Serie_temporal_Pablo

Pablo Hernandez Camara

2020-10-20

Contents

Librerias

En primer lugar, cargamos las librerias necesarias para analizar la serie temporal:

```
library(readr)
library(forecast)
```

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

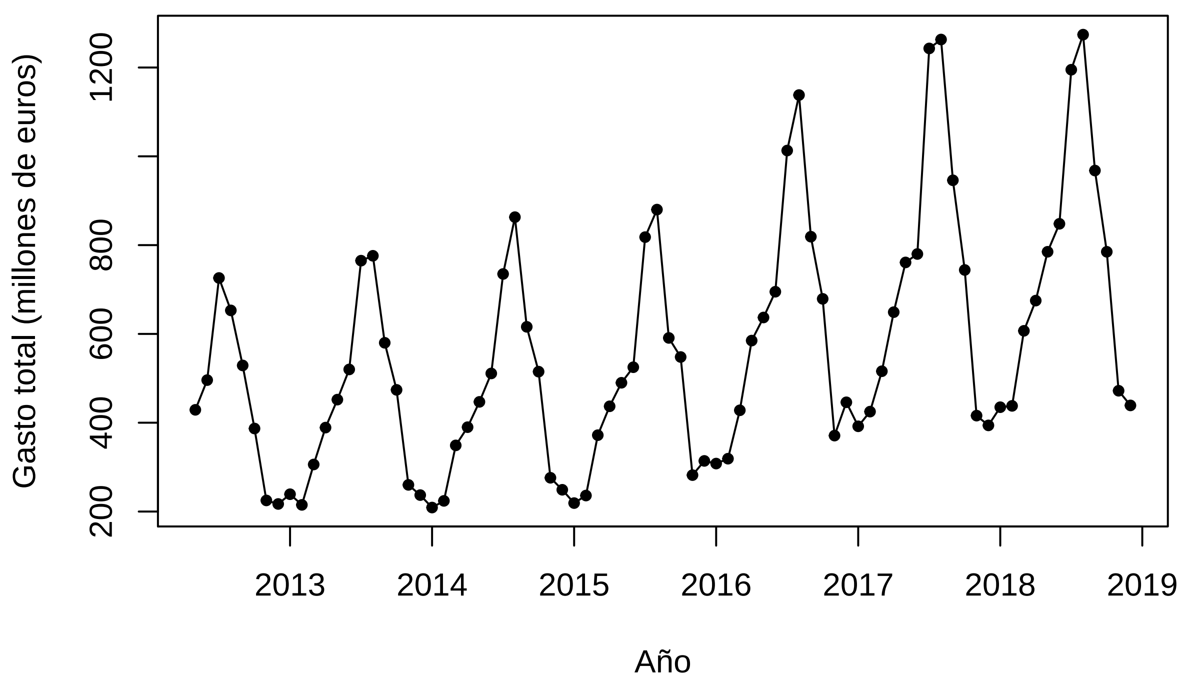
Carga y visualizacion de los datos

Leemos el archivo csv con el gasto total de los turistas extrangeros en la comunidad valenciana descargado de https://www.epdata.es/datos/turistas-internacionales-comunidad-autonoma/68/comunidad-valenciana/299:

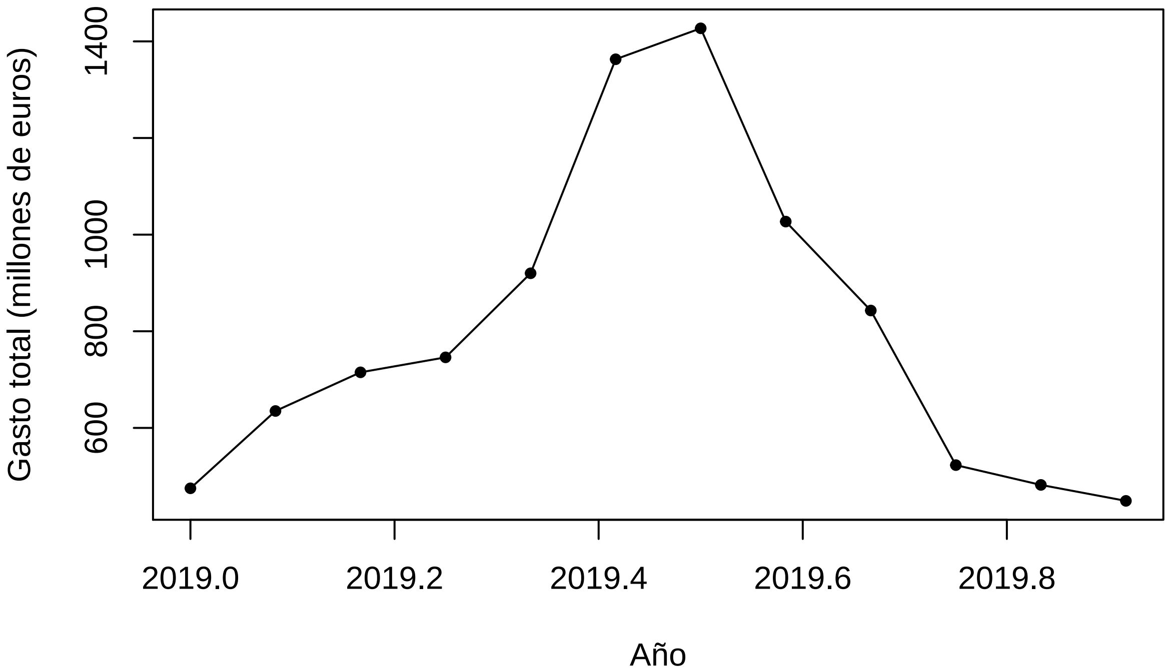
```
Data <- read.csv(file="./Datos/gasto_total_de_los_turist.csv",header=TRUE,sep=";")
```

Realizaremos el ajuste utilizando los datos de entre Mayo del 2012 (cuando empieza la serie) y 2018, dejando los datos de 2019 y 2020 para calcular unas predicciones y poder comparar con ellos:

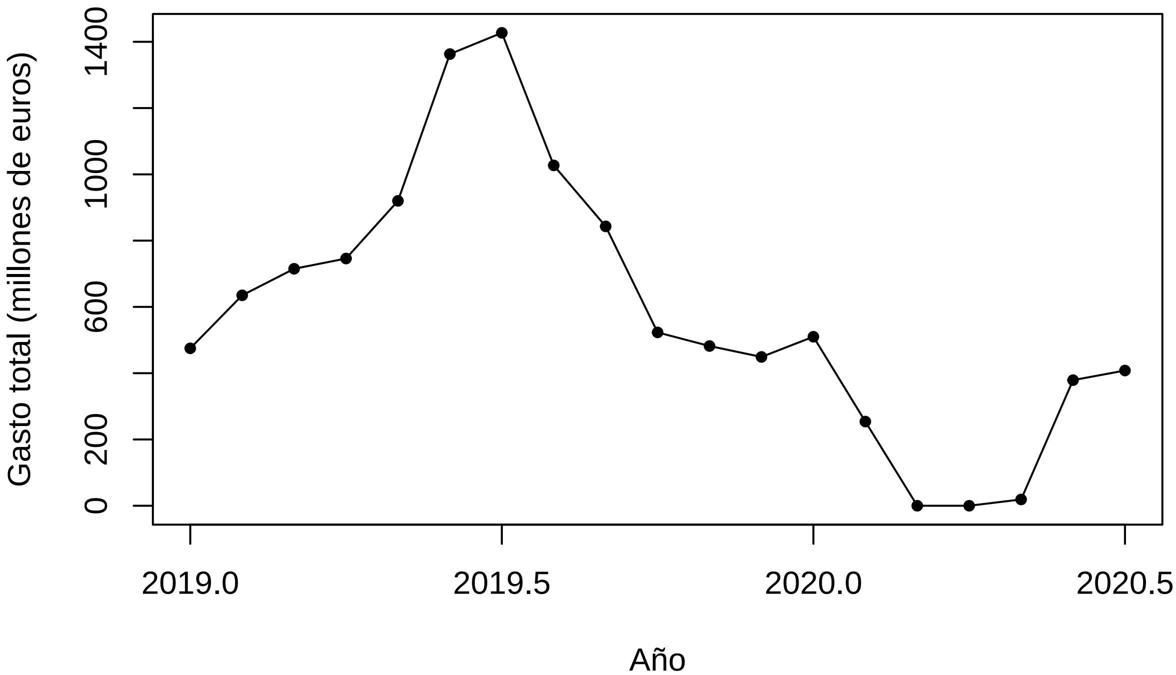
```
# Datos mensuales del gasto total de turistas extrangeros entre Mayo de 2012 y Diciembre de 2018:
insample <- ts(Data[1:80,3],start=c(2012,5),frequency=12)

# Datos mensuales del gasto total de turistas extrangeros en 2019 y en 2019-2020:
outsample_2019 <- ts(Data[82:93,3],start=c(2019,1),frequency=12)
outsample_20192020 <- ts(Data[82:100, 3],start=c(2019,1),frequency=12)

# Grafico de la serie temporal que usaremos para realizar el ajuste
plot(insample, xlab = 'Año', ylab = 'Gasto total (millones de euros)', type = 'o', pch = 20)
```




```
#Grafico de la serie temporal  de 2019 que intentaremos predecir mediante nuestro modelo
plot(outsample_2019, xlab = 'Año', ylab = 'Gasto total (millones de euros)', type = 'o', pch = 20)
```

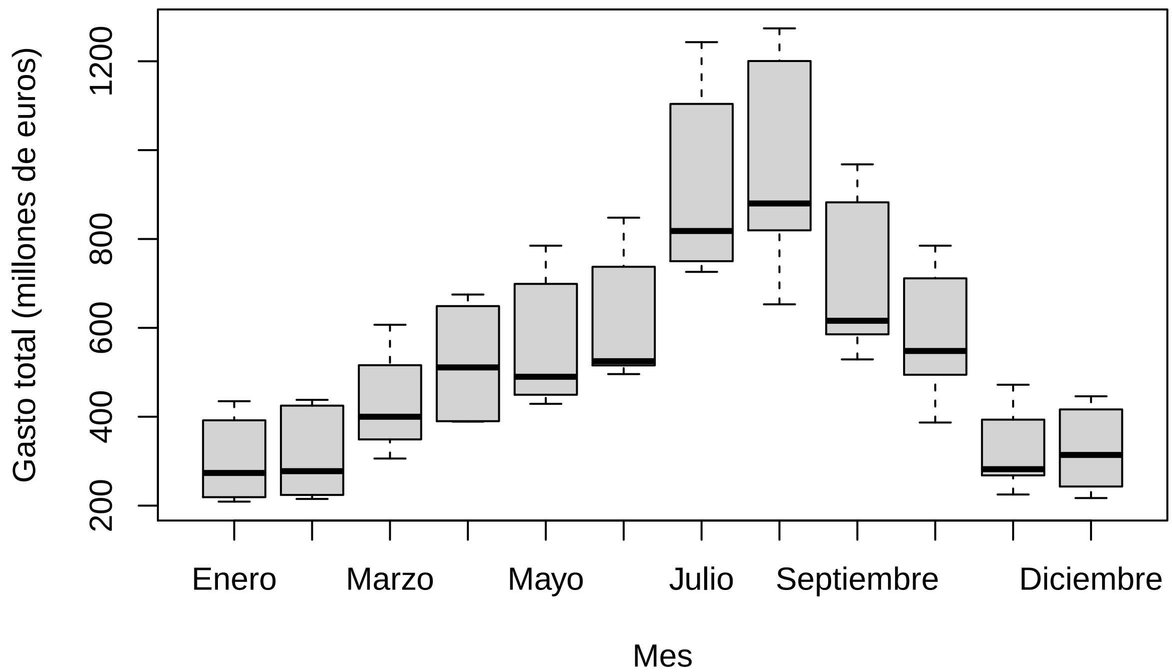



```r
#Grafico de la serie temporal  de 2020 que intentaremos predecir mediante nuestro modelo
plot(outsample_20192020, xlab = 'Año', ylab = 'Gasto total (millones de euros)', type = 'o', pch = 20)
```



A primera vista en el grafico temporal observamos que existe una clara tendencia ascendente, es decir la media de los datos va aumentando con el paso de los años. En cuanto a la varianza, se observa como va aumentando, es decir, a priori podemos suponer que se trata de un esquema multiplicativo. Por otro lado, se observa una clara estacionalidad (comportamiento ciclico repetido cada año). Realizamos un grafico de cajas para observar mejor la estacionalidad:

```
# Creamos un factor con los meses del año
mesord <- factor(Data[1:80,2], levels = c('Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio', 'Julio
boxplot(Data[1:80,3] ~ mesord, xlab = 'Mes', ylab = 'Gasto total (millones de euros)')
```

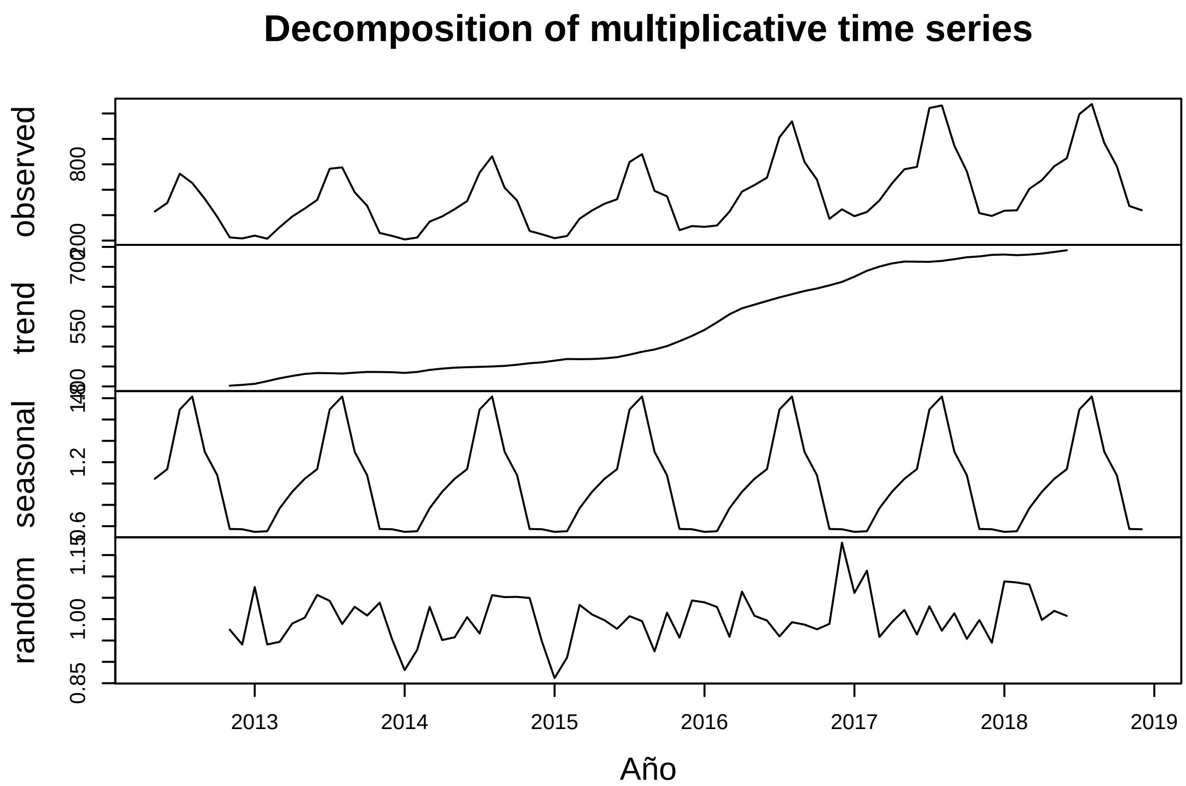


Se observa claramente que los meses de verano hay mucha mas variabilidad que el resto del año (el rango intercuartilico es mayor) y la estacionalidad, que alcanza el maximo sobre los meses de Junio y Julio.

