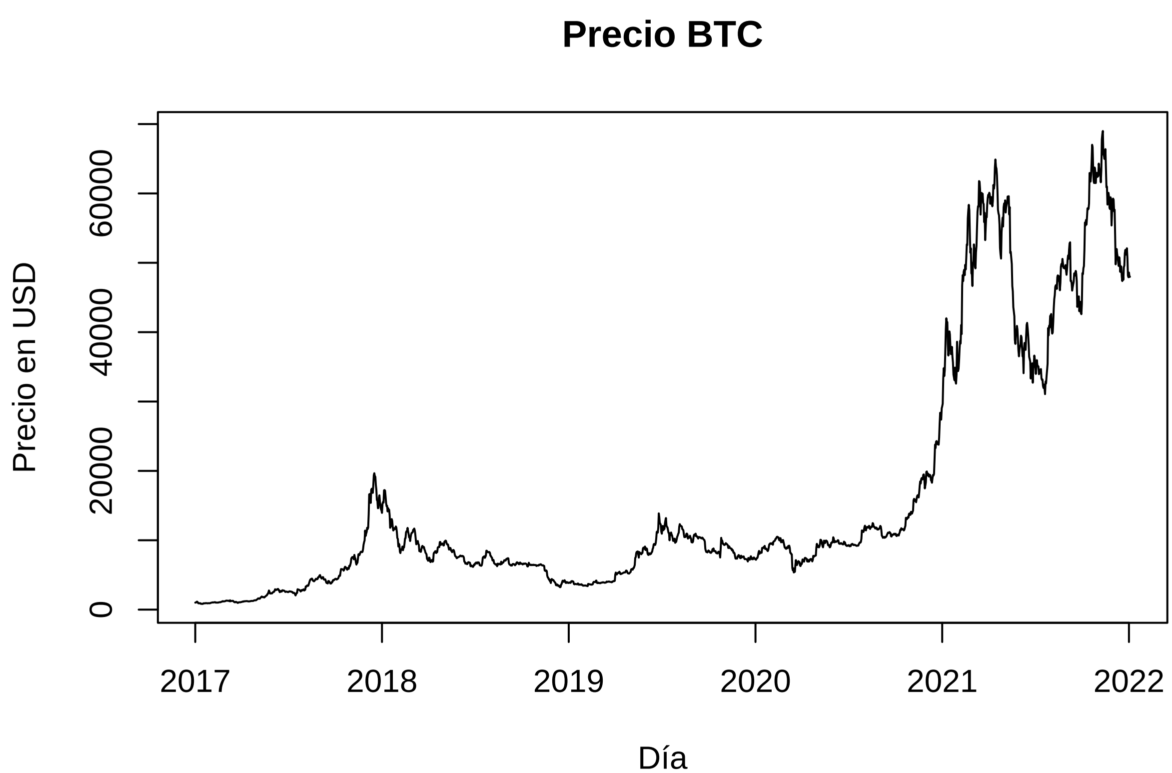
1 Descripcion de la serie temporal con tendencia: Máximo precio diario del Bitcoin en USD desde 2017 hasta 2022

1.1 Carga de los datos

```
btc <- read_csv("BTC-Daily.csv")
btc <- btc[c(2,5)] # nos quedamos con las columnas date y high
btc <- btc[0:1886,] # partimos desde 2017
btc <- btc[with(btc, order(btc$date)), ] # Ordenamos de menor fecha a mayor
```

1.2 Representación gráfica de los datos

```
btc_ts <- ts(btc$high,start=c(2017,1),end=c(2022,3),frequency=365)
plot(btc_ts,xlab="Día",ylab="Precio en USD",main="Precio BTC")
```



1.3 Análisis descriptivo de la serie temporal

```
n <- length(btc$date)
first <- btc[1,]
last <- btc[n,]
btc_head <- btc[with(btc, order(btc$high,decreasing = TRUE)), ]
btc_tail <- btc[with(btc, order(btc$high,decreasing = FALSE)), ]
meda <- mean(btc$high)
medn <- median(btc$high)
```

```
max_min_diff <- btc_head[1,2] - btc_tail[1,2]
time_diff <- last - first
```

Nos encontramos con un conjunto de datos de 1886 observaciones sobre el precio máximo por día del Bitcoin, desde enero de 2017 hasta marzo de 2022.

El valor inicial observado es 1.005$/BTC el 01/01/2017.

El último valor que encontramos es 43.626,49$/BTC el 01/03/2022.

La diferencia entre la primera y la última observacion es de 42.621,49$/BTC.

El máximo valor ha sido de 69.000$/BTC el 10/11/2021.

El menor valor desde 2017 ha sido de 823,45$/BTC el 01/15/2017.

La diferencia entre el máximo y mínimo valor es de 68.176,55$/BTC.

La media de los datos es de 16.750$/BTC a lo largo del tiempo.

La mediana es de 9.019$/BTC.

Observamos picos pronunciados, seguidos de bajadas, en cuatro puntos diferentes:

- Finales de 2017 con una escalada de ~ 20.000$ desde su mínimo local.
- Mitad de 2019 con una escalada de ~ 10.000$ desde su mínimo local.
- Principios de 2021 con una escalada de ~ 50.000$ desde su mínimo local.
- Mitad de 2021 con una escalada de ~ 35.000$ desde su mínimo local.

La tendencia de la serie es positiva.

La tendencia de la serie en los últimos datos de la misma (desde finales de 2021) es decreciente.

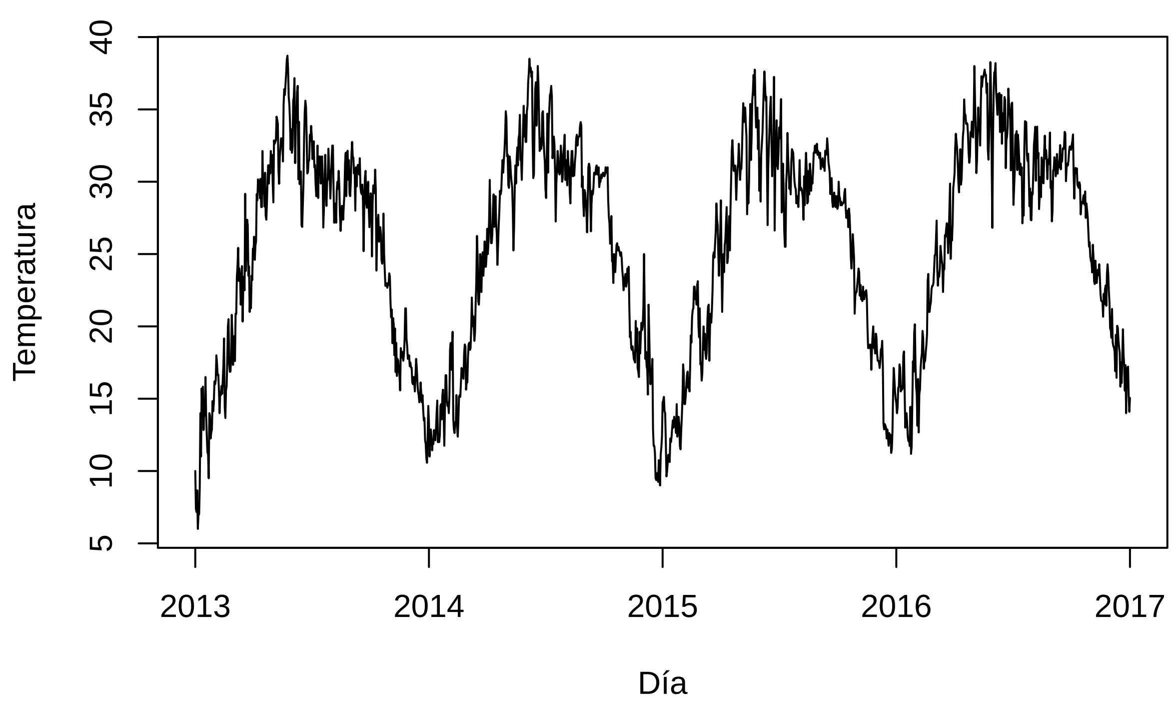
2 Descripcion de la serie temporal con tendencia + estacionalidad: Clima diario de la India desde 2013 hasta 2017

2.1 Carga de los datos

```
clima <- read_csv("DailyDelhiClimateTrain.csv",
    col_types = cols(date = col_date(format = "%Y-%m-%d")))
clima <- clima[c(1,2)] # nos quedamos con la fecha y la temperatura
```

2.2 Representación gráfica de los datos

```
clima_ts <- ts(clima$meantemp,start=c(2013,1),end=c(2017,1),frequency=365)
plot(clima_ts,xlab="Día",ylab="Temperatura")
```



2.3 Análisis descriptivo de la serie temporal

```
n <- length(clima$date)
first <- clima[1,]
last <- clima[n,]
clima_head <- clima[with(clima, order(clima$meantemp,decreasing = TRUE)), ]
clima_tail <- clima[with(clima, order(clima$meantemp,decreasing = FALSE)), ]
meda <- mean(clima$meantemp)
```

Nos encontramos con un conjunto de datos de 1462 observaciones sobre la temperatura media diaria de la India, desde 2013 hasta 2017.

El valor inicial observado es 10ºC el 01/01/2013.

El último valor que encontramos es 10ºC el 01/01/2017.

No hay diferencia entre la primera y la última observacion.

La temperatura máxima ha sido de 38.7ºC el 25/05/2013.

La temperatura mínima sido de 6ºC el 05/01/2013.

La diferencia entre el máximo y mínimo valor es de 32.7ºC.

La temperatura media es de 25.5ºC a lo largo del tiempo.

Observamos picos pronunciados entre los meses de mayo y agosto.

Observamos mínimos entre los meses de diciembre y enero.

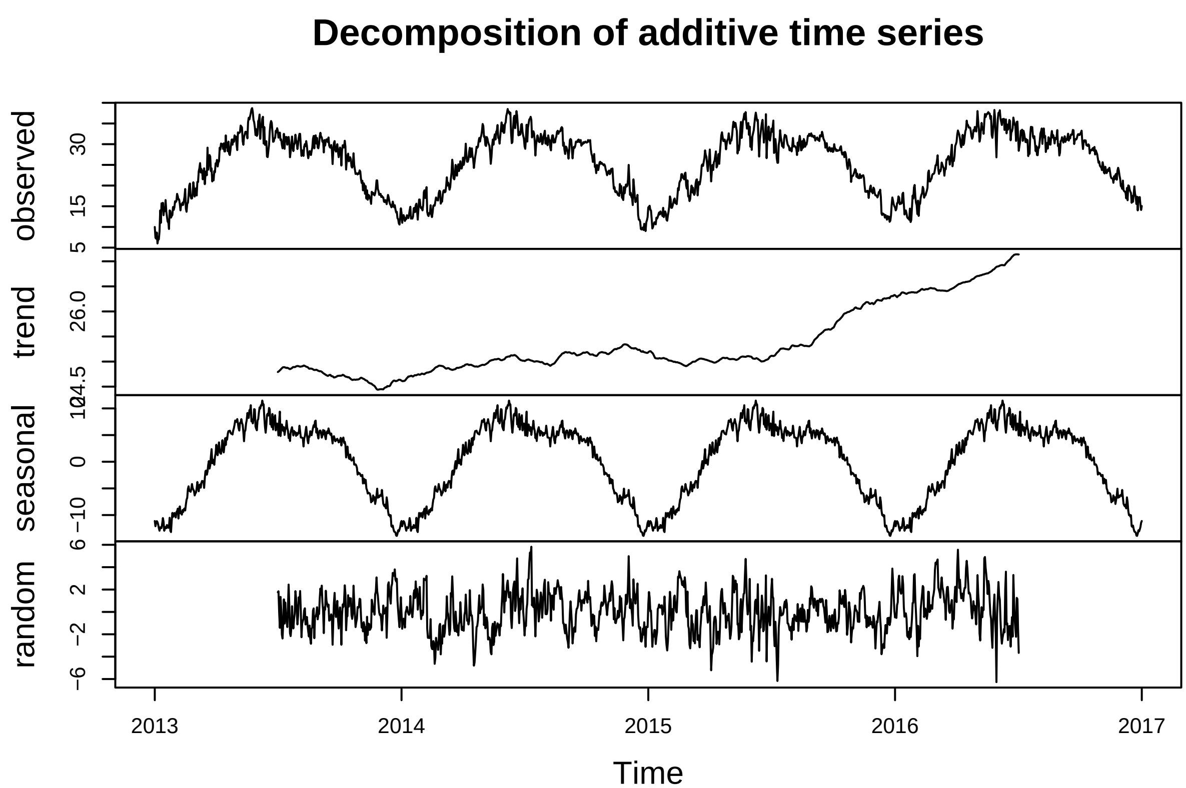
Con lo último podemos concluir que hay estacionalidad.

La longitud de un ciclo estacional es de un año.

2.4 Descomposición de la serie temporal

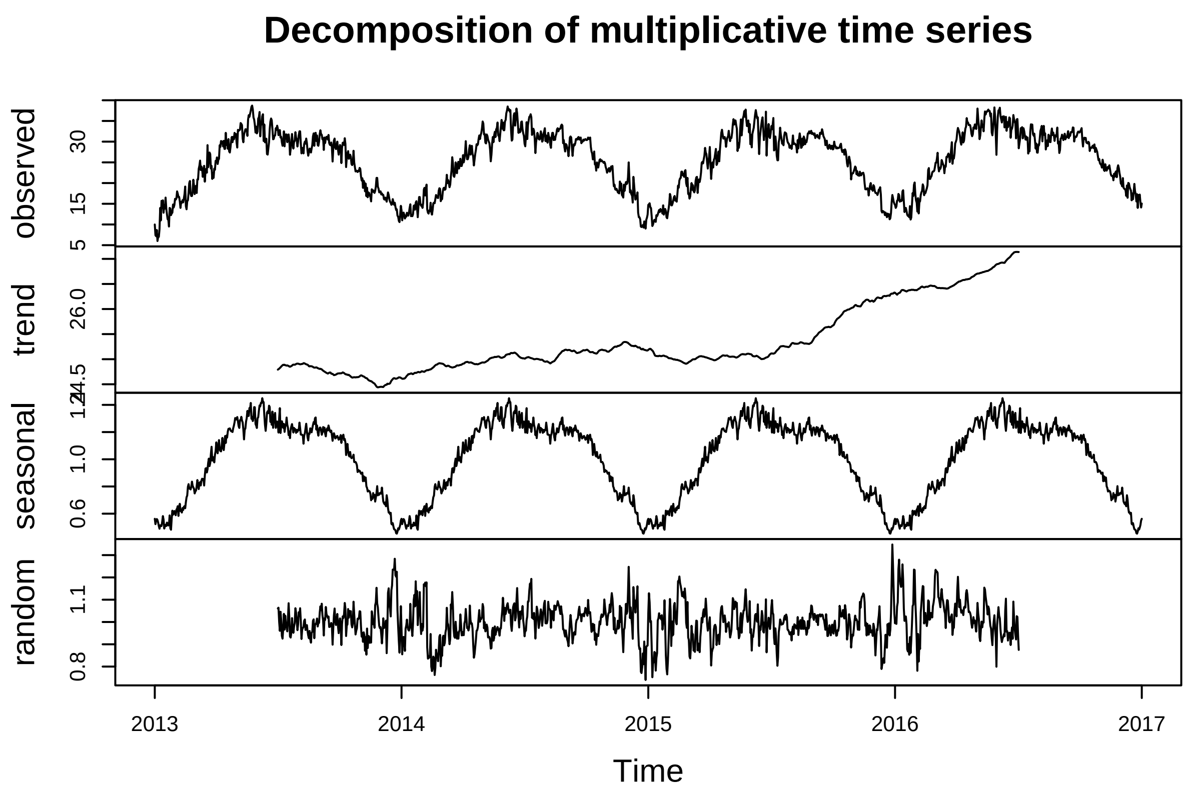
2.4.1 Descomposición Aditiva

```
comp <- decompose(clima_ts, type="additive")
plot(comp)
```



2.4.2 Descomposición Multiplicativa

```
comp <- decompose(clima_ts, type="multiplicative")
plot(comp)
```



En ambos casos observamos que la tendencia es positiva. La estacionalidad asciende, se mantiene en valores altos y vuelve a descender.

En la descomposición aditiva, los errores no tienen outliers, aparecen en una única banda. La varianza se mantiene constante a lo largo del tiempo. Podemos decir que los errores tienen homocedasticidad. Como conclusión decimos que los errores no muestran estructura.

En el caso multiplicativo los errores muestran picos al final e inicio de cada año. Podemos deducir que existen outliers que nuestro modelo no es capaz de interpretar. En este caso no tenemos un ruido blanco constante.

3 Elección del modelo para la serie temporal con tendencia

3.1 División del conjunto de datos

```
insample_btc <- window(btc_ts,start=c(2017,1),end=c(2022,1))   # Ajuste
outsample_btc <- window(btc_ts,start=c(2022,2),end=c(2022,3)) # Predicción
```

3.2 Suavizado exponencial simple vs Suavizado exponencial doble

```
sal_ses_btc <- HoltWinters(insample_btc,beta=FALSE,gamma=FALSE) # Simple
sal_holt_btc <- HoltWinters(insample_btc,gamma=FALSE) # Doble
```

```
fitval_ses_btc <- fitted(sal_ses_btc);
fitval_holt_btc <- fitted(sal_holt_btc);
```

3.3 RMSE

```
rmse_ses_btc <- sqrt(mean((insample_btc[2:n]-fitval_ses_btc[,1])^2)); rmse_ses_btc
```

```
## [1] 20433.73
```

```
rmse_holt_btc <- sqrt(mean((insample_btc[3:n]-fitval_holt_btc[,1])^2)); rmse_holt_btc
```

```
## [1] 20434.79
```

El RMSE del modelo Suavizado exponencial doble es menor, por lo que los datos predichos son más parecidos a los datos reales en este modelos.

3.4 RMSE utilizando la suma de errores al cuadrado

```
sqrt(sal_ses_btc$SSE/length(fitval_ses_btc[,1]))
```

```
## [1] 799.1455
```

```
sqrt(sal_holt_btc$SSE/length(fitval_holt_btc[,1]))
```

```
## [1] 798.9328
```

El RMSE utilizando la suma de errores al cuadrado del modelo Suavizado exponencial doble es menor, por lo que este modelo sigue siendo la mejor opción.

3.5 MAPE

```
mape_ses <- 100*mean(abs(insample_btc[2:n]-fitval_ses_btc[,1])/insample_btc[2:n]);
mape_ses
```

```
## [1] 442.2812
```

```
mape_holt <- 100*mean(abs(insample_btc[3:n]-fitval_holt_btc[,1])/insample_btc[3:n]);
mape_holt
```

```
## [1] 440.3942
```

En los errores porcentuales medios absolutos sigue siendo mejor el modelo de Holt.

Como conclusión tenemos que el modelo Suavizado exponencial doble es mejor opción que el modelo Suavizado exponencial simple.

3.6 Ecuación del modelo ajustado

```
sal_holt_btc$coefficients
```

```
##        a        b
## 48589.47    27.00
```

```
sal_holt_btc$alpha
```

```
## alpha
##     1
```

```
sal_holt_btc$beta
```

```
## beta
##    0
```

$\hat{x} = L_{(t-1)} + T_{(t-1)}$

$L_t = \alpha * x_t + (1 - \alpha) * (L_{(t-1)} + T_{(t-1)})$

$T_t = \beta * (L_t - L_{(t-1)}) + (1 - \beta) * T_{(t-1)}$

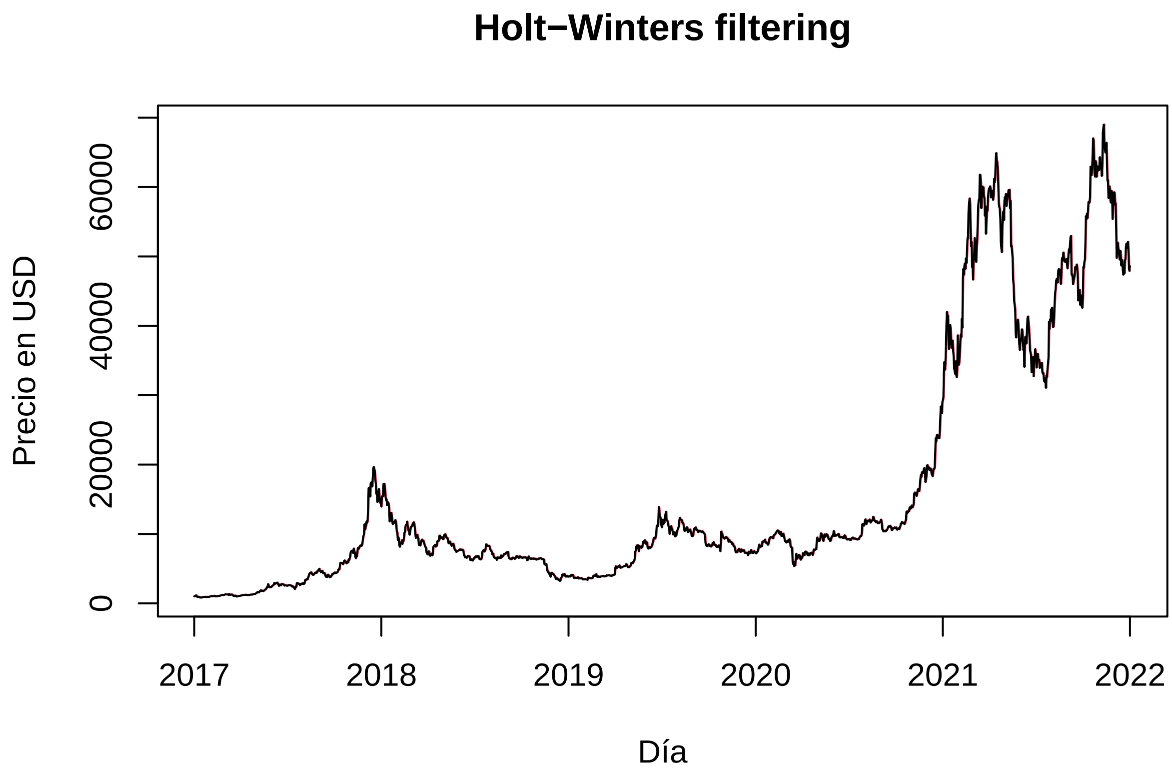
Sustituyendo los valores tenemos:

$L_t = x_t$

$T_t = T_{(t-1)}$

3.7 Representamos la serie real frente a la serie ajustada

```
plot(sal_holt_btc,xlab="Día",ylab="Precio en USD")
```

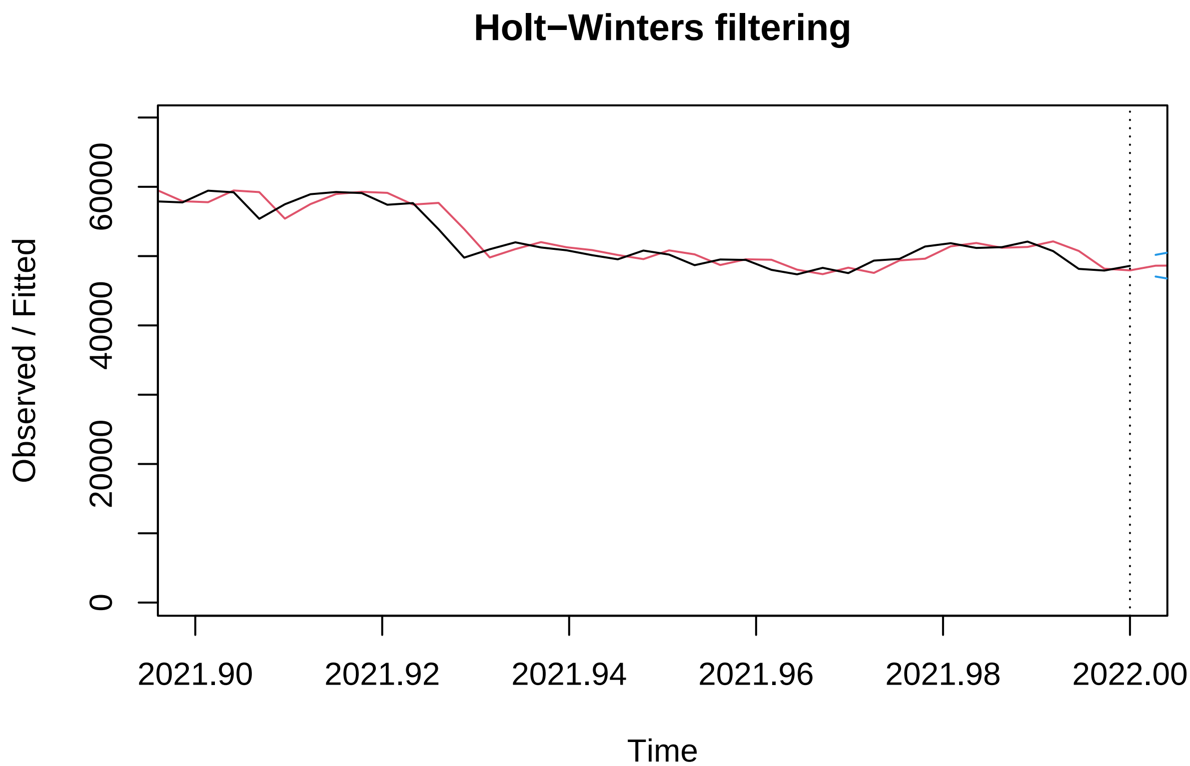


3.8 Calculamos la predicción para h=2 instantes temporales futuros

```
pred_btc_intervalos <- predict(sal_holt_btc,n.ahead=2,prediction.interval=TRUE,level=0.95)
```

3.9 Representación gráfica de la serie junto a la predicción obtenida

```
plot(sal_holt_btc, pred_btc_intervalos,xlim=c(2021.9,2022))
```



Holt−Winters filtering
Observed / Fitted
0
20000
40000
60000
2021.90
2021.92
2021.94
2021.96
2021.98
2022.00
Time

4 Elección del modelo para la serie temporal con tendencia + estacionalidad

4.1 División del conjunto de datos

```
insample_clima <- window(clima_ts,start=c(2013,1),end=c(2015,365))   # Datos
# utilizados para el ajuste
outsample_clima <- window(clima_ts,start=c(2016,1),end=c(2016,365))
# Observaciones reservadas para valorar la predicción
```

4.2 Suavizado exponencial doble vs Suavizado exponencial triple

Comparamos el modelo de Suavizado exponencial doble (modelo de Holt) con el modelo de Suavizado exponencial triple (modelo de Holt-Winters) en sus dos métodos de estacionalidad: aditiva y multiplicativa.

```
sal_holt_clima <- HoltWinters(insample_clima,gamma=FALSE)
sal_hw_clima_add <- HoltWinters(insample_clima,seasonal="additive")
sal_hw_clima_mult <- HoltWinters(insample_clima,seasonal="multiplicative")
```

```
fitval_holt <- fitted(sal_holt_clima);
fitval_hw_add <- fitted(sal_hw_clima_add);
fitval_hw_mult <- fitted(sal_hw_clima_mult);
```

4.3 RMSE

```
insample_clima_2 <- window(insample_clima,start=c(2014,1),end=c(2015,365))
```

```
rmse_holt <- sqrt(mean((insample_clima_2-fitval_holt[,1])^2)); rmse_holt
```

```
## [1] 1.631185
```

```
rmse_hw_add <- sqrt(mean((insample_clima_2-fitval_hw_add[,1])^2)); rmse_hw_add
```

```
## [1] 1.769573
```

```
rmse_hw_mult <- sqrt(mean((insample_clima_2-fitval_hw_mult[,1])^2)); rmse_hw_mult
```

```
## [1] 1.783369
```

El modelo de Suavizado exponencial simple es el que mejor RMSE consigue.

4.4 RMSE utilizando la suma de errores al cuadrado

```
sqrt(sal_holt_clima$SSE/length(fitval_holt[,1]))
```

```
## [1] 1.707426
```

```
sqrt(sal_hw_clima_add$SSE/length(fitval_hw_add[,1]))
```

```
## [1] 1.769573
```

```
sqrt(sal_hw_clima_mult$SSE/length(fitval_hw_mult[,1]))
```

```
## [1] 1.783369
```

El modelo de Suavizado exponencial simple continúa obteniendo el mejor resultado.

4.5 MAPE

```
mape_holt <- 100*mean(abs(insample_clima_2-fitval_holt[,1])/insample_clima_2); mape_holt
```

```
## [1] 5.20086
```

```
mape_hw_add <- 100*mean(abs(insample_clima_2-fitval_hw_add[,1])/insample_clima_2); mape_hw_add
```

```
## [1] 5.134157
```

```
mape_hw_mult <- 100*mean(abs(insample_clima_2-fitval_hw_mult[,1])/insample_clima_2); mape_hw_mult
```

```
## [1] 5.162215
```

En cuanto a los errores porcentuales medios absolutos tenemos que el modelo de Suavizado exponencial triple con estacionalidad aditiva es la mejor opción. Concluimos que el modelo que mejor se ajusta a nuestros datos es nombrado anteriormente.

4.6 Ecuación del modelo ajustado

```
sal_hw_clima_add$alpha
```

```
##     alpha
## 0.6749052
```

```
sal_hw_clima_add$beta
```

```
## beta
##    0
```

```
sal_hw_clima_add$gamma
```

```
## gamma
##     1
```

$\hat{x} = L_{(t-1)} + T_{(t-1)} + S_{(t-c)}$

$L_t = \alpha * (x_t - S_{(t-c)}) + (1 - \alpha) * (L_{(t-1)} + T_{(t-1)})$

$T_t = \beta * (L_t - L_{(t-1)}) + (1 - \beta) * T_{(t-1)}$

$S_t = \gamma * (x_t - L_{(t-1)}) + (1 - \gamma) * S_{(t-c)}$

Sustituyendo los valores tenemos:

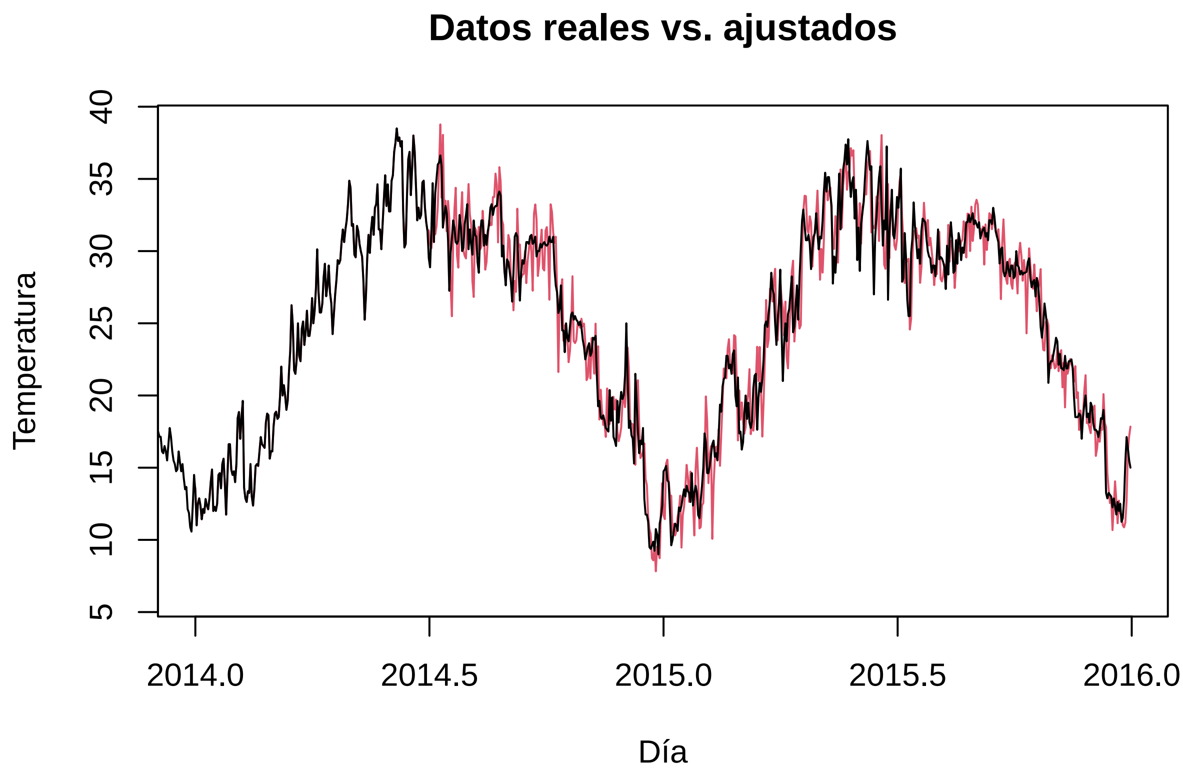
$L_t = 0.7560656 * (x_t - S_{(t-c)}) + (1 - 0.2439344) * (L_(t-1) + T_{(t-1)})$

$T_t = T_{(t-1)}$

$S_t = S_{(t-c)}$

4.7 Representamos la serie real frente a la serie ajustada

```
plot(sal_hw_clima_add,ylab="Temperatura",xlab="Día",main="Datos reales vs. ajustados")
```