Descomposicion de la serie temporal

Por todo lo visto anteriormente, que la serie tiene tendencia, estacionalidad y heterocedasticidad, hemos deducido que probablemente se trate de una serie multiplicativa y a continuacion la descompondremos en sus componentes: tendencia, estacionalidad y ruido:

```
comp <- decompose(insample, type="multiplicative")
plot(comp, xlab = 'Año')
```



La descomposicion de la serie nos permite ver lo que habiamos deducido del grafico temporal: la serie tiene tendencia y estacionalidad.

Coeficientes de variacion

Para analizar de forma exacta si se trata de una serie con esquema aditivo o multiplicativo calculamos los coeficientes de variacion. En primer lugar hacemos las diferencias absolutas entre los datos:

```
diferencia_absoluta <- diff(insample)
diferencia_absoluta
```

```
##       Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
## 2012                           67  230  -73 -124 -142 -162   -8
## 2013   22  -24   91   83   63   68  245   11 -196 -106 -214  -23
## 2014  -28   15  125   41   57   64  224  128 -247 -101 -239  -27
## 2015  -30   17  136   65   53   35  293   62 -289  -43 -266   32
## 2016   -6   11  109  157   52   58  318  125 -319 -140 -308   75
## 2017  -54   33   91  133  112   19  463   20 -317 -202 -328  -22
## 2018   41    3  169   68  110   63  347   79 -306 -183 -313  -33
```

A continuacion calculamos las diferencias relativas:

```
diferencia_relativa <- diferencia_absoluta
for (i in 2:length(diferencia_relativa)){
```

```r
  diferencia_relativa[i-1] <- insample[i]/insample[i-1]}
diferencia_relativa
```

```
##             Jan         Feb         Mar         Apr         May         Jun
## 2012                                                               1.1561772
## 2013   1.1013825   0.8995816   1.4232558   1.2712418   1.1619537   1.1504425
## 2014   0.8818565   1.0717703   1.5580357   1.1174785   1.1461538   1.1431767
## 2015   0.8795181   1.0776256   1.5762712   1.1747312   1.1212815   1.0714286
## 2016   0.9808917   1.0357143   1.3416928   1.3668224   1.0888889   1.0910518
## 2017   0.8789238   1.0841837   1.2141176   1.2577519   1.1725732   1.0249671
## 2018   1.1040609   1.0068966   1.3858447   1.1120264   1.1629630   1.0802548
##             Jul         Aug         Sep         Oct         Nov         Dec
## 2012   1.4637097   0.8994490   0.8101072   0.7315690   0.5813953   0.9644444
## 2013   1.4711538   1.0143791   0.7474227   0.8172414   0.5485232   0.9115385
## 2014   1.4383562   1.1741497   0.7137891   0.8360390   0.5359223   0.9021739
## 2015   1.5580952   1.0757946   0.6715909   0.9272420   0.5145985   1.1134752
## 2016   1.4575540   1.1233959   0.7196837   0.8290598   0.5463918   1.2021563
## 2017   1.5935897   1.0160901   0.7490103   0.7864693   0.5591398   0.9471154
## 2018   1.4091981   1.0661088   0.7598116   0.8109504   0.6012739 -33.0000000
```

Y calculamos los coeficientes de variacion para las diferencias absolutas y relativas:

```r
cv_diferencia_absoluta <- sd(diferencia_absoluta)/mean(diferencia_absoluta)
cv_diferencia_relativa <- sd(diferencia_relativa)/mean(diferencia_relativa)
cv_diferencia_relativa < cv_diferencia_absoluta # gana el multiplicativo
```

```
## [1] TRUE
```

Como en este caso, el coeficiente de variacion de las diferencias absolutas es mayor que el coeficiente de variacion de las diferencias relativas concluimos que tal y como habiamos supuesto la serie sigue un esquema multiplicativo.

Analisis de la serie mediante suavizado exponencial

Por lo visto anteriormente, presenta tendencia y estacionalidad, el metodo adecuado para su analisis es el metodo de Holt-Winters. Aunque hemos calculado que la serie sigue un esquema multiplicativo, realizaremos el ajuste usando el metodo de Holt-Winters aditivo tambien, para ver si el ajuste fuera mas preciso.

Analisis con Holt-Winters aditivo

Realizamos el ajuste mediante la funcion hw asumiendo una estacionalidad aditiva con un periodo anual (12 meses):

```r
fit_gasto <- hw(insample,h=12,seasonal="additive")
fit_gasto$model
```

```
## Holt-Winters' additive method
##
## Call:
##  hw(y = insample, h = 12, seasonal = "additive")
```

```
##
##   Smoothing parameters:
##     alpha = 0.8542
##     beta  = 1e-04
##     gamma = 2e-04
##
##   Initial states:
##     l = 367.2688
##     b = 5.4342
##     s = -40.1509 -129.7777 -241.9864 -243.1255 -228.2577 -227.2252
##            39.35 162.0713 436.1757 373.8142 74.8907 24.2215
##
##   sigma:  53.7343
##
##      AIC     AICc      BIC
## 1004.159 1014.030 1044.653
```

Este modelo aditivo asume que la observacion en un instante x, x_t se describe como:

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

Partiendo de unas condiciones iniciales (en t = 0), el nivel, la tendencia y la componente estacional se actualizan en cada instante t teniendo en cuenta la observación xt. En este ejemplo, las ecuaciones de actualización son:

$$L_t = 0.8542(x_t - S_{t-12}) + (1 - 0.7218)(L_{t-1} + T_{t-1})$$

$$T_t = 0.0001(L_t - L_{t-1}) + (1 - 0.0085)T_{t-1}$$

$$S_t = 0.0002(x_t - L_t) + (1 - 0.0001)S_{t-12}$$

Y en este caso las condiciones iniciales son:

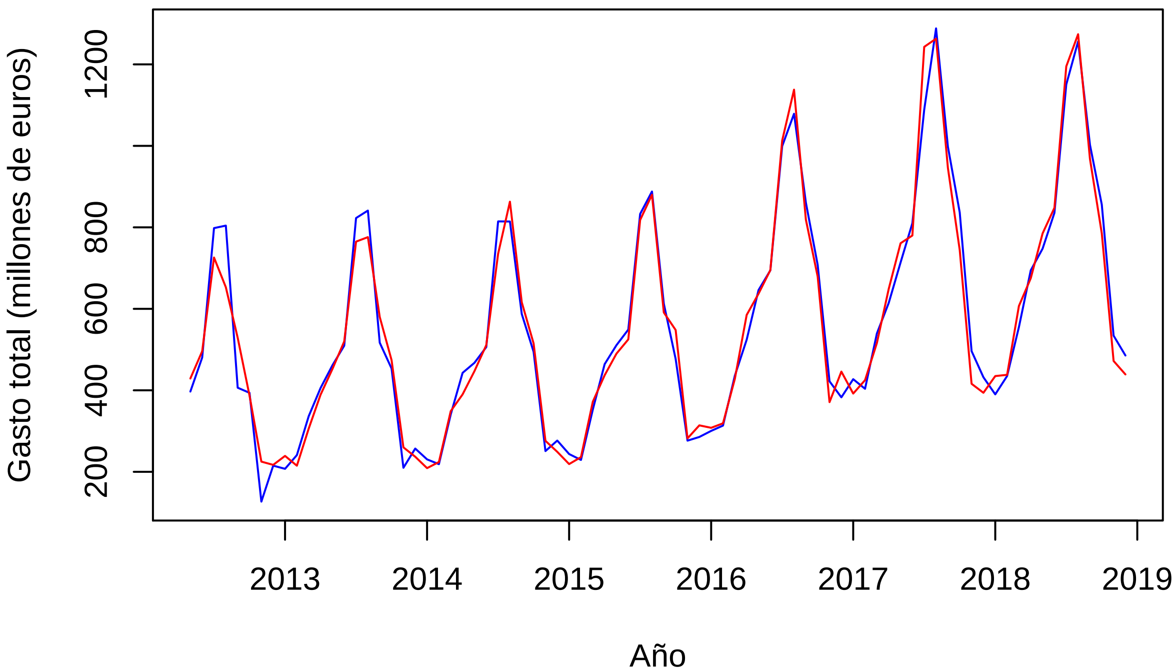
$$L_0 = 367.2688$$

$$T_0 = 5.4342$$

$$(S_{-11}, S_{-10}, S_{-9}, S_{-8}, S_{-7}, S_{-6}, S_{-5}, S_{-4}, S_{-3}, S_{-2}, S_{-1}, S_0) = (-40.1509, -129.7777, -241.9864, -243.1255, -228.2577, -22$$

A continuacion, representamos la serie real (en rojo) junto con la serie estimada mediante dicho metodo (en azul) para valorar el resultado del ajuste obtenido:

```r
fitval <- fit_gasto$fitted # serie de valores ajustados
plot(fitval,col="blue",ylab="Gasto total (millones de euros)", xlab = 'Año')
lines(insample, col = 'red')
```



Calculamos tambien la raiz del error cuadratico medio y del error absoluto porcentual medio para tener un valor numerico de la bondad del ajuste:

```
rmse <- sqrt(mean((insample-fitval)^2))
mape <- 100*mean(abs(insample-fitval)/insample)
rmse
```

```
## [1] 48.0614
```

```
mape
```

```
## [1] 7.338042
```

Analisis con Holt-Winters multiplicativo

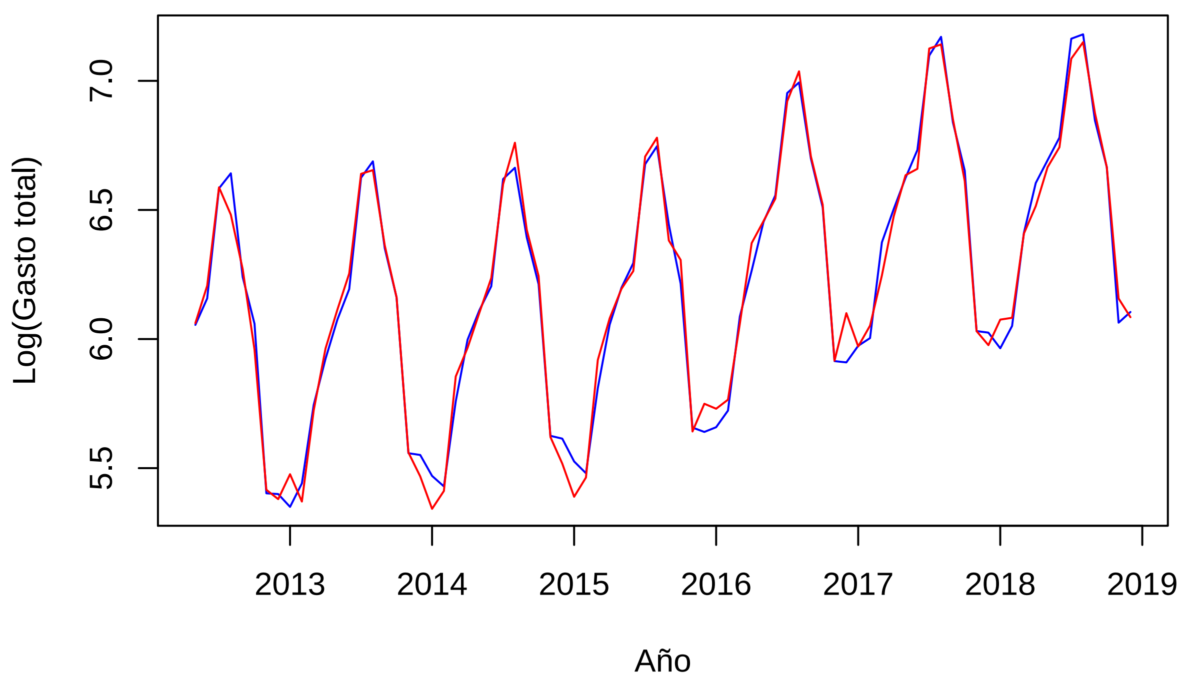
A continuacion realizamos el ajuste mediante el metodo Holt-Winters pero considerando un esquema multiplicativo, que es el que habiamos obtenido del analisis de los coeficientes de variacion. Para ello, hemos de aplicar la transformacion logaritmica a la serie y posteriormente aplicar el mismo procedimiento que antes pero con la nueva serie transformada.

```
loginsample <- log(insample) # Serie transformada

fitlog_gasto <- hw(loginsample,h=12,seasonal="additive")
fitlog_gasto$model # Vemos el modelo ajustado
```

```
## Holt-Winters' additive method
##
## Call:
##  hw(y = loginsample, h = 12, seasonal = "additive")
##
##   Smoothing parameters:
##     alpha = 0.5115
##     beta  = 0.0103
##     gamma = 1e-04
##
##   Initial states:
##     l = 5.9115
##     b = 0.0069
##     s = 0.0112 -0.1731 -0.5073 -0.5273 -0.4827 -0.4676
##            0.1425 0.3456 0.6712 0.6224 0.2287 0.1364
##
##   sigma:  0.0701
##
##       AIC      AICc       BIC
## -58.62718 -48.75621 -18.13272
```

A continuacion, representamos la serie original (rojo) y el ajuste obtenido (azul):

```
fitlogval <- fitlog_gasto$fitted # serie de valores ajustados
plot(fitlogval,col="blue",ylab="Log(Gasto total)", xlab = 'Año')
lines(loginsample, col = 'red')
```



Valoramos la bondad del ajuste mediante los siguientes valores estadisticos:

```r
rmse <- sqrt(mean((loginsample-fitlogval)^2))
mape <- 100*mean(abs(loginsample-fitlogval)/loginsample)
rmse
```

```
## [1] 0.06266619
```

```r
mape
```

```
## [1] 0.7579545
```

Como vemos dichos estadisticos que nos indican la bondad del ajuste son mucho mejores en el caso del esquema multiplicativo que en el aditivo, lo cual tiene sentido ya que es el esquema que habiamos calculado que sigue la serie mediante los coeficientes de variacion. Por lo tanto, el esquema multiplicativo es el que usaremos para realizar las predicciones de 2019 y 2019-2020.