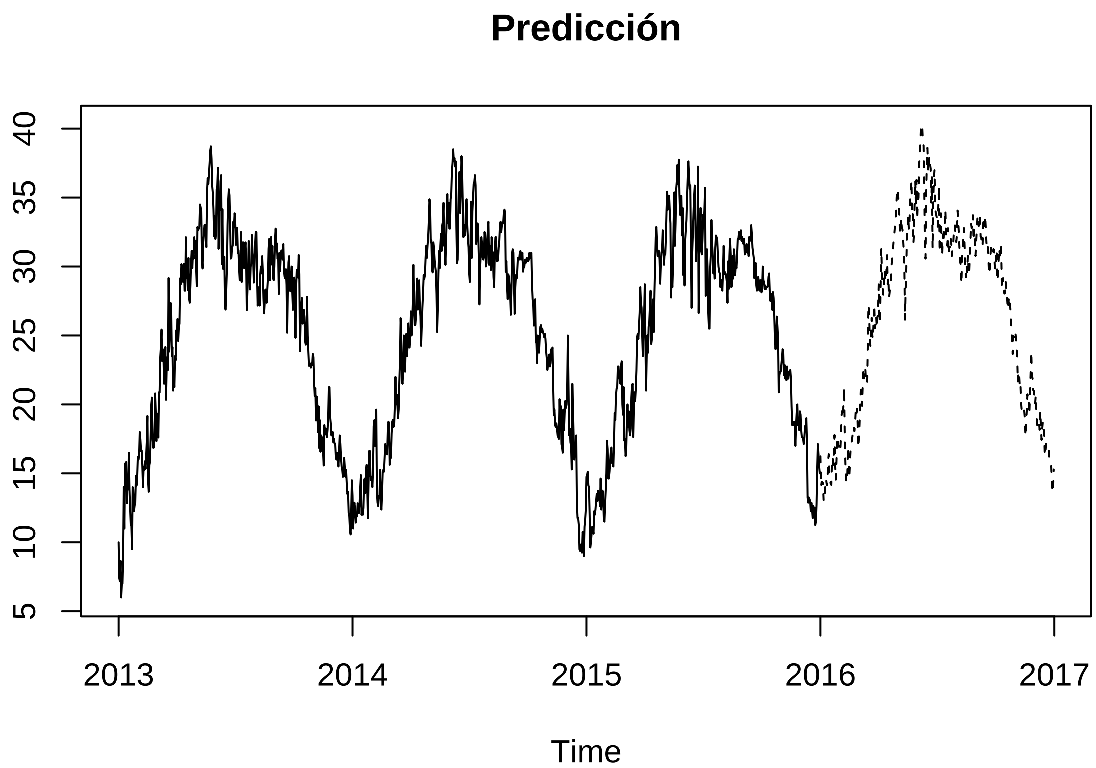


4.8 Calculamos la predicción para h=c instantes temporales futuros

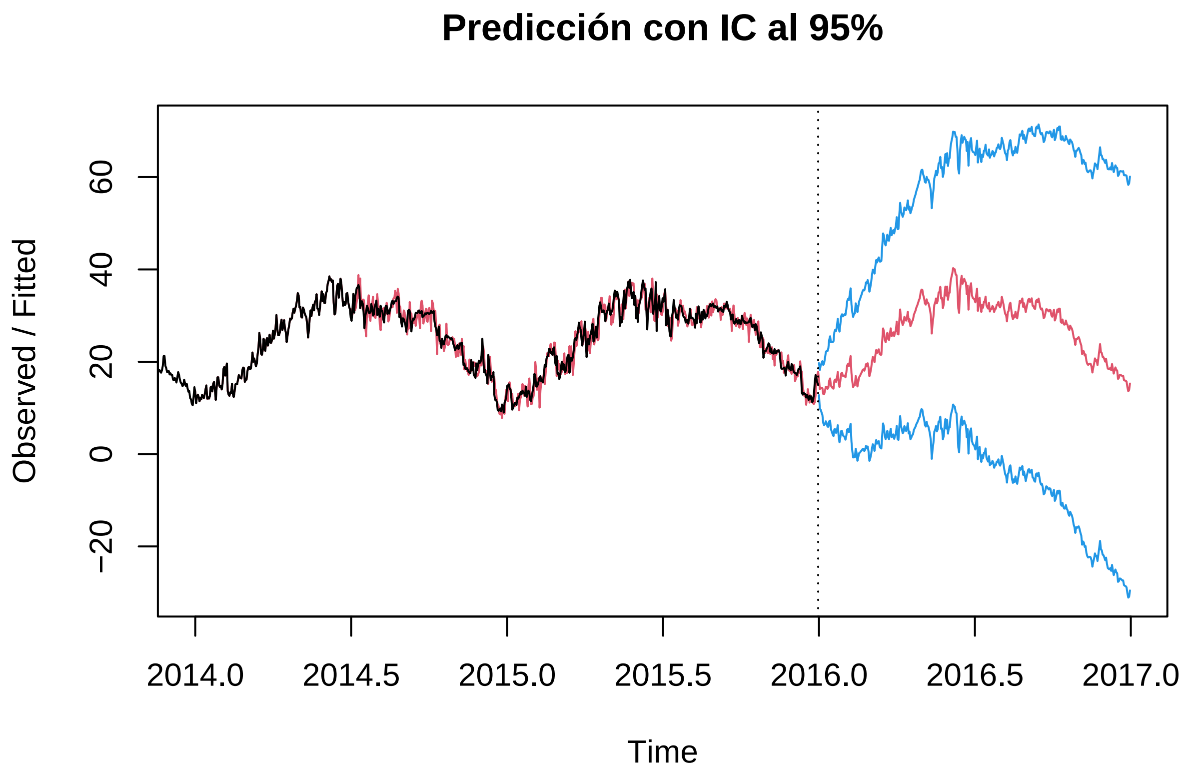
```
pred_clima <- predict(sal_hw_clima_add,365)
pred_clima_intervalos <- predict(sal_hw_clima_add,n.ahead=365,prediction.interval=TRUE,level=0.95)
```

4.9 Representación gráfica de la serie junto a la predicción obtenida

```
ts.plot(insample_clima,pred_clima,lty=1:2,main="Predicción")
```



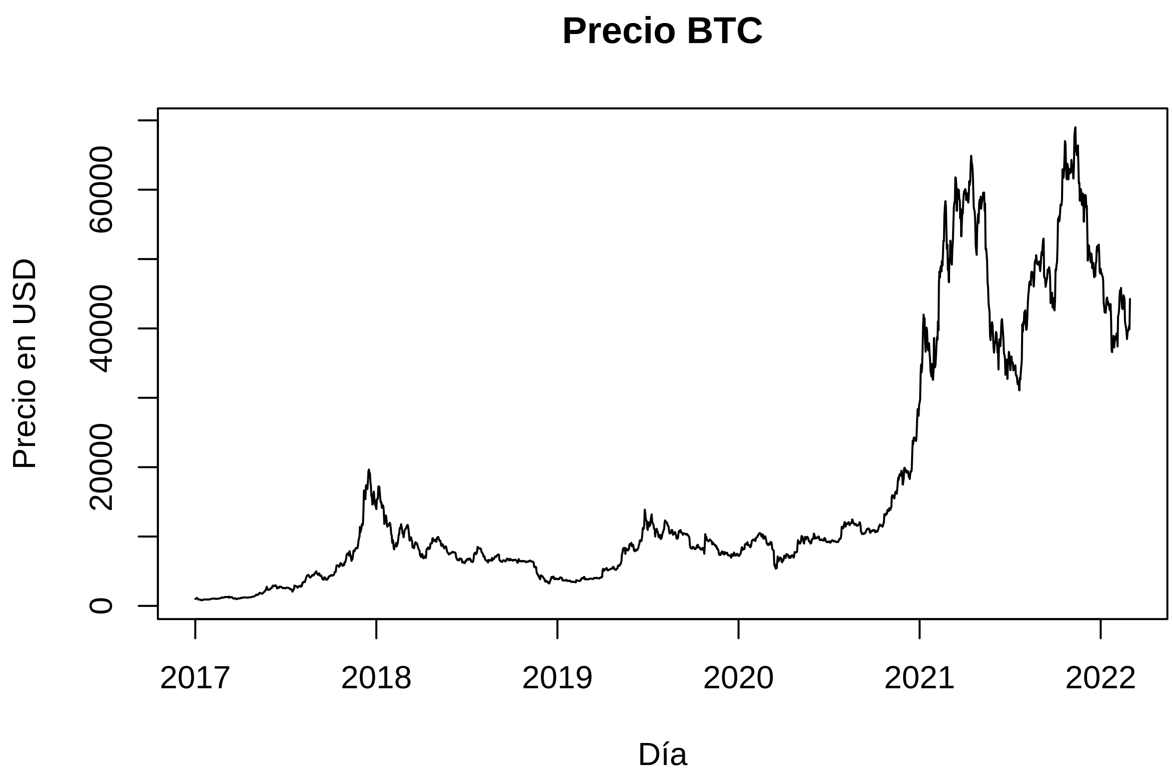

```
plot(sal_hw_clima_add, pred_clima_intervalos,main="Predicción con IC al 95%")
```


Predicción con IC al 95%
Observed / Fitted
−20
0
20
40
60
2014.0
2014.5
2015.0
2015.5
2016.0
2016.5
2017.0
Time

5 Modelo ARIMA en serie con tendencia

5.1 Transformar la serie a un modelo estacionario

```
btc_ts <- ts(btc$high,start=c(2017,1),end=c(2022,60),frequency=365)
plot(btc_ts,xlab="Día",ylab="Precio en USD",main="Precio BTC")
```

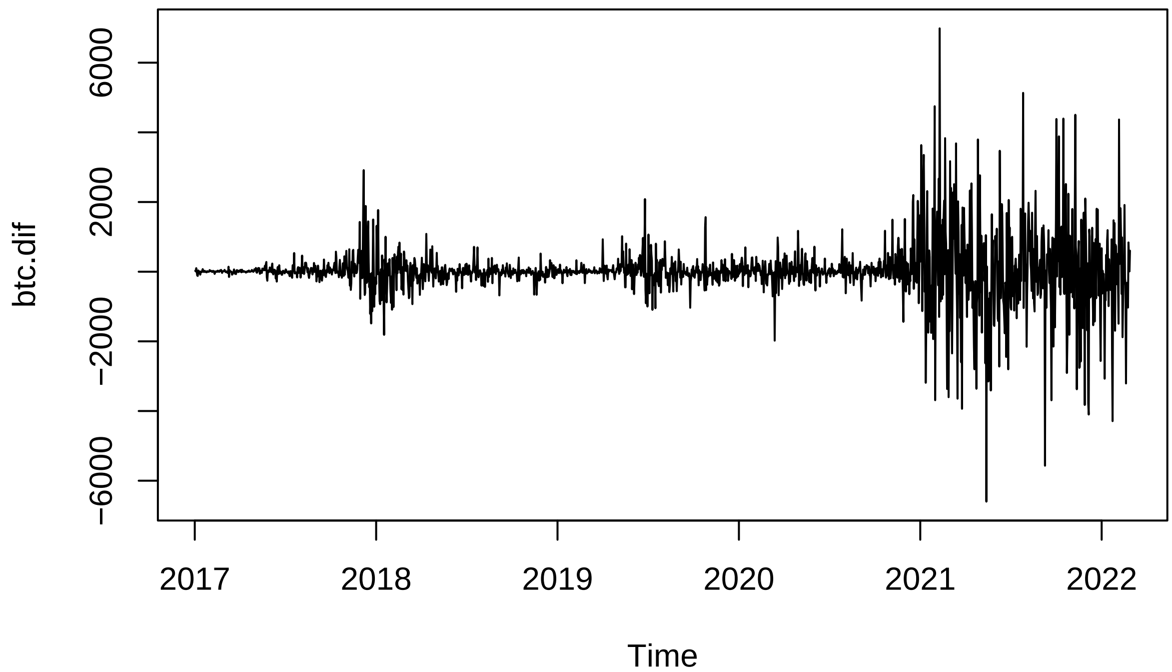



```
outsample <- window(btc_ts,start=c(2022,59),end=c(2022,60))
insample <- window(btc_ts,start=c(2017,1),end=c(2022,58))
```

Como se observa a simple vista nuestra serie no es estacionaria.

Lo que tenemos que hacer a continuación es pasar a proceso estacionario.

```
btc.dif <- diff(insample)
plot(btc.dif,type="l")
```




```
mean(btc.dif)
```

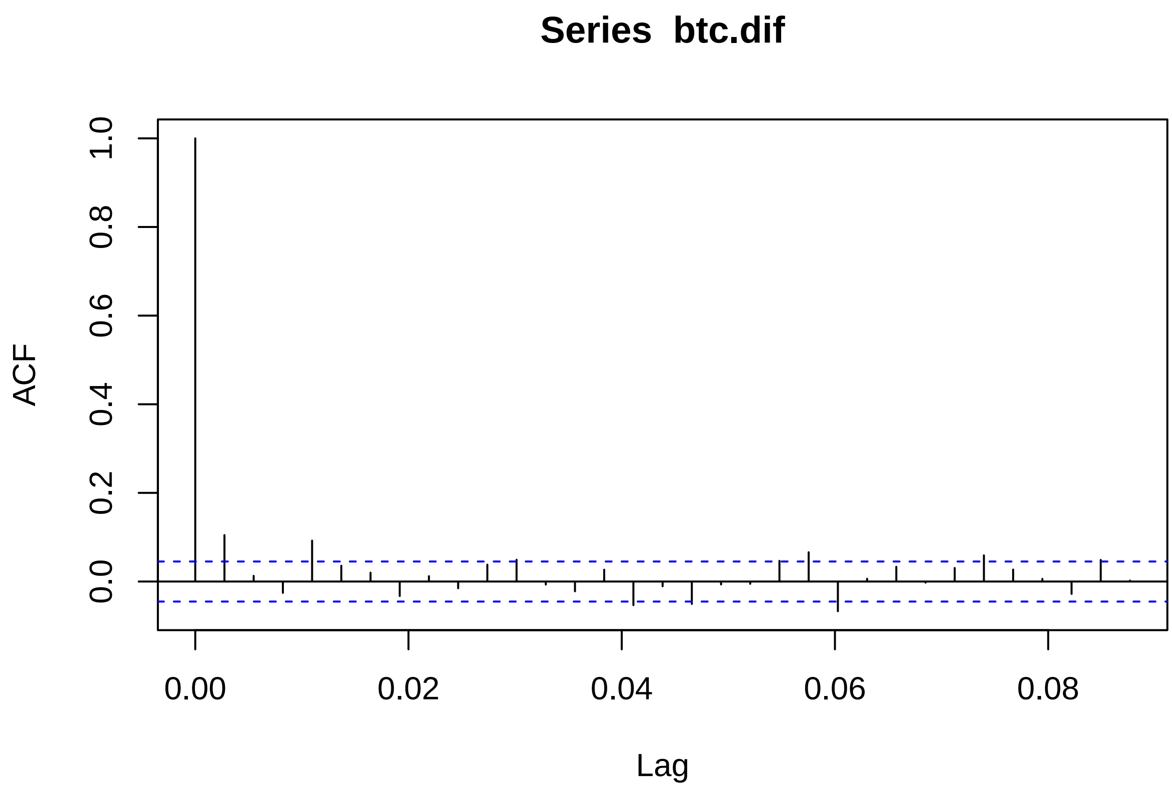
```
## [1] 20.89585
```

Ahora tenemos que nuestra serie es un proceso estacionario en cuanto a la media.

5.2 Función de autocorrelación y de autocorrelación parcial

Vamos a observar como se comporta nuestra serie tras aplicar la función de autocorrelación parcial.

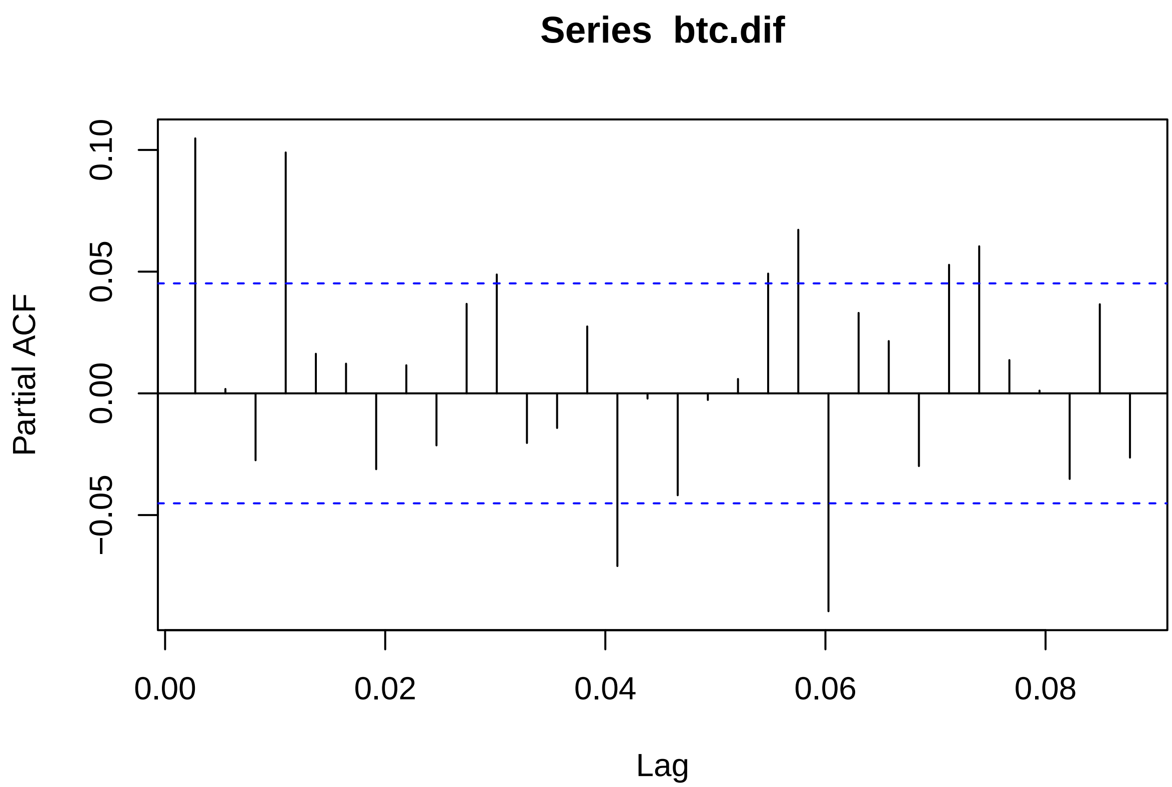
```
acf(btc.dif)
```



Vemos que tenemos dos (2) coeficientes no nulos, el 1 y el 4, podemos probar con procesos de media movil de los dos coeficientes MA(1) y MA(4).

Ahora aplicamos con la función de autocorrelación parcial.

```
pacf(btc.dif)
```



No observamos tendencia. Podemos ver dos (2) coeficientes no nulos, el 1 y el 4. Podemos probar con AR(1) y AR(4).

5.3 Encontrar el modelo ARIMA(p,d,q)

Tenemos que a nuestros datos les podemos aplicar varios modelos ARIMA.

- ARIMA(1,1,1)
- ARIMA(4,1,1)
- ARIMA(4,1,4)
- ARIMA(1,1,4)

ARIMA(1,1,1)

```
fit.1 <- arima(btc.dif, order=c(1,0,1))
fit.1
```

```
##
## Call:
## arima(x = btc.dif, order = c(1, 0, 1))
##
## Coefficients:
##          ar1     ma1  intercept
##       0.0993  0.0053    20.9790
## s.e.  0.1767  0.1766    20.9666
##
## sigma^2 estimated as 664235:  log likelihood = -15285.86,  aic = 30579.72
```

ARIMA(4,1,1)

```
fit.2 <- arima(btc.dif, order=c(4,0,1))
fit.2
```

```
##
## Call:
## arima(x = btc.dif, order = c(4, 0, 1))
##
## Coefficients:
##          ar1      ar2      ar3     ar4      ma1  intercept
##       0.2754  -0.0134  -0.0386  0.1035  -0.1698    21.5198
## s.e.  0.2116   0.0324   0.0238  0.0230   0.2127    23.0416
##
## sigma^2 estimated as 657025:  log likelihood = -15275.61,  aic = 30565.23
```

ARIMA(4,1,4)

```
fit.3 <- arima(btc.dif, order=c(4,0,4))
fit.3
```

```
##
## Call:
## arima(x = btc.dif, order = c(4, 0, 4))
##
## Coefficients:
##          ar1     ar2      ar3     ar4      ma1      ma2     ma3      ma4
##       0.2010  0.0659  -0.4814  0.2098  -0.0943  -0.0741  0.4435  -0.0655
## s.e.  0.1879  0.1917   0.1402  0.1944   0.1904   0.2053  0.1412   0.2057
##       intercept
##         20.3577
## s.e.    22.4646
##
## sigma^2 estimated as 655503:  log likelihood = -15273.44,  aic = 30566.88
```

ARIMA(1,1,4)

```
fit.4 <- arima(btc.dif, order=c(1,0,4))
fit.4
```

```
##
## Call:
## arima(x = btc.dif, order = c(1, 0, 4))
##
## Coefficients:
##          ar1      ma1      ma2      ma3     ma4  intercept
##       0.3397  -0.2349  -0.0195  -0.0340  0.0956    21.5003
## s.e.  0.1894   0.1885   0.0304   0.0243  0.0220    22.8433
##
## sigma^2 estimated as 657552:  log likelihood = -15276.37,  aic = 30566.73
```

A priori los modelos ARIMA(4,1,1), ARIMA(4,1,4) y ARIMA(1,1,4) logran aic's parecidos, siendo ARIMA(4,1,1) algo mejor (30565.23).

Vamos a comparar con la función auto.arima

```
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
```

```
##   method            from
##   as.zoo.data.frame zoo
```

```
fit.auto <- auto.arima(insample)
fit.auto
```

```
## Series: insample
## ARIMA(3,1,4)
##
## Coefficients:
##          ar1     ar2      ar3      ma1      ma2     ma3     ma4
##       0.4287  0.1446  -0.7957  -0.3269  -0.1676  0.7553  0.1587
## s.e.  0.0827  0.1071   0.0799   0.0838   0.1064  0.0802  0.0237
##
## sigma^2 = 653088:  log likelihood = -15266.75
## AIC=30549.51   AICc=30549.58   BIC=30593.83
```

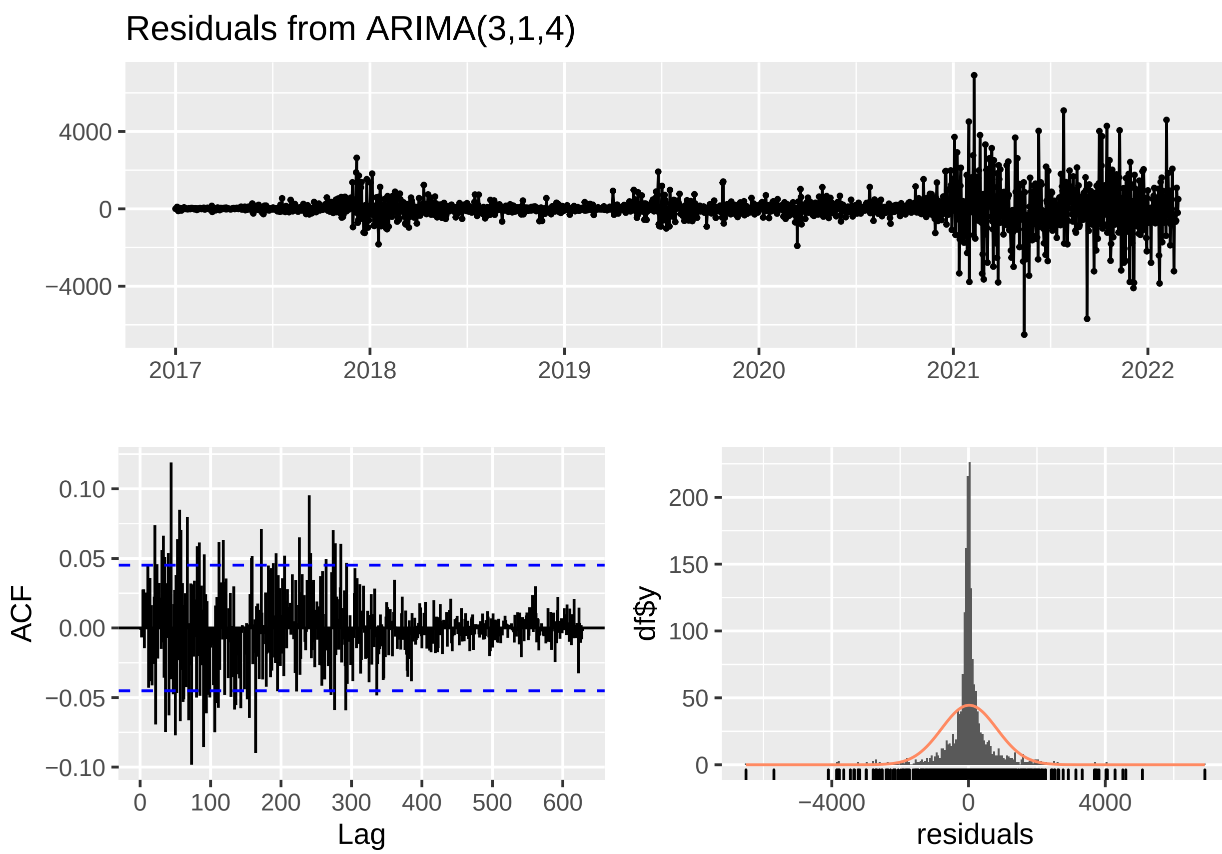
Como podemos ver logra un aic algo mejor (30549.51) con un modelo **ARIMA(3,1,4)**.

5.4 Calculo de la bondad del ajuste

```
accuracy(fit.auto)
```

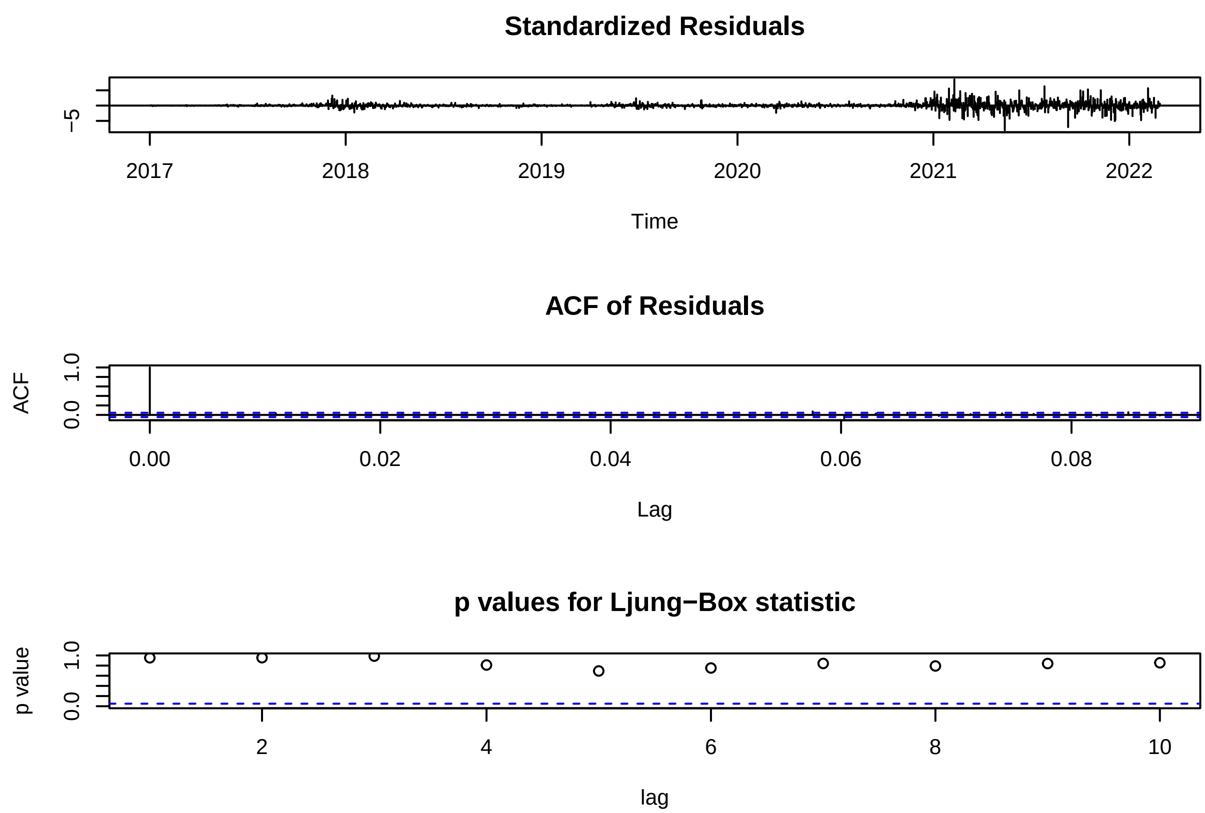
```
##                    ME     RMSE      MAE       MPE     MAPE       MASE
## Training set 18.06035 806.4204 412.9755 0.1136865 2.485472 0.03246267
##                      ACF1
## Training set -0.001343427
```

```
checkresiduals(fit.auto,plot=TRUE)
```



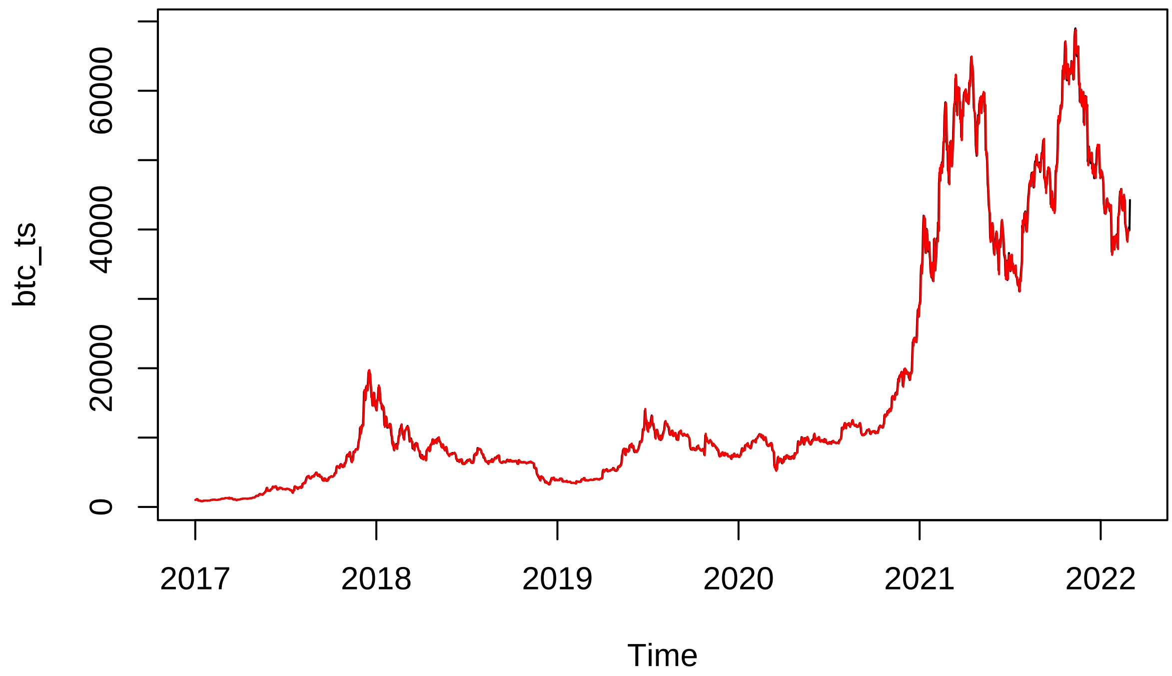

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,1,4)
## Q* = 836.07, df = 370, p-value < 2.2e-16
##
## Model df: 7.   Total lags used: 377
```

```
tsdiag(fit.auto)
```



5.5 Representación de la serie real vs la ajustada

```
fitval <- fit.auto$fitted
plot(btc_ts,col="black")
lines(fitval,col="red")
```



5.6 Predicción

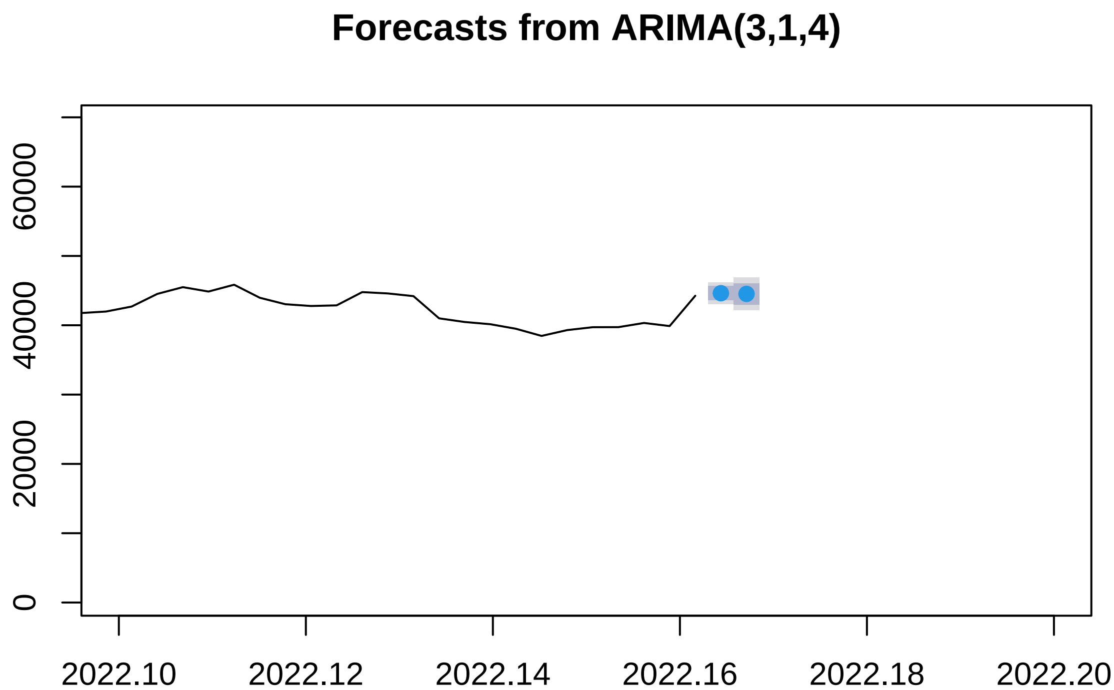
```
forecast(fit.auto, h=2)$mean;outsample
```

```
## Time Series:
## Start = c(2022, 59)
## End = c(2022, 60)
## Frequency = 365
## [1] 40667.89 40733.83
```