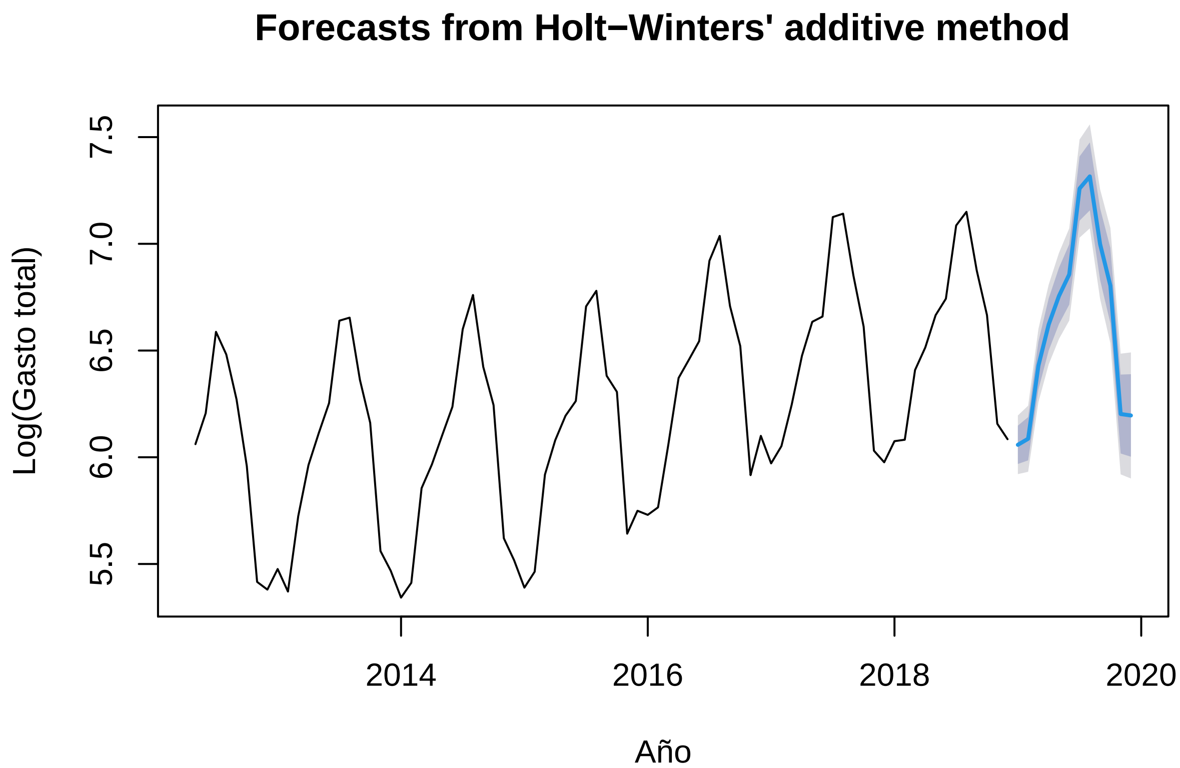
Prediccion 2019

Como hemos comentado anteriormente usaremos el ajuste obtenido para hacer una prediccion de los valores de 2019:

```r
fitlog_gasto
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2019       6.058307 5.968517 6.148096 5.920986 6.195627
## Feb 2019       6.086719 5.985440 6.187997 5.931826 6.241611
## Mar 2019       6.429388 6.317405 6.541371 6.258125 6.600651
## Apr 2019       6.622157 6.500040 6.744273 6.435396 6.808917
## May 2019       6.755812 6.623995 6.887629 6.554216 6.957408
## Jun 2019       6.856594 6.715414 6.997773 6.640679 7.072509
## Jul 2019       7.258678 7.108405 7.408951 7.028856 7.488501
## Aug 2019       7.315948 7.156798 7.475097 7.072550 7.559346
## Sep 2019       6.998789 6.830941 7.166637 6.742087 7.255491
## Oct 2019       6.804191 6.627791 6.980591 6.534410 7.073972
## Nov 2019       6.202512 6.017681 6.387343 5.919838 6.485187
## Dec 2019       6.195861 6.002701 6.389022 5.900448 6.491275
```

```r
plot(fitlog_gasto, ylab = 'Log(Gasto total)', xlab = 'Año')
```



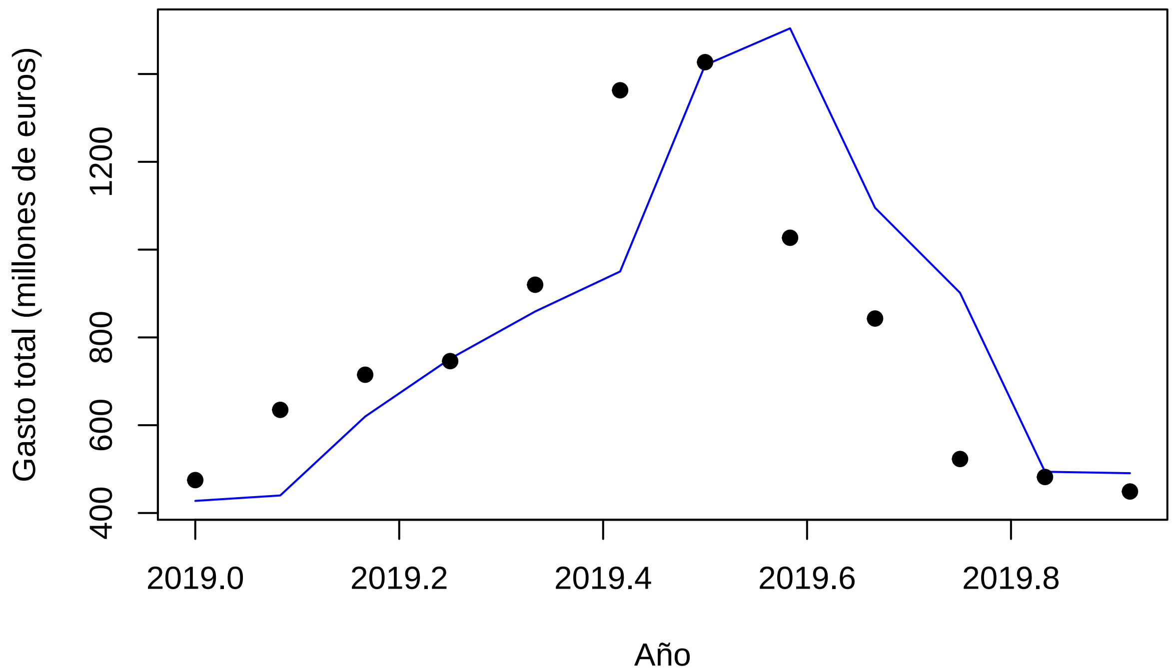
Las predicciones puntuales en la escala original podemos calcularlas realizando la transformación inversa (exponencial):

```
pred <- exp(fitlog_gasto$mean)
pred
```

```
##              Jan       Feb       Mar       Apr       May       Jun       Jul
## 2019   427.6506  439.9753  619.7945  751.5642  859.0370  950.1251 1420.3780
##              Aug       Sep       Oct       Nov       Dec
## 2019 1504.0969 1095.3060  901.6181  493.9885  490.7139
```

Representamos graficamente los datos reales de 2019 (puntos negros) junto con las predicciones realizadas por el ajuste (linea azul):

```
plot(pred,type="l",col="blue",xlab="Año", ylab = 'Gasto total (millones de euros)')
points(outsample_2019,pch=19)
```



A continuacion, calculamos la diferencia entre los valores reales y las predicciones:

```
diferencia_2019 <- outsample_2019 - pred
diferencia_2019
```

```
##              Jan         Feb         Mar         Apr         May         Jun
## 2019   47.349378  195.024663   95.205456   -5.564223   60.962971  412.874917
##              Jul         Aug         Sep         Oct         Nov         Dec
## 2019    6.621973 -477.096921 -252.306048 -378.618055  -11.988550  -41.713945
```

```
print(sum(diferencia_2019))
```

```
## [1] -349.2484
```

Por lo tanto, vemos que nuestro modelo ha predecido que en 2019 se iban a gasta 349.25 millones mas de los que se gastaron en realidad. Gracias a la grafica de las predicciones y los valores reales de 2019 observamos que nuestro ajuste predice que el gasto se produce un poco mas tarde que los meses en los que en realidad sucede, es decir, el ajuste predice el maximo gasto los meses de Julio Y Agosto cuando en realidad en 2019 el maximo gasto se produjo los meses de Junio y Julio. Ademas, predice menos gasto del que sucede (subestima) durante la primera mitad del año y mas del gasto real (sobreestima) durante la segunda mitad.