```
## Time Series:
## Start = c(2022, 59)
## End = c(2022, 60)
## Frequency = 365
## [1] 39886.92 44256.08
```

Vemos que nuestra predicción nos muestra que los dos últimos datos pertenecientes al precio del BTC en los días 28/02/2022 y 01/03/2022 es de 40667.89 y 40733.83. Si lo comparamos con los datos reales tenemos que el valor en esos días fue de 39886.92 y 44256.08

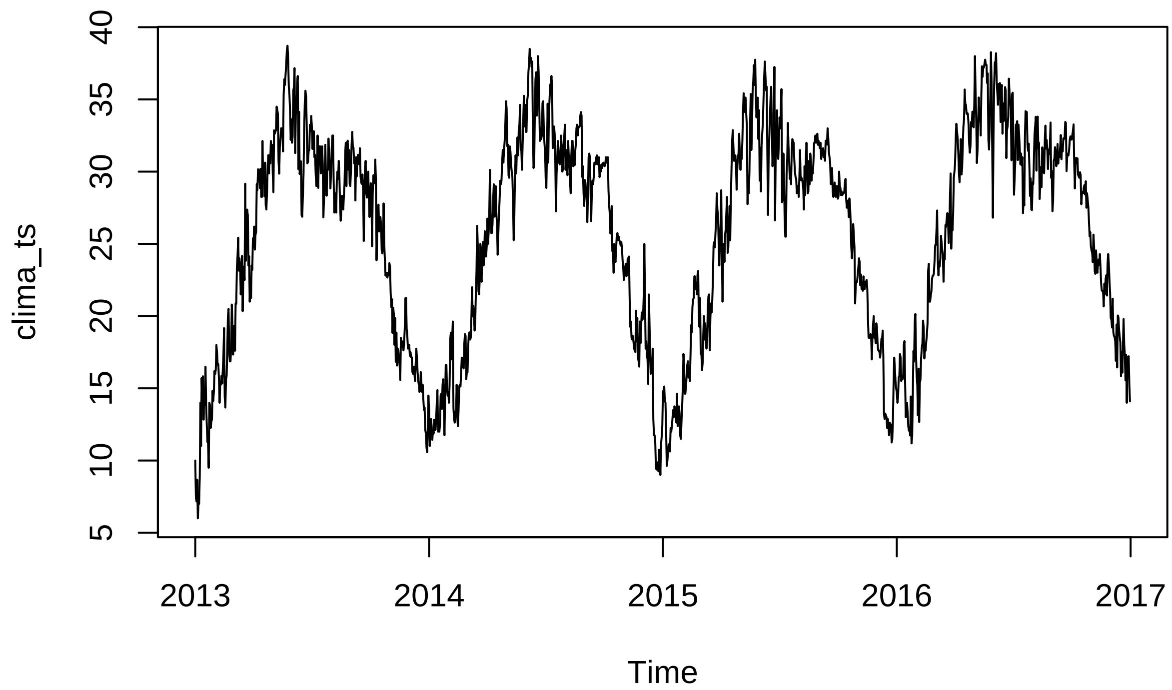
```
plot(forecast(btc_ts,h=2,model=fit.auto),xlim=c(2022.1,2022.2),type = "l")
```


Forecasts from ARIMA(3,1,4)
0
20000
40000
60000
2022.10
2022.12
2022.14
2022.16
2022.18
2022.20

6 Modelo ARIMA en serie con tendencia + estacionalidad

6.1 Transformar la serie a un modelo estacionario

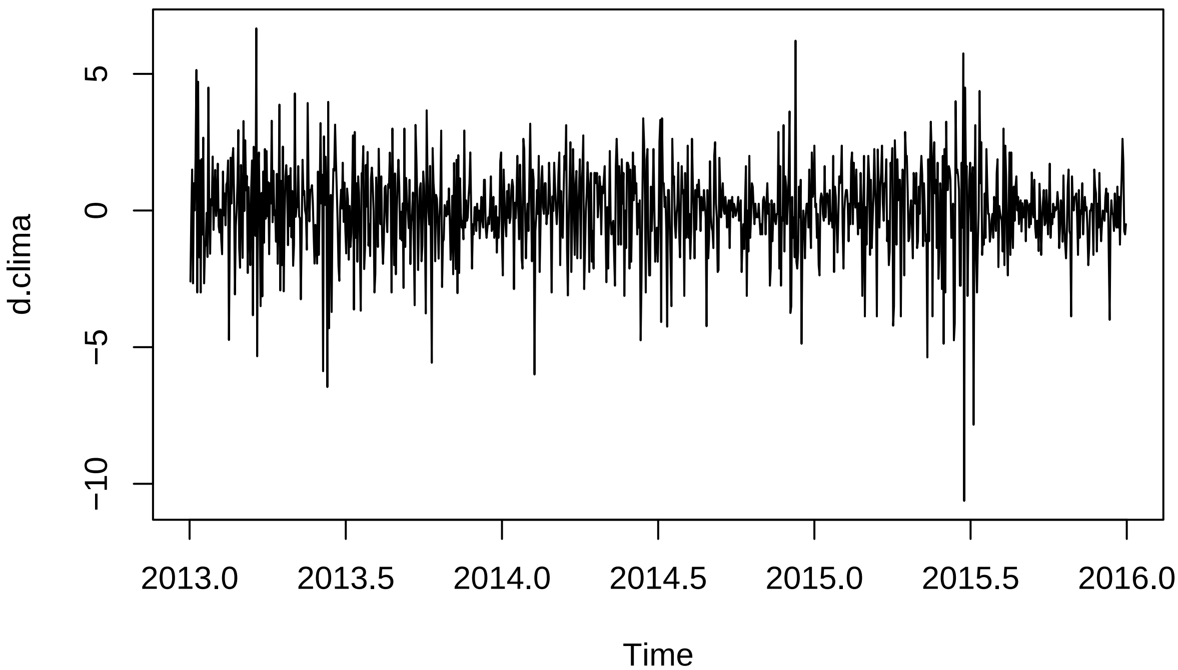
```
clima_ts <- ts(clima$meantemp,start=c(2013,1),end=c(2016,365),frequency=365)
plot(clima_ts)
```




```
insample <- window(clima_ts,start=c(2013,1),end=c(2015,365))
outsample <- window(clima_ts,start=c(2016,1),end=c(2016,365))
```

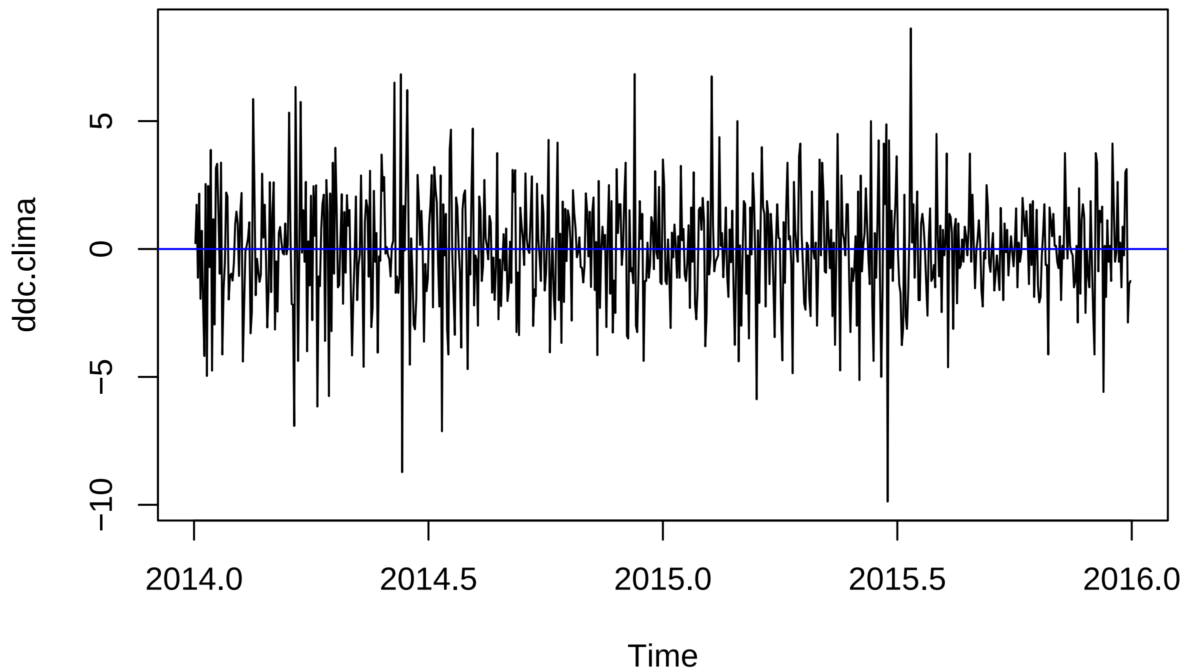
Serie diferenciada una vez

```
d.clima <- diff(insample) # Serie diferenciada una vez
plot(d.clima,type="l")
```



Hemos quitado la tendencia con la diferencia regular, pero aún queda la estacionalidad.

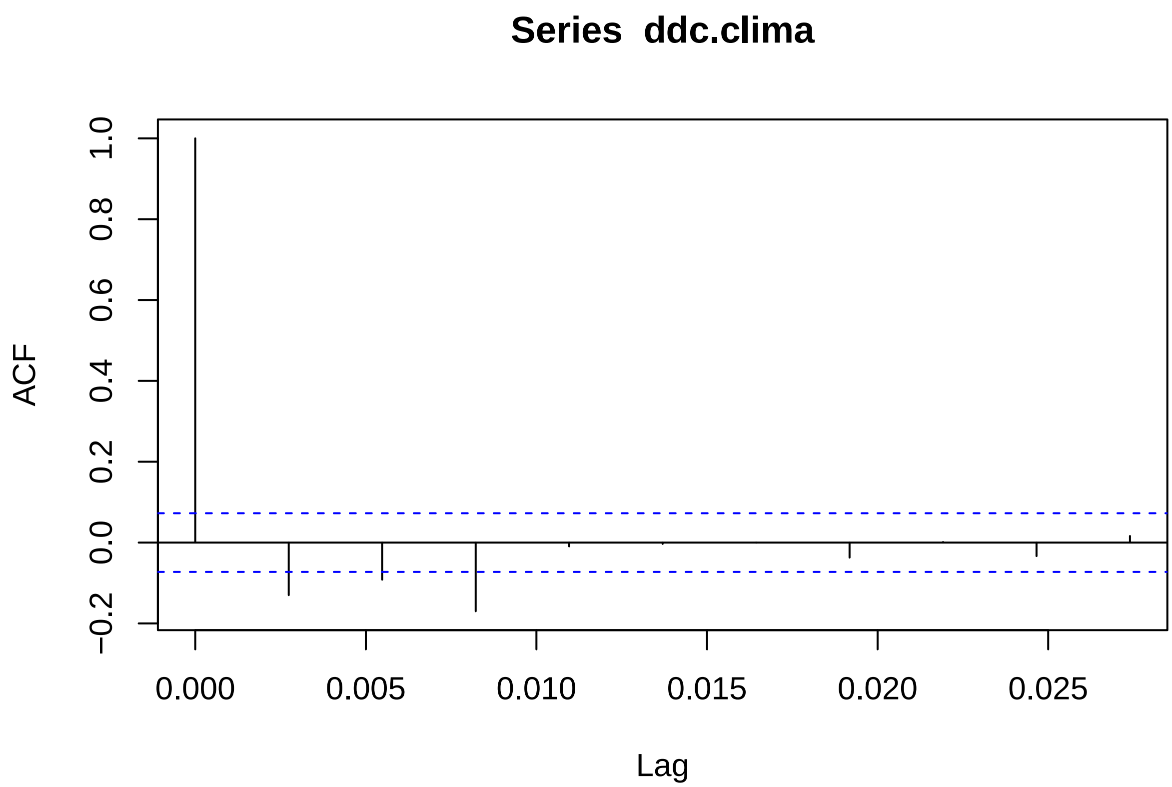
```
ddc.clima <- diff(d.clima,365) # Diferencia estacional
plot(ddc.clima,type="l")
abline(h = mean(ddc.clima), col = "blue")
```



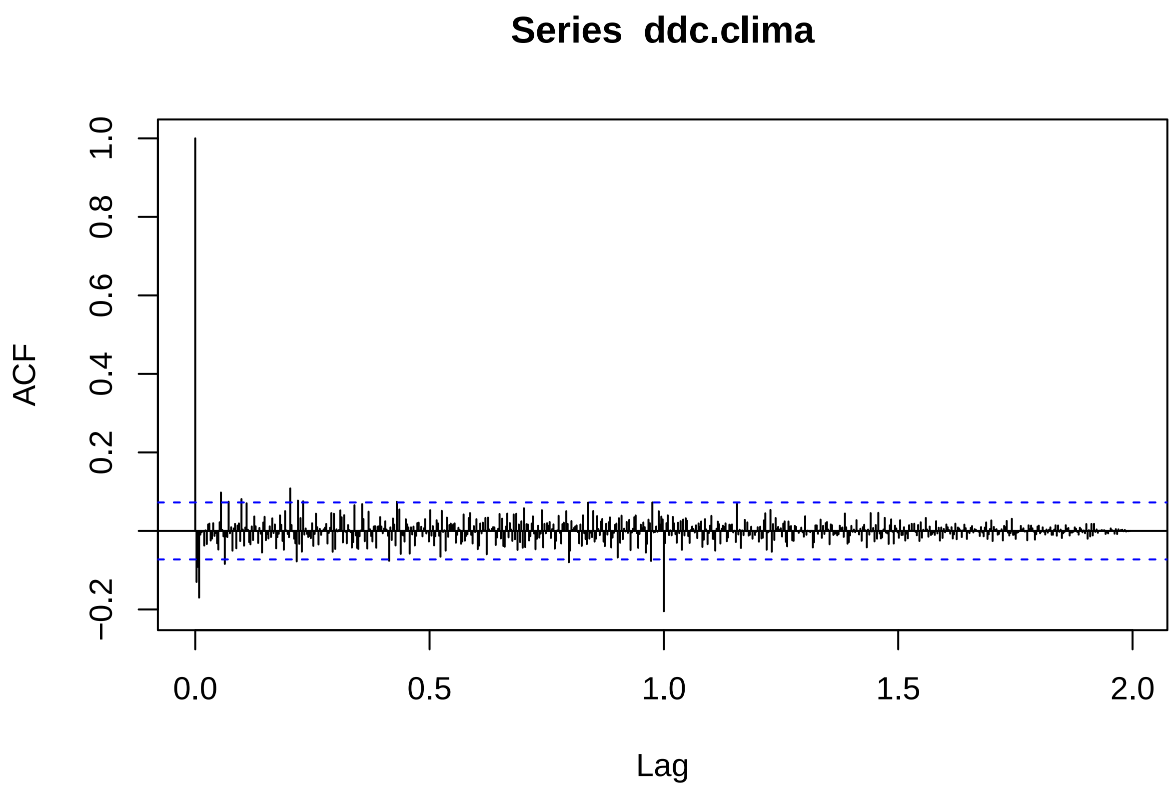
6.2 Función de autocorrelación y de autocorrelación parcial

Pasamos pues a examinar el correlograma y correlograma parcial.

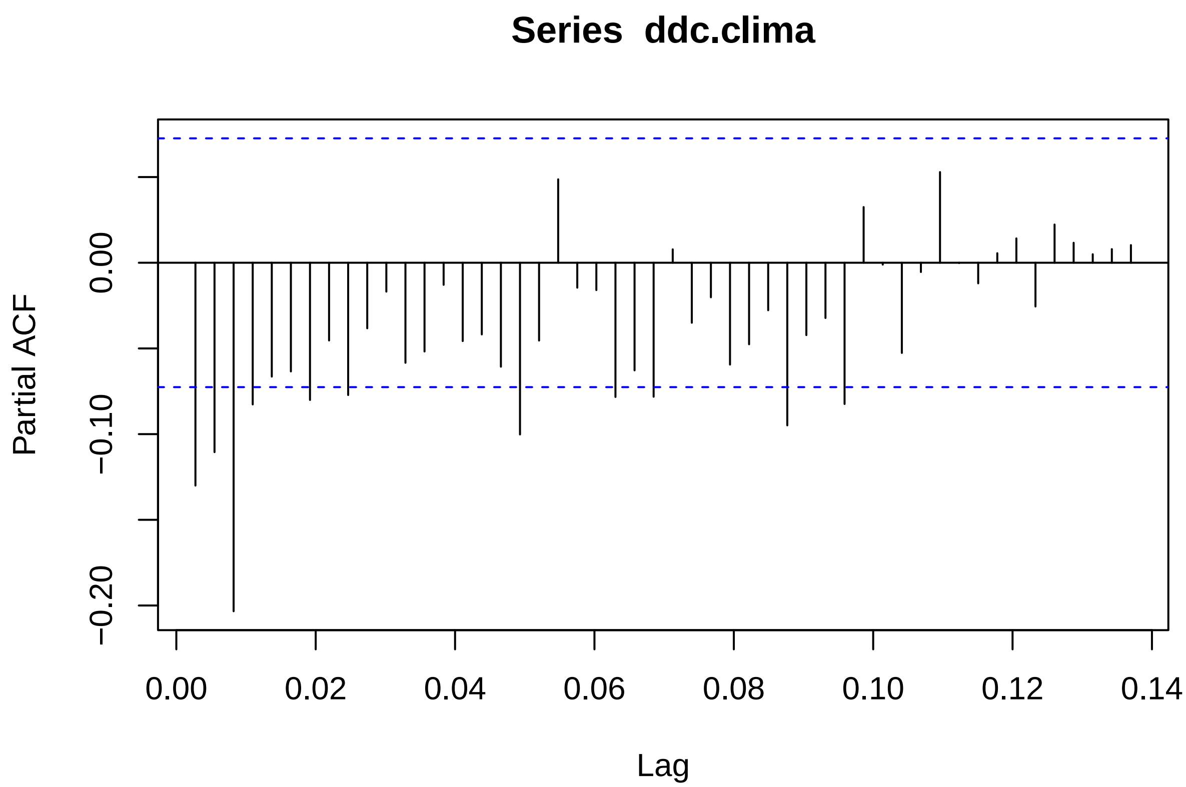
```
acf(ddc.clima,lag.max=10) # q = 3
```



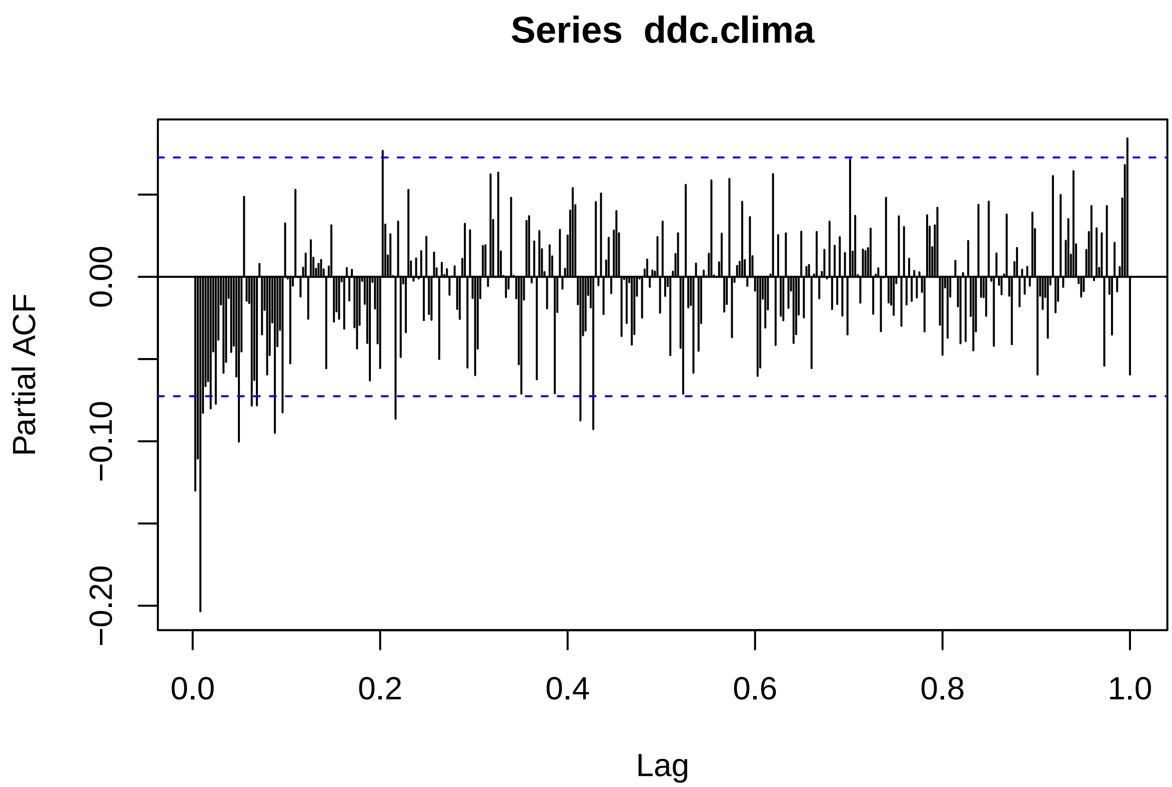