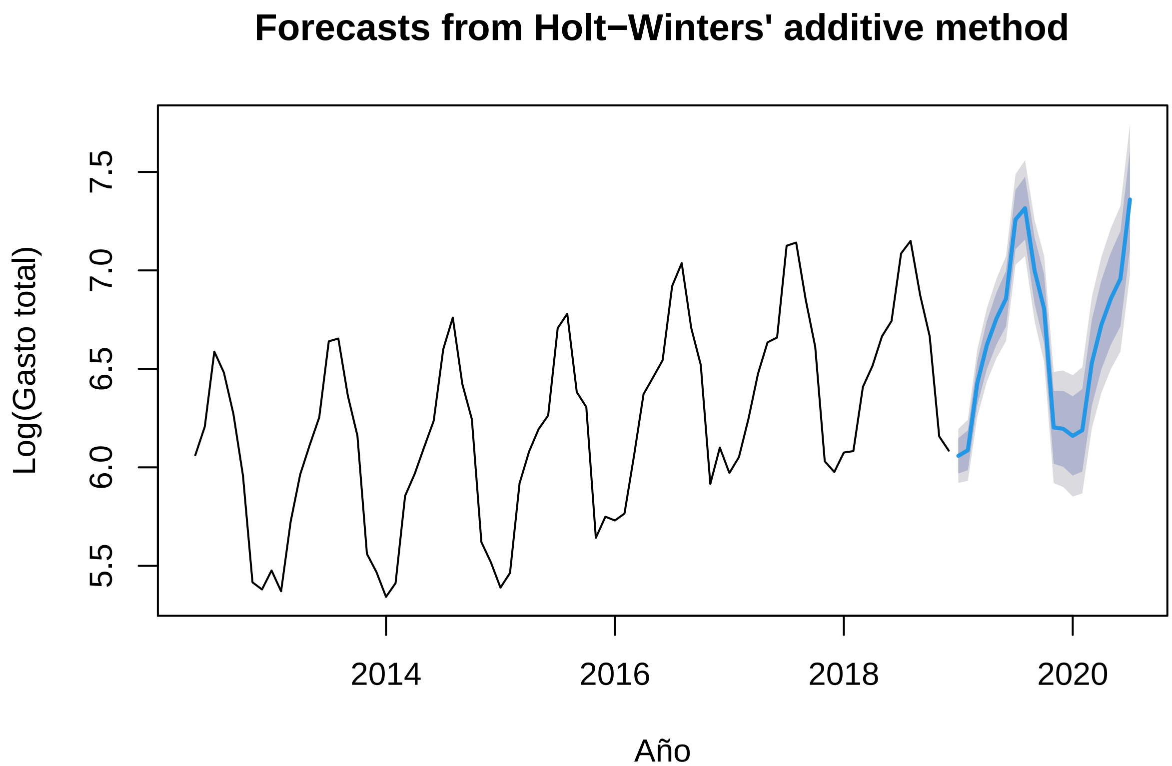
Prediccion 2019-2020

Como en el caso anterior, utilizamos el esquema multiplicativo ya que segun lo obtenido previamente con el que se obtiene el mejor ajuste. Repetimos el proceso, pero ahora predecimos 19 meses, es decir, 2019 y 2020 (hasta donde se dispone de datos:

```r
loginsample <- log(insample) # Serie transformada

fitlog_gasto <- hw(loginsample,h=19,seasonal="additive")
#fitlog_gasto$model

fitlogval <- fitlog_gasto$fitted # serie de valores ajustados

fitlog_gasto
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2019       6.058307 5.968517 6.148096 5.920986 6.195627
## Feb 2019       6.086719 5.985440 6.187997 5.931826 6.241611
## Mar 2019       6.429388 6.317405 6.541371 6.258125 6.600651
## Apr 2019       6.622157 6.500040 6.744273 6.435396 6.808917
## May 2019       6.755812 6.623995 6.887629 6.554216 6.957408
## Jun 2019       6.856594 6.715414 6.997773 6.640679 7.072509
## Jul 2019       7.258678 7.108405 7.408951 7.028856 7.488501
## Aug 2019       7.315948 7.156798 7.475097 7.072550 7.559346
## Sep 2019       6.998789 6.830941 7.166637 6.742087 7.255491
## Oct 2019       6.804191 6.627791 6.980591 6.534410 7.073972
## Nov 2019       6.202512 6.017681 6.387343 5.919838 6.485187
## Dec 2019       6.195861 6.002701 6.389022 5.900448 6.491275
## Jan 2020       6.159756 5.958348 6.361163 5.851729 6.467782
## Feb 2020       6.188168 5.978586 6.397750 5.867639 6.508696
## Mar 2020       6.530837 6.313139 6.748535 6.197897 6.863778
## Apr 2020       6.723606 6.497840 6.949372 6.378327 7.068885
## May 2020       6.857261 6.623468 7.091055 6.499705 7.214818
## Jun 2020       6.958043 6.716254 7.199832 6.588258 7.327827
## Jul 2020       7.360127 7.110368 7.609887 6.978154 7.742101
```

```r
plot(fitlog_gasto, ylab = 'Log(Gasto total)', xlab = 'Año')
```



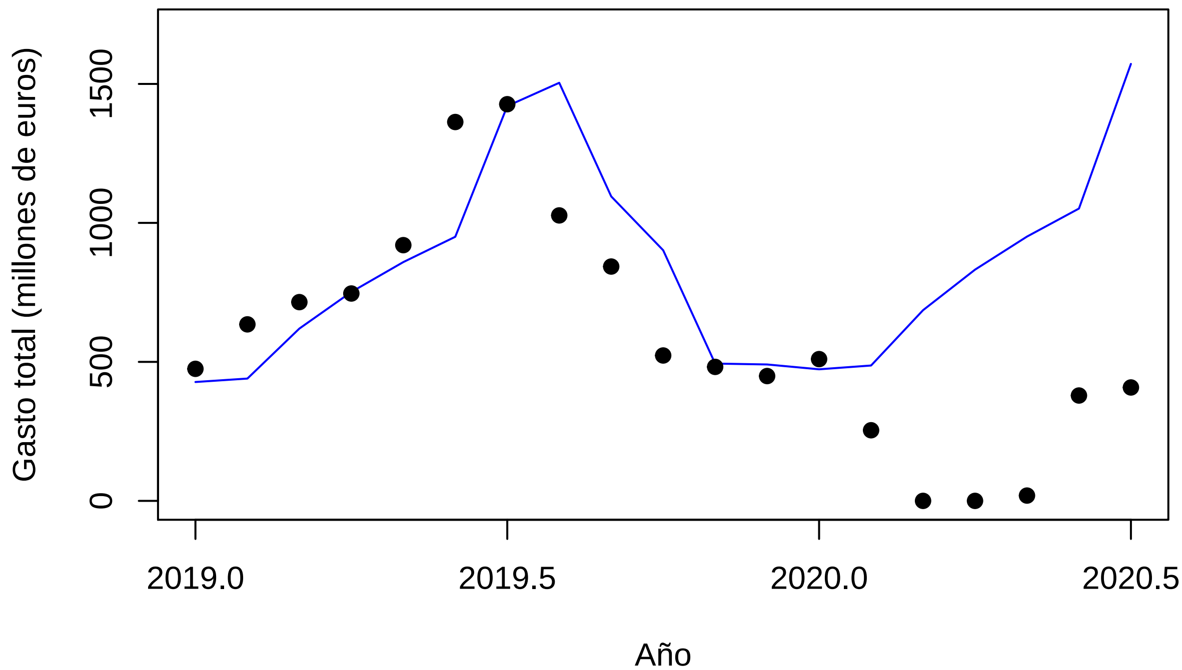
Las predicciones puntuales en la escala original podemos calcularlas, al igual que antes, realizando la transformación inversa (exponencial):

```
pred <- exp(fitlog_gasto$mean)
pred
```

```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2019  427.6506  439.9753  619.7945  751.5642  859.0370  950.1251 1420.3780
## 2020  473.3124  486.9531  685.9722  831.8115  950.7595 1051.5734 1572.0369
##            Aug       Sep       Oct       Nov       Dec
## 2019 1504.0969 1095.3060  901.6181  493.9885  490.7139
## 2020
```

Representamos graficamente los datos reales de 2019 (puntos negros) junto con las predicciones realizadas por el ajuste (linea azul):

```
plot(pred,type="l",col="blue",xlab="Año", ylab = 'Gasto total (millones de euros)', ylim=c(0,1700))
points(outsample_20192020,pch=19)
```



Igual que hemos hecho para la prediccion de 2019, calculamos las diferencias entre la prediccion del modelo y los datos reales:

```
diferencia_20192020 <- outsample_20192020 - pred
diferencia_20192020
```

```
##                Jan          Feb          Mar          Apr          May
## 2019     47.349378   195.024663    95.205456    -5.564223    60.962971
## 2020     36.687575  -232.953094  -685.972247  -831.811452  -931.759518
##                Jun          Jul          Aug          Sep          Oct
## 2019    412.874917     6.621973  -477.096921  -252.306048  -378.618055
## 2020   -672.573373 -1164.036923
##                Nov          Dec
## 2019    -11.988550   -41.713945
## 2020
```

```
print(sum(diferencia_20192020))
```

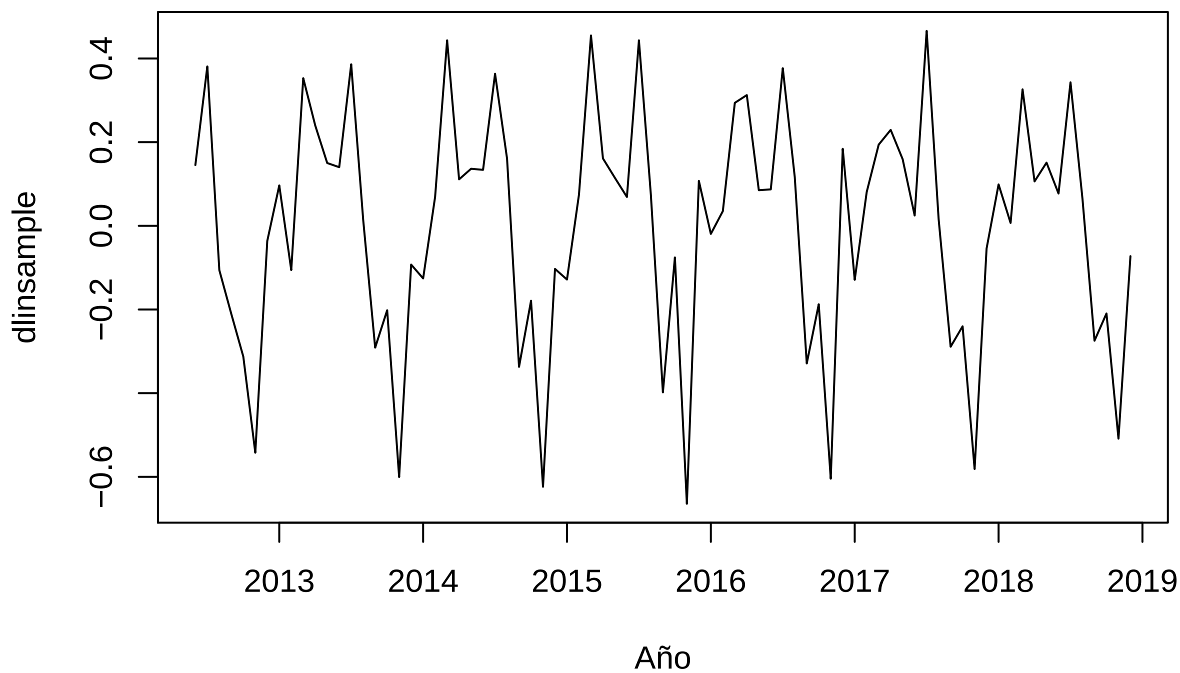
```
## [1] -4831.667
```

Como vemos en este caso el modelo ha predicho un gasto entre 2019 - Julio 2020 de 4831.67 millones mayor del que se ha producido en realidad segun los datos. Esto es debido al Covid-19, ya que la pandemia y el confinamiento durante los meses de Marzo, Abril y Mayo ha supuesto que el gasto realizado durante estos meses sea practicamente 0, cosa que el modelo obviamente no es capaz de predecir.

Analisis mediante metodologia de Box-Jenkins

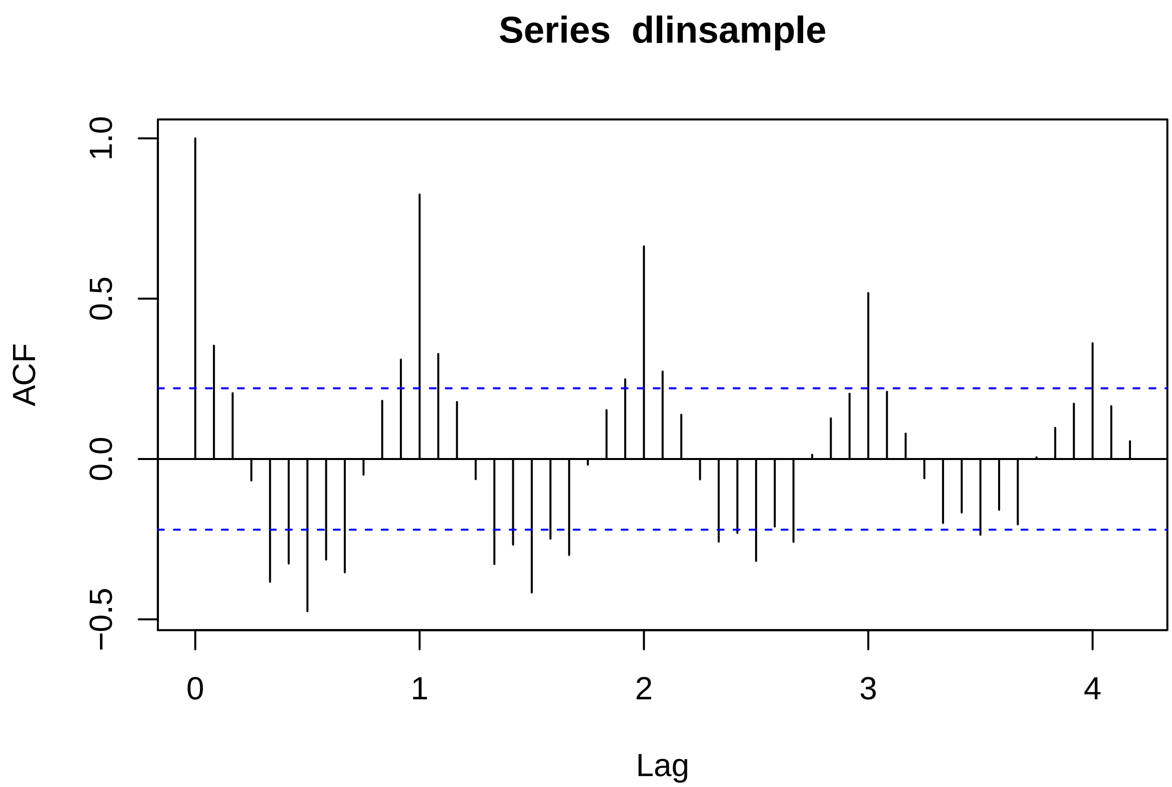
Dada nuestra serie que presenta estacionalidad y tendencia en primer hemos de obtener la serie estacionaria. En primer lugar quitamos la heterocedasticidad mediante el logaritmo y la tendencia diferenciando una vez ($d = 1$):

```
dlinsample <- diff(log(insample))
plot(dlinsample, xlab = 'Año')
```