```
acf(ddc.clima,lag.max=1095) # Q = 1
```



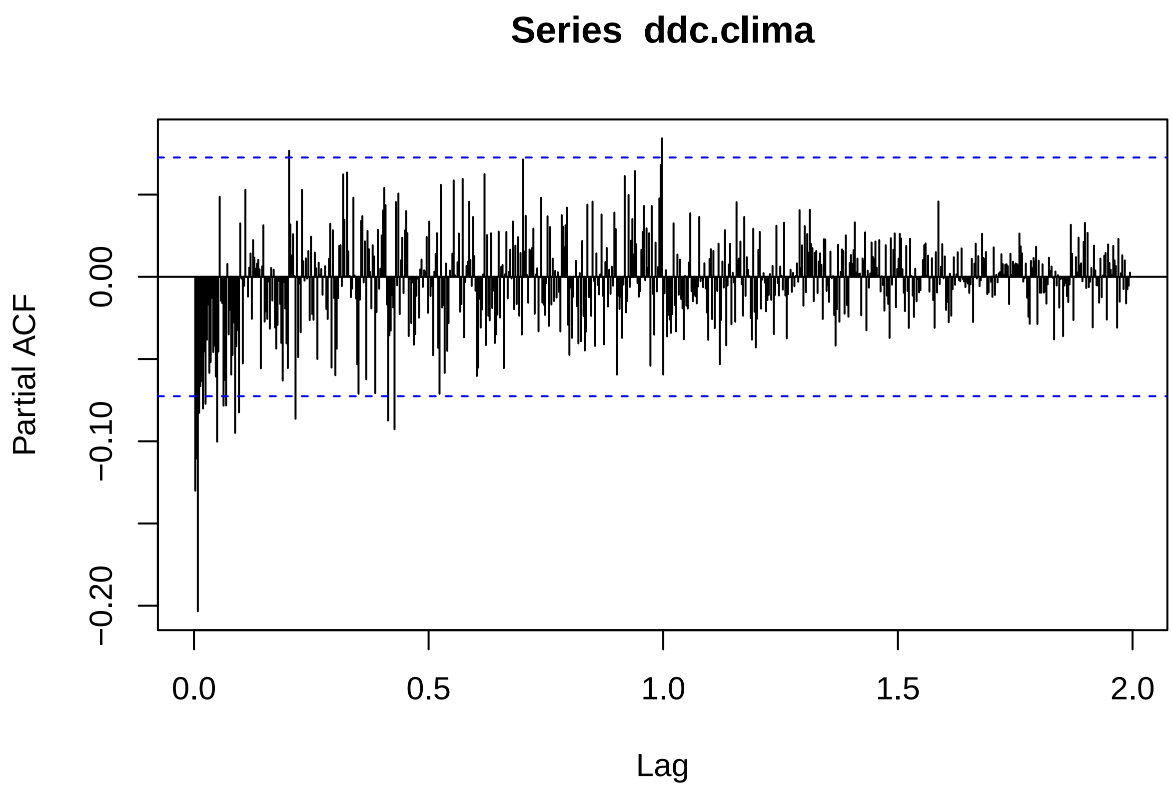

```
pacf(ddc.clima,lag.max = 50) # p = 3
```




```
pacf(ddc.clima,lag.max = 365) # P = 0
```




```
pacf(ddc.clima,lag.max = 1095)
```



6.3 Encontrar el modelo sARIMA

Probamos el modelo sARIMA que hemos intuido.

- sARIMA(3,0,3)(0,0,1)

```
# fit.1 <- arima(ddc.clima, order=c(3,0,3), seasonal=list(order=c(0,0,1), period=365))
# fit.1
```

Al probar el modelo de forma manual vemos como se genera un error:

Error in makeARIMA(trarma[[1L]], trarma[[2L]], Delta, kappa, SSinit) : maximum supported lag is 350

Esto es debido a la longitud del periodo estacional, ya que estamos trabajando con datos diarios.

Vamos a ver que resultado nos da la función auto.arima

```
library(forecast)
fit.auto <- auto.arima(insample)
fit.auto
```

```
## Series: insample
## ARIMA(1,0,0)(0,1,0)[365]
##
## Coefficients:
##          ar1
##       0.7250
## s.e.  0.0255
##
```

```
## sigma^2 = 4.319:  log likelihood = -1569.67
## AIC=3143.33   AICc=3143.35   BIC=3152.52
```

El modelo resultante es un ARIMA(1,0,0)(0,1,0)

Logra un AIC de 3143.33.

La definición de la ecuación es: $\left(1-\theta_1 B-\theta_2 B^2-\ldots-\theta_q B^q\right)\left(1-\Theta_1 B^c-\Theta_2 B^{2c}-\ldots-\Theta_Q B^{Qc}\right)\epsilon_t$

Entonces con $(p=1, d=0, q=0)(P=0, D=1, Q=0)$ nuestra ecuación sería:

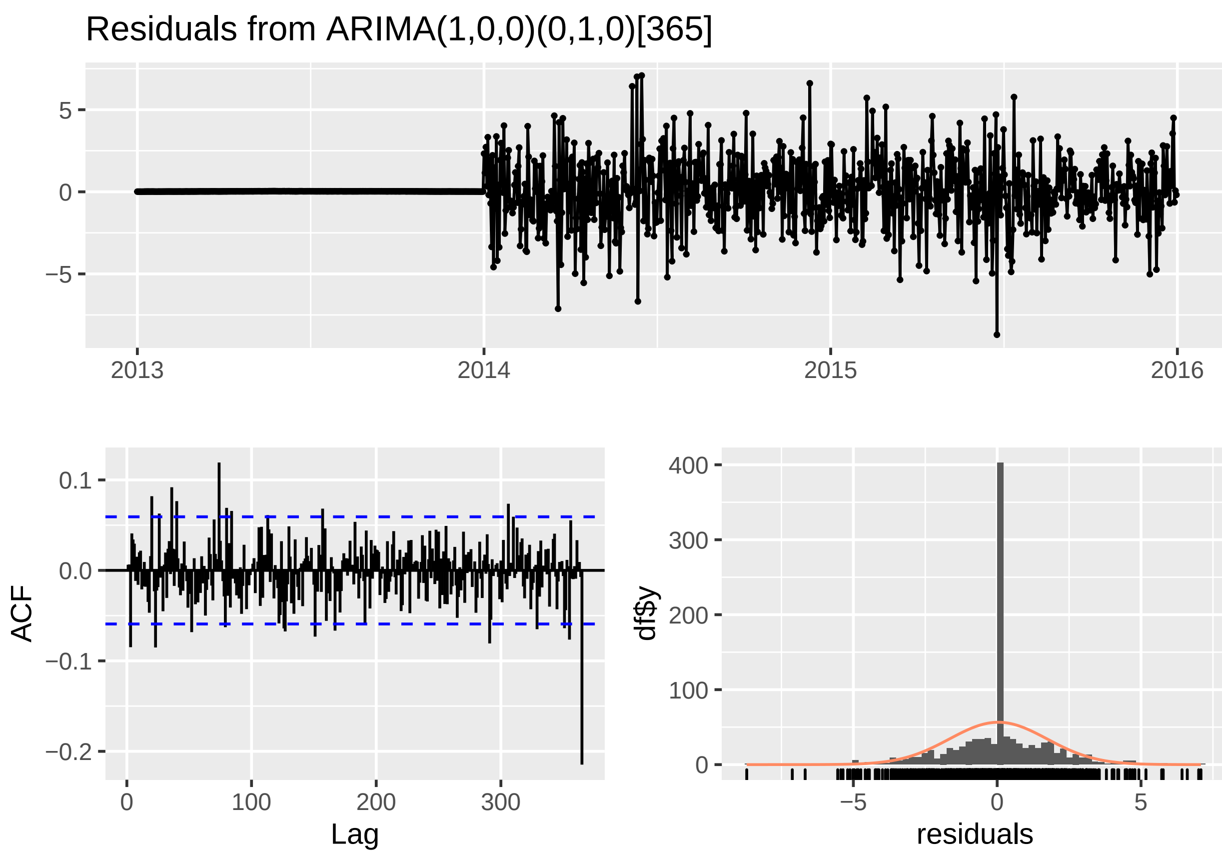
$$(1-\phi_1 B)(1-B)x_t$$
$$(1-\theta_1 B)(1-\Theta_1 B^{365})\epsilon_t$$

6.4 Calculo de la bondad del ajuste

```
accuracy(fit.auto)
```

```
##                      ME     RMSE      MAE         MPE     MAPE      MASE
## Training set 0.03865953 1.695683 1.074014 -0.07910276 4.778318 0.4441814
##                     ACF1
## Training set 0.006269335
```

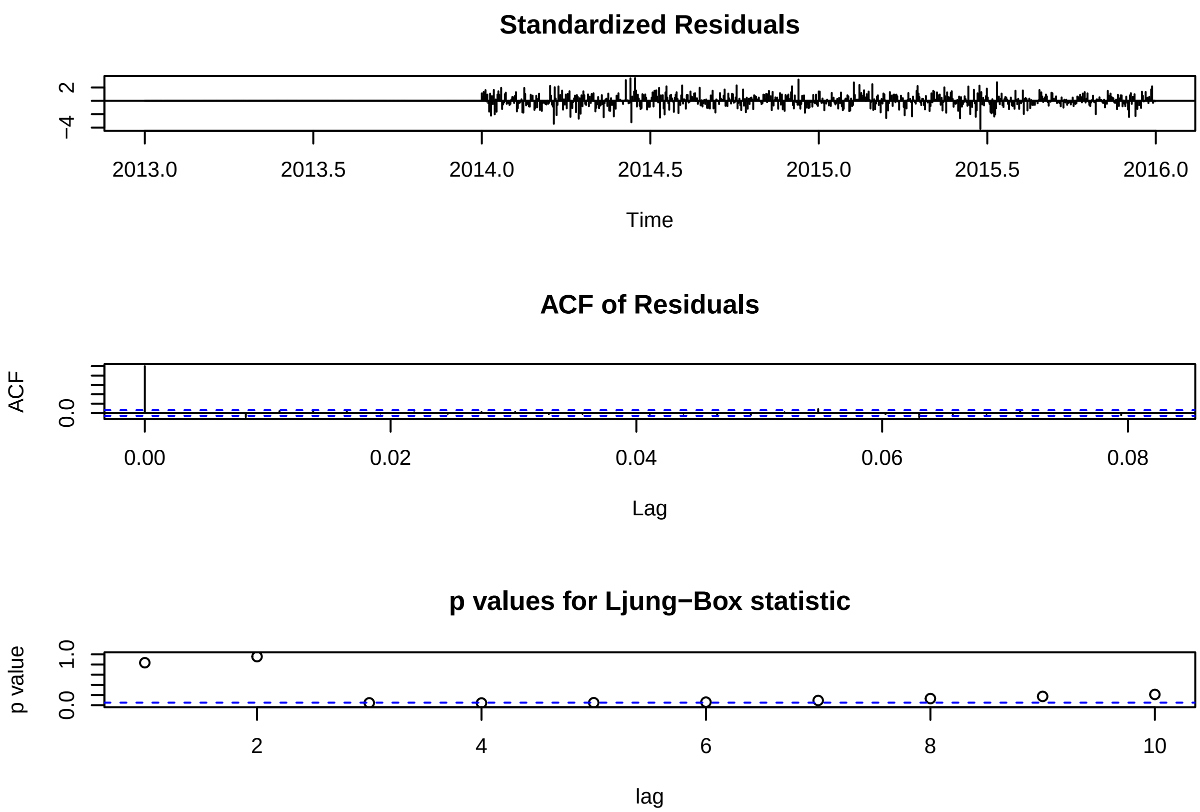
```
checkresiduals(fit.auto,plot=TRUE)
```




```
##
##  Ljung-Box test
##
```

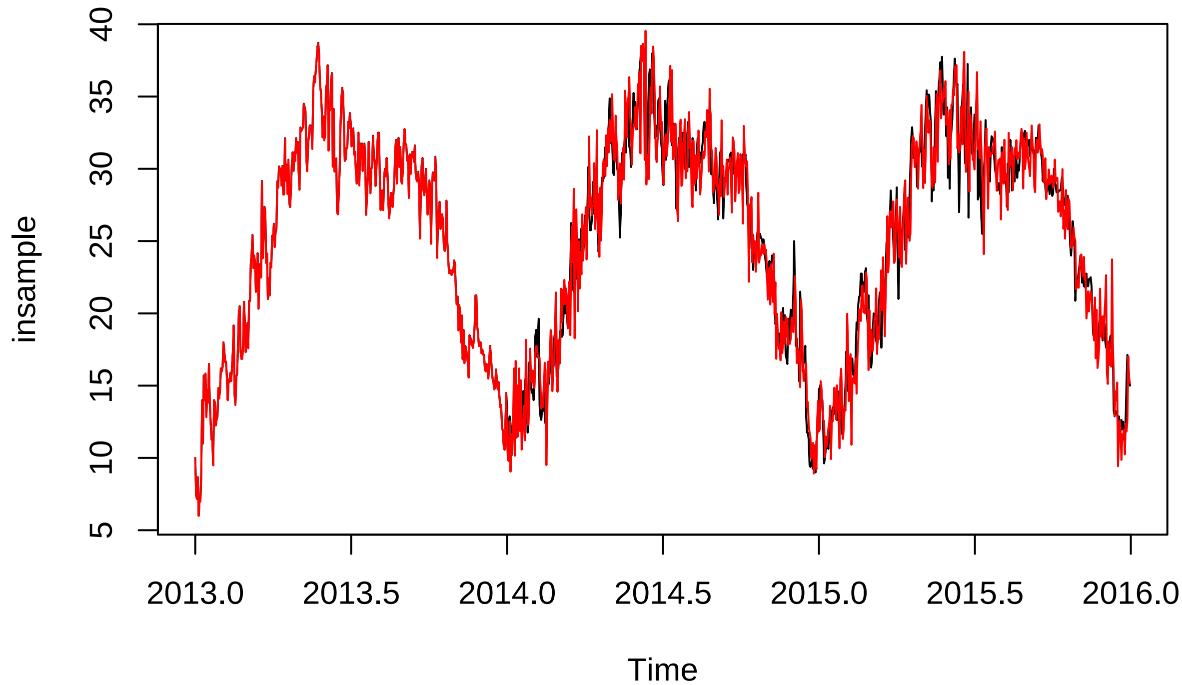
```
## data:  Residuals from ARIMA(1,0,0)(0,1,0)[365]
## Q* = 275.51, df = 218, p-value = 0.005015
##
## Model df: 1.   Total lags used: 219
```

```r
tsdiag(fit.auto)
```



6.5 Representación de la serie real vs la ajustada

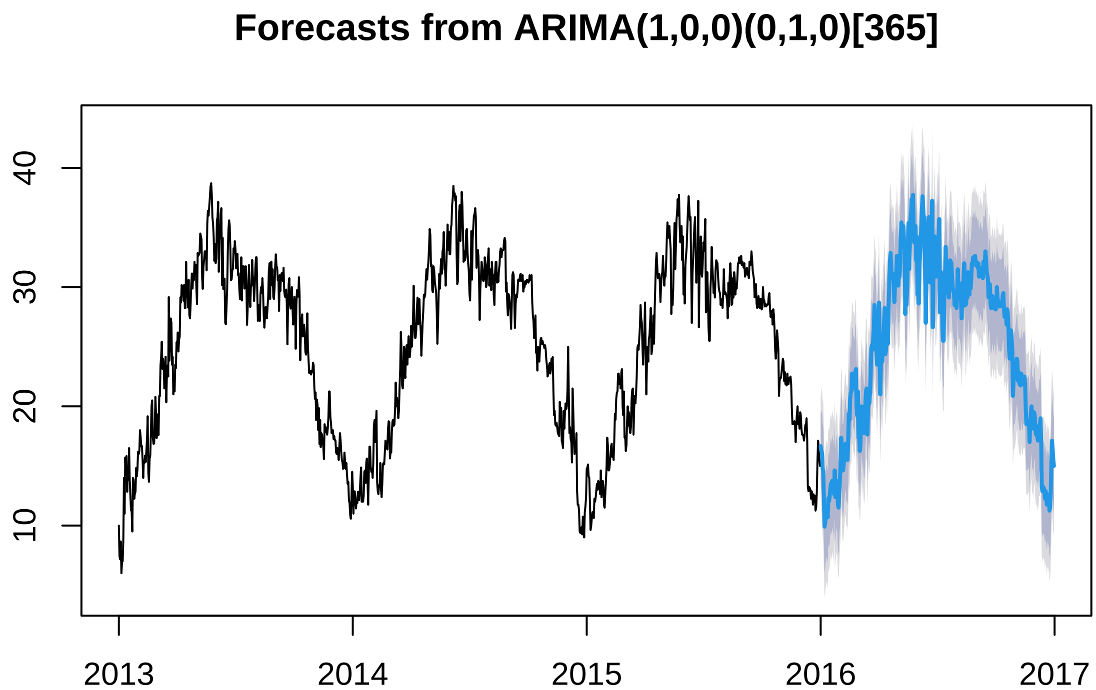
```r
fitval <- fit.auto$fitted
plot(insample,col="black")
lines(fitval,col="red")
```



6.6 Predicción

```
# forecast(fit.auto, h=365)$mean
```

```
plot(forecast(insample,h=365,model=fit.auto),type = "l")
```



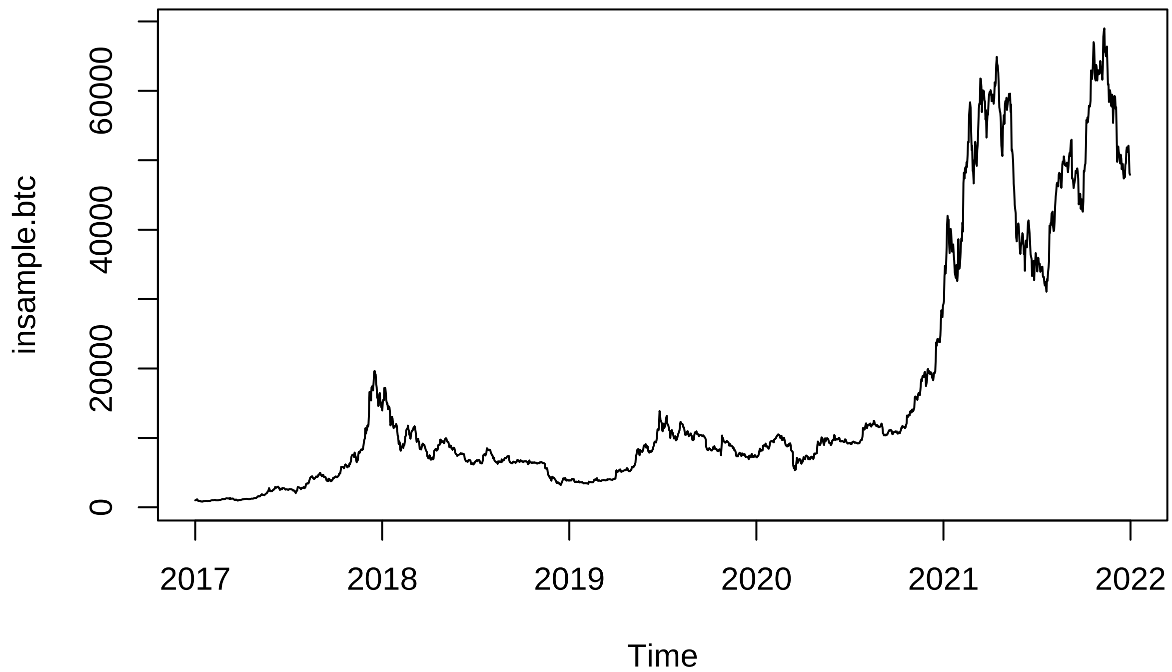
7 Modelo NAR en serie con tendencia

7.1 Encontrar el modelo NAR

```
library(forecast)
outsample.btc <- window(btc_ts,start=c(2022,1),end=c(2022,60))
insample.btc <- window(btc_ts,start=c(2017,1),end=c(2021,365))
```

- Hemos dividido la serie temporal dejando dos meses para la predicción.

```
plot(insample.btc)
```



La función 'nnetar' del paquete 'forecast' de R nos permite ajustar un modelo NNAR(p,k)$_c$ donde:

- p = número de observaciones previas consideradas
- k = número de nodos en la capa intermedia.

```
set.seed(123)
btc.fit.nar.p2.s2 <- nnetar(insample.btc) # NNAR(2,1,2)[365]
btc.fit.nar.p4.s2 <- nnetar(insample.btc, p=4, size=2) # probamos a aumentar p
btc.fit.nar.p2.s4 <- nnetar(insample.btc, p=2, size=4) # probamos a aumentar k
```

7.2 Cálculo de la bondad de ajuste

```
accuracy(btc.fit.nar.p2.s2)
```

```
##                     ME    RMSE      MAE        MPE     MAPE       MASE
## Training set 0.1368009 865.602 457.3792 -0.1645078 2.395915 0.03530978
##                      ACF1
## Training set -0.001598146
```

```
accuracy(btc.fit.nar.p4.s2)
```

```
##                      ME     RMSE      MAE        MPE     MAPE       MASE
## Training set -0.3261809 864.3386 458.1012 -0.2185455 2.411576 0.03536552
##                     ACF1
## Training set 0.003795655
```

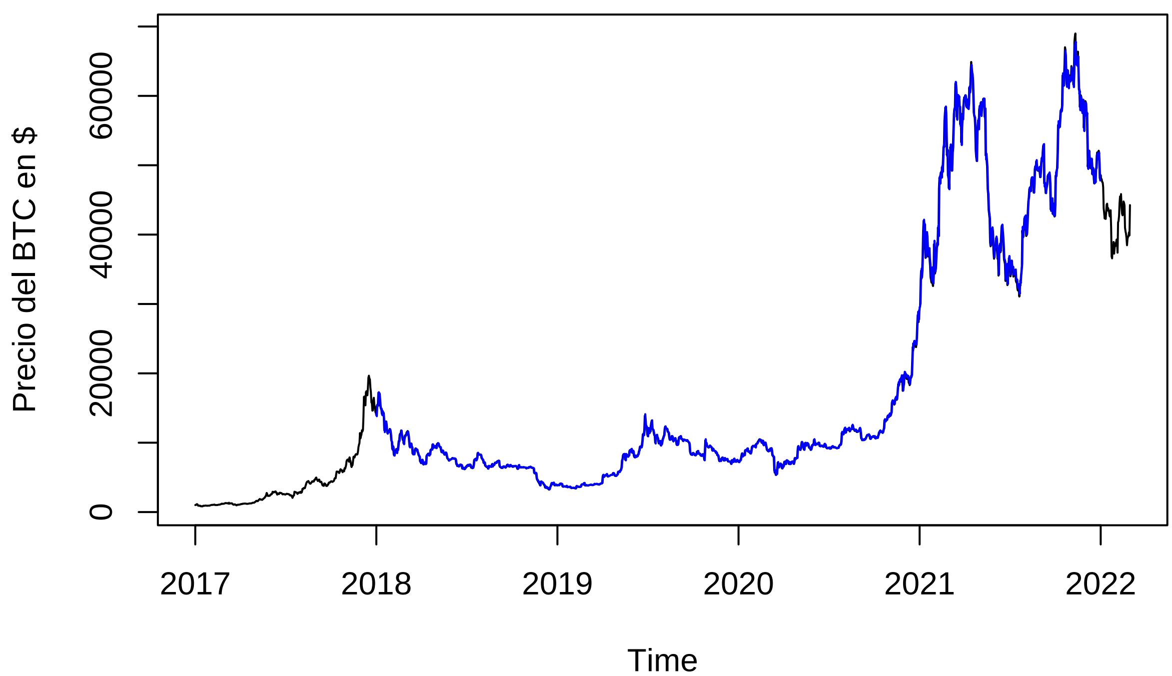
```r
accuracy(btc.fit.nar.p2.s4)
```

```
##                     ME     RMSE      MAE        MPE     MAPE       MASE
## Training set 0.5095583 860.2726 453.2957 -0.1335004 2.330718 0.03499453
##                      ACF1
## Training set -0.005219131
```

- A priori, aumentando k logramos mejores resultados.

7.3 Representación de la serie real vs. ajustada

```r
fitval <- fitted.values(btc.fit.nar.p2.s4)
plot(btc_ts,ylab="Precio del BTC en $")
lines(fitval, col="blue")
```



7.4 Calculo de la predicción para h=60 instantes temporales futuros

```r
pred.p2.s2 <- forecast(btc.fit.nar.p2.s2, h = 60)  # Predicción puntual para h = 60 (dos meses)
pred.p4.s2 <- forecast(btc.fit.nar.p4.s2, h = 60)
pred.p2.s4 <- forecast(btc.fit.nar.p2.s4, h = 60)
```

```r
rmse_pred.p2.s2 <- sqrt(mean((outsample.btc - pred.p2.s2$mean)^2));rmse_pred.p2.s2
```

```
## [1] 3361.892
```

```
rmse_pred.p4.s2 <- sqrt(mean((outsample.btc - pred.p4.s2$mean)^2));rmse_pred.p4.s2
```

```
## [1] 3891.437
```

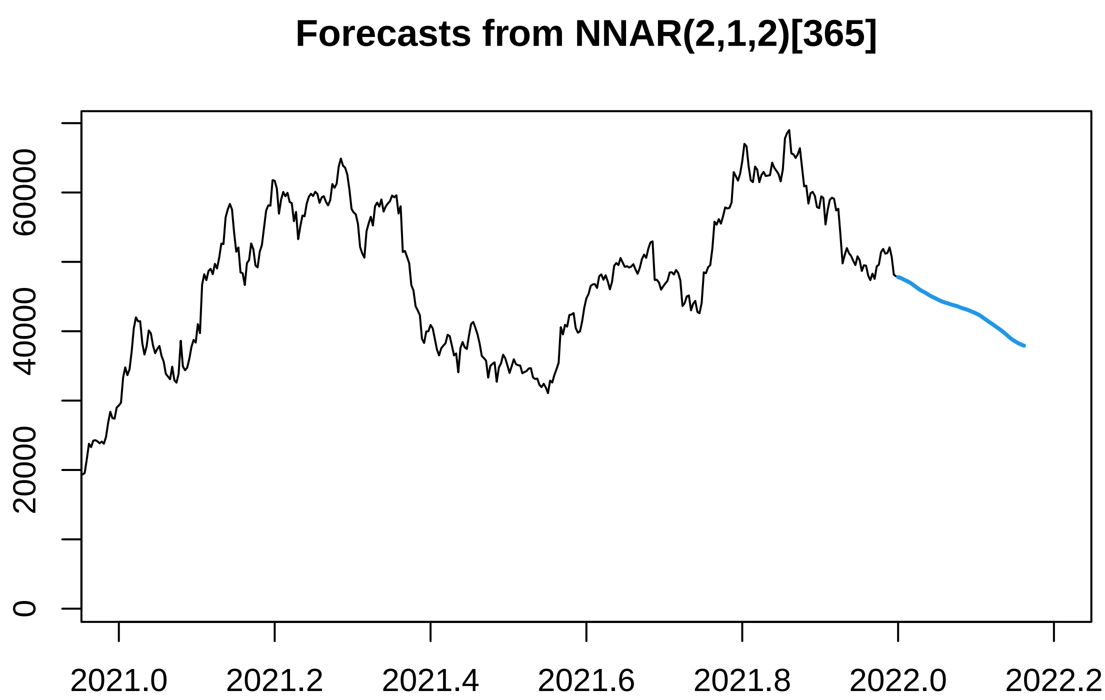
```
rmse_pred.p2.s4 <- sqrt(mean((outsample.btc - pred.p2.s4$mean)^2));rmse_pred.p2.s4
```

```
## [1] 4388.542
```

- Tras calcular la bondad de ajuste con los datos de la predicción y el test, se observa que el modelo generado automáticamente NNAR(2,1,2)[365] es el que mejor resultados proporciona. $RMSE = 3361.892$.

7.5 Representación gráfica de la serie junto a la predicción obtenida

```
plot(pred.p2.s2,xlim=c(2021,2022.2))
```

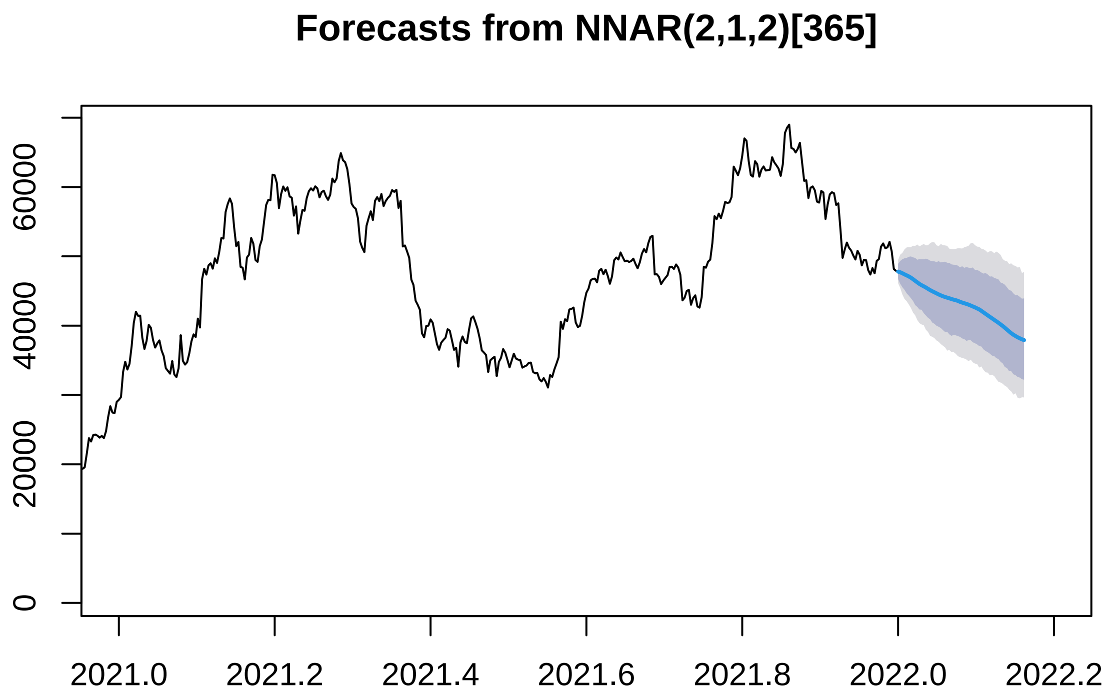



```
df.comp <- data.frame(outsample.btc,pred.p2.s2$mean)
df.comp
```

```
##   outsample.btc pred.p2.s2.mean
## 1      48589.47        47803.61
## 2      47960.98        47701.44
## 3      47989.00        47561.25
## 4      47586.58        47402.47
## 5      47526.00        47256.16
## 6      46855.06        47103.83
## 7      43782.86        46924.78
```

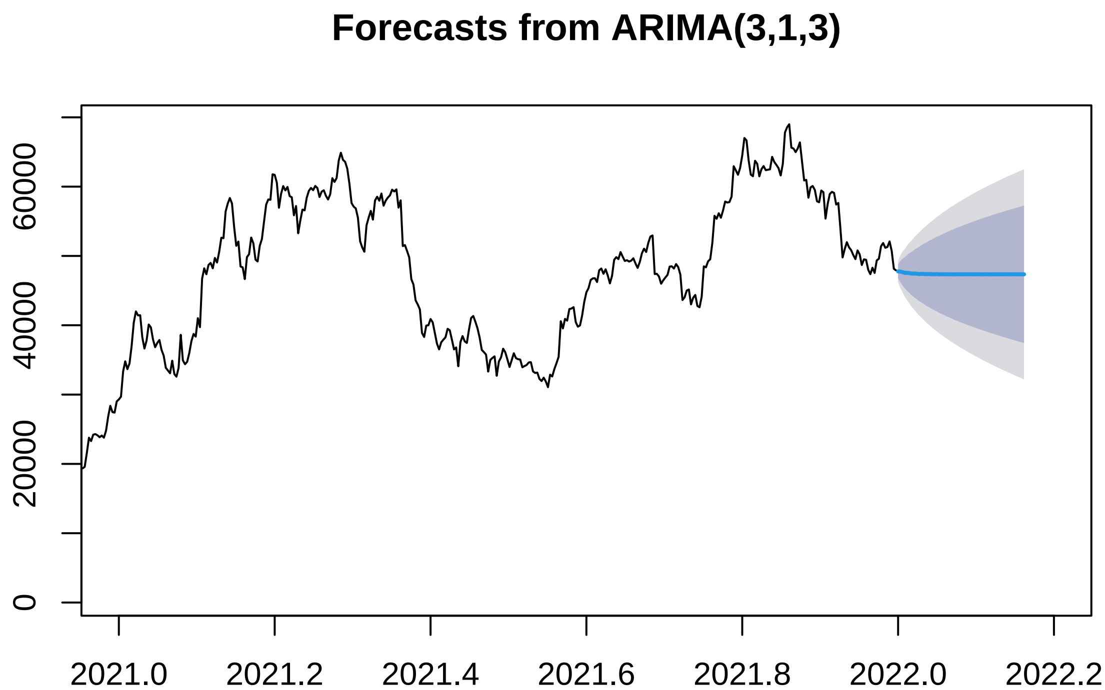
```
## 8        43135.35         46708.63
## 9        42315.31         46474.17
## 10       42796.49         46247.39
## 11       42256.55         46024.36
## 12       43144.74         45839.11
## 13       44337.26         45677.66
## 14       44456.34         45508.54
## 15       43468.95         45316.44
## 16       43826.80         45129.11
## 17       43495.59         44963.32
## 18       43209.47         44813.66
## 19       42685.25         44661.38
## 20       42589.90         44506.56
## 21       43518.69         44368.50
## 22       41115.58         44242.55
## 23       36825.98         44137.61
## 24       36574.47         44040.33
## 25       38050.00         43949.15
## 26       37552.30         43841.34
## 27       38946.00         43752.96
## 28       37251.00         43671.56
## 29       38022.11         43578.90
## 30       38741.67         43438.39
## 31       38378.88         43332.15
## 32       38776.33         43236.48
## 33       39285.00         43139.69
## 34       38883.96         43031.45
## 35       37391.74         42906.44
## 36       41760.39         42771.53
## 37       41983.12         42641.53
## 38       42701.86         42487.78
## 39       44524.18         42346.84
## 40       45501.00         42141.21
## 41       44865.72         41918.07
## 42       45850.00         41705.53
## 43       43969.72         41485.23
## 44       43034.00         41265.44
## 45       42779.60         41057.33
## 46       42871.68         40840.32
## 47       44785.66         40632.93
## 48       44590.75         40416.14
## 49       44204.78         40183.96
## 50       40996.31         39955.77
## 51       40471.27         39697.87
## 52       40151.62         39432.31
## 53       39494.11         39164.87
## 54       38463.88         38910.44
## 55       39303.24         38692.81
## 56       39720.00         38507.56
## 57       39727.97         38324.07
## 58       40330.99         38175.83
## 59       39886.92         38035.05
## 60       44256.08         37912.48
```

```
pred.p2.s2.IC <- forecast(btc.fit.nar.p2.s2, PI = TRUE, h=60)  # Predicción puntual e intervalos de pre
plot(pred.p2.s2.IC,xlim=c(2021,2022.2))
```



7.6 Comparación con el mejor modelo ARIMA

```
fit.auto <- auto.arima(insample.btc)
plot(forecast(insample.btc,h=60,model=fit.auto),xlim=c(2021,2022.2),type = "l")
```




```
rmse_pred.arima <- sqrt(mean((outsample.btc - forecast(fit.auto, h=60)$mean)^2));rmse_pred.arima
```

```
## [1] 6069.468
```

- Por el RMSE y como observamos el comportamiento de la predicción de la serie en el gráfico, llegamos a la conclusión de que el modelo NAR es mejor que el mejor modelo ARIMA. Aún así, podríamos alcanzar mejores bondades de ajuste de forma iterativa.

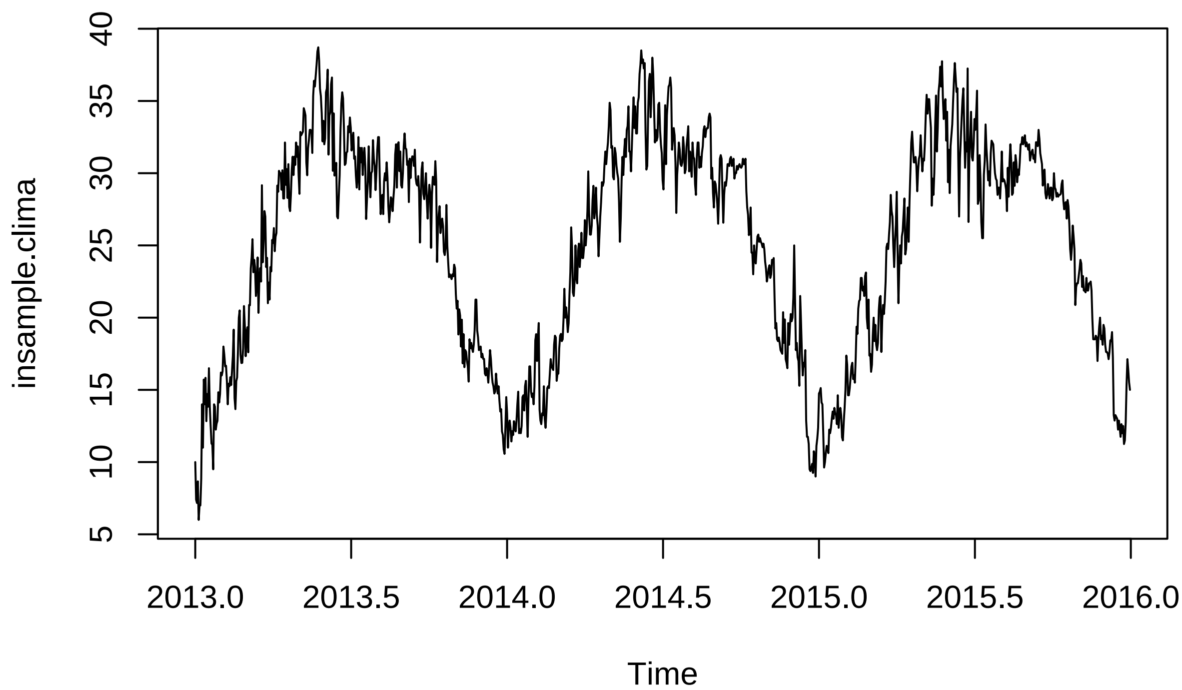
8 Modelo NAR en serie con tendencia + estacionalidad

8.1 Encontrar el modelo NAR

```
library(forecast)
insample.clima <- window(clima_ts,start=c(2013,1),end=c(2015,365))
outsample.clima <- window(clima_ts,start=c(2016,1),end=c(2016,365))
```

- Hemos dividido la serie temporal dejando un año para la predicción.

```
plot(insample.clima)
```



La función ‘nnetar’ del paquete ‘forecast’ de R nos permite ajustar un modelo NNAR(p,P,k)$_c$ donde:

- p = número de observaciones previas consideradas.
- P = número de observaciones del mismo periodo en ciclos anteriores consideradas.
- k = número de nodos en la capa intermedia.

```
set.seed(123)
fit.nar.1 <- nnetar(insample.clima); # NNAR(4,1,3)
fit.nar.2 <- nnetar(insample.clima, p=6, P=1 ,size=3) # aumentamos p
fit.nar.3 <- nnetar(insample.clima, p=4, P=1 ,size=5) # aumentamos k
```

8.2 Cálculo de la bondad de ajuste

```
accuracy(fit.nar.1)
```

```
##                        ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.001140047 1.491301 1.125415 -0.456522 4.896773 0.4654396
##                      ACF1
## Training set -0.003713001
```

```
accuracy(fit.nar.2)
```

```
##                        ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.001848899 1.480736 1.116928 -0.4507163 4.853291 0.4619296
##                      ACF1
## Training set -0.002002985
```

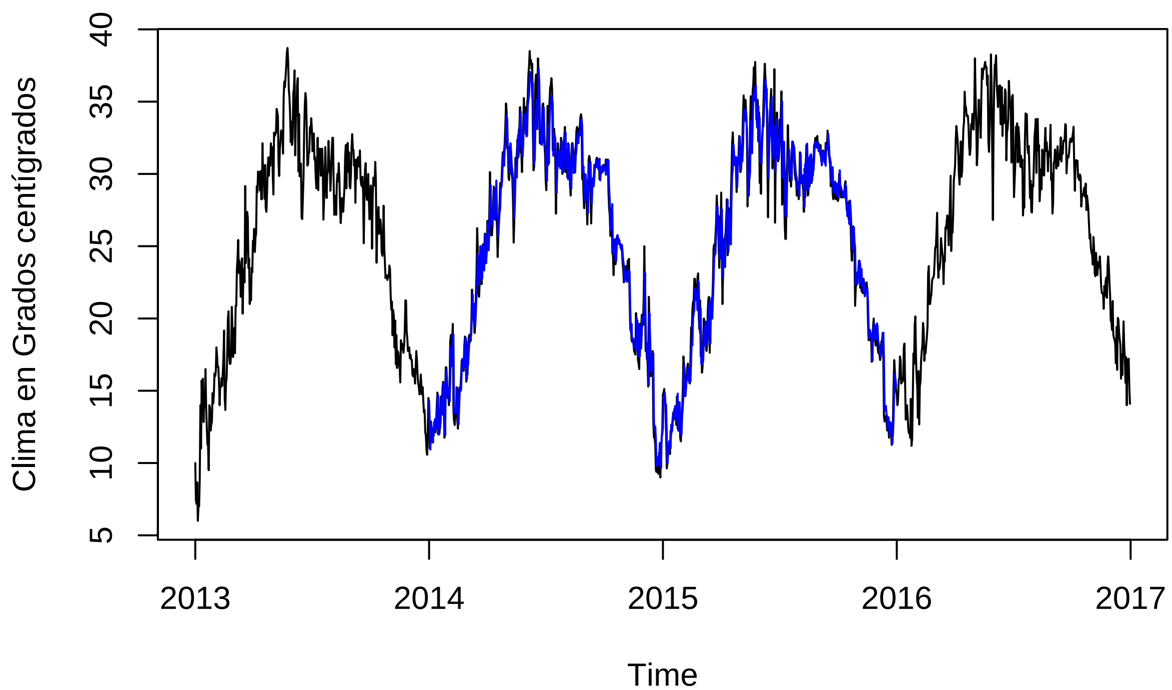
```r
accuracy(fit.nar.3)
```

```
##                       ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.00219381 1.456654 1.099385 -0.4385307 4.793073 0.4546741
##                     ACF1
## Training set -0.01063233
```

- A priori, aumentando k logramos mejores resultados.

8.3 Representación de la serie real vs. ajustada

```r
fitval <- fitted.values(fit.nar.3)
plot(clima_ts,ylab="Clima en Grados centígrados")
lines(fitval, col="blue")
```



8.4 Calculo de la predicción para h=c instantes temporales futuros

```r
pred.1 <- forecast(fit.nar.1, h = 365)  # Predicción puntual para h = 365
pred.2 <- forecast(fit.nar.2, h = 365)
pred.3 <- forecast(fit.nar.3, h = 365)
```

```r
rmse_pred.1 <- sqrt(mean((outsample.clima - pred.1$mean)^2));rmse_pred.1
```

```
## [1] 3.195646
```

```r
rmse_pred.2 <- sqrt(mean((outsample.clima - pred.2$mean)^2));rmse_pred.2
```

```
## [1] 3.235604
```

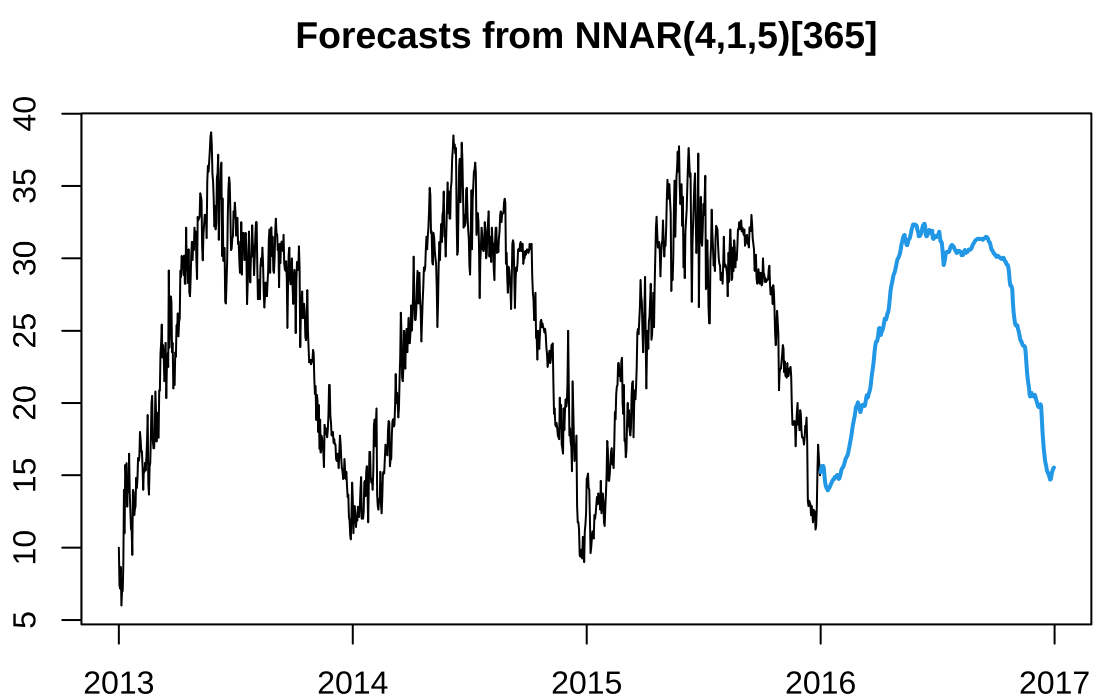
```r
rmse_pred.3 <- sqrt(mean((outsample.clima - pred.3$mean)^2));rmse_pred.3
```

```
## [1] 3.114776
```

- Podemos ver como efectivamente, aumentando el número de nodos en la capa intermedia logramos un mejor modelo. Nuestro mejor modelo sería: NNAR(4,1,5)[365].

8.5 Representación gráfica de la serie junto a la predicción obtenida

```r
plot(pred.3)
```



- La predicción con intervalos de confianza es muy costosa computacionalmente y por esta razón tenemos que obviarla.

8.6 Comparación con el mejor modelo ARIMA

```r
fit.auto <- auto.arima(insample.clima)
#plot(forecast(insample.clima,h=365,model=fit.auto),type = "l")
rmse_pred.arima <- sqrt(mean((outsample.clima - forecast(fit.auto, h=365)$mean)^2));rmse_pred.arima
```

```
## [1] 3.622999
```

- Por el RMSE y como observamos el comportamiento de la predicción de la serie en el gráfico, llegamos a la conclusión otra vez de que el modelo NAR es mejor que el mejor modelo ARIMA. Aún así, podríamos alcanzar mejores bondades de ajuste de forma iterativa.