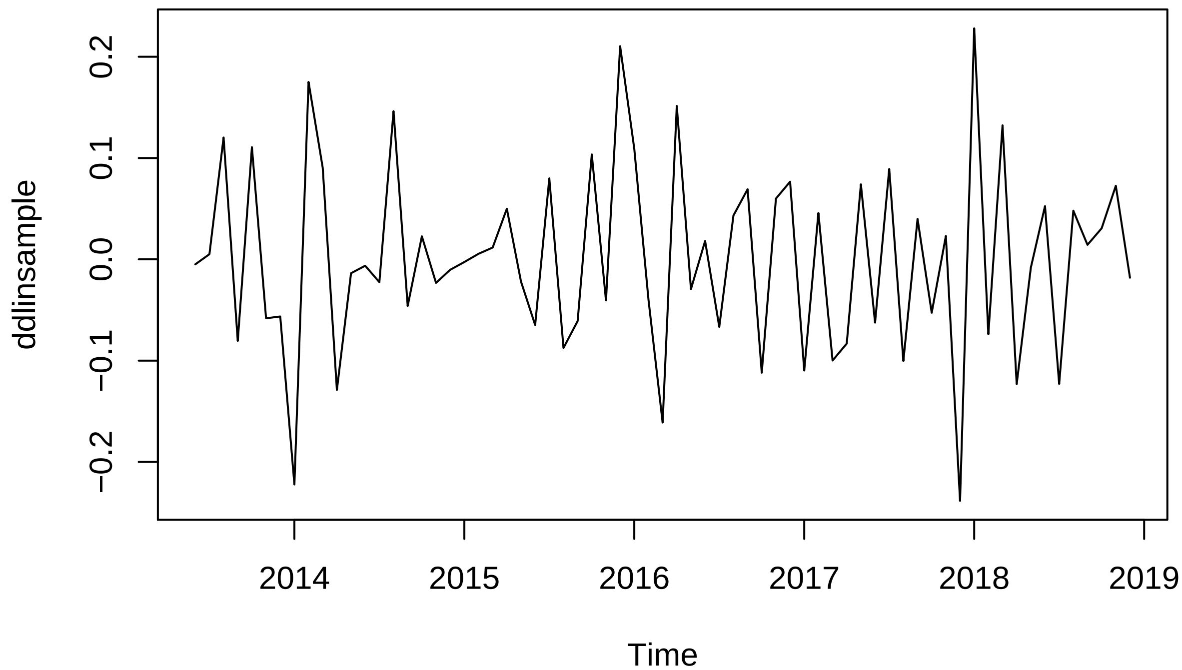
Como vemos, hemos eliminado la heterocedasticidad y la tendencia, pero seguimos teniendo presente la estacionalidad, que se puede observar claramente en la funcion de autocorrelacion:

```
acf(dlinsample, 50)
```



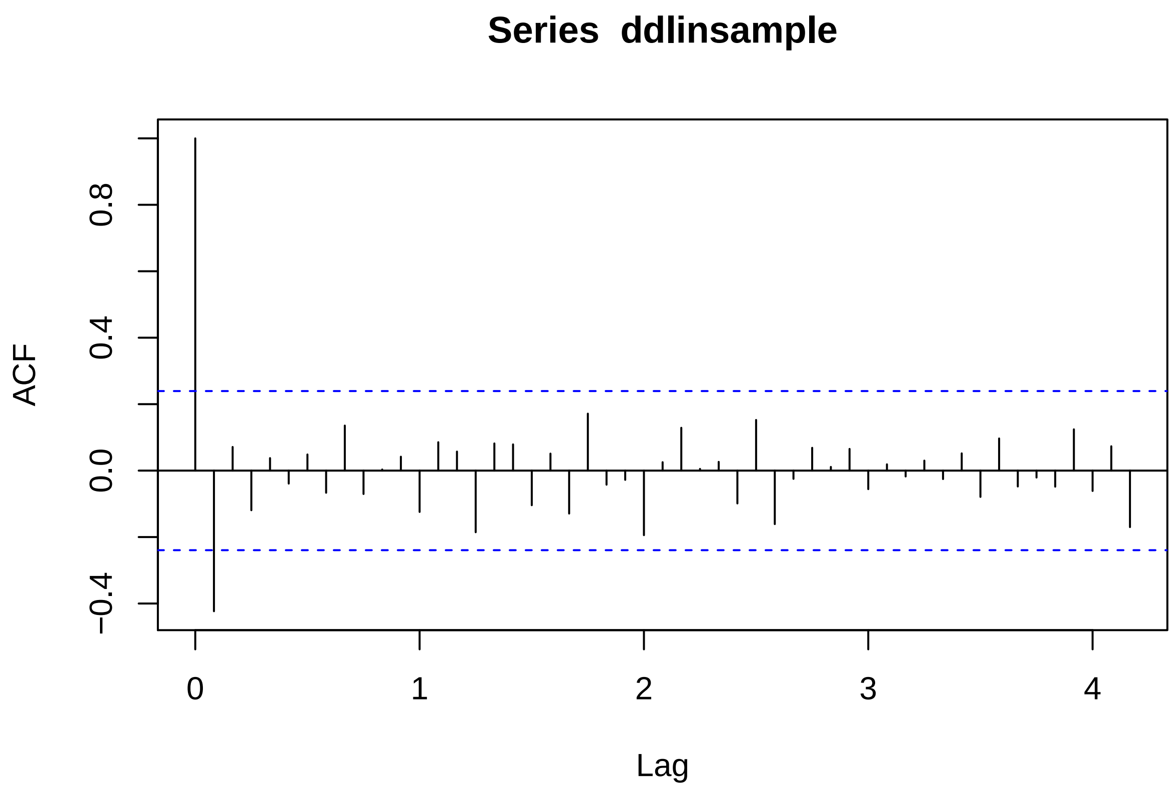
Eliminamos la estacionalidad, diferenciando con un periodo de $s = 12$ meses y, por lo tanto, $D = 1$:

```
ddlinsample <- diff(dlinsample, 12)
plot(ddlinsample)
```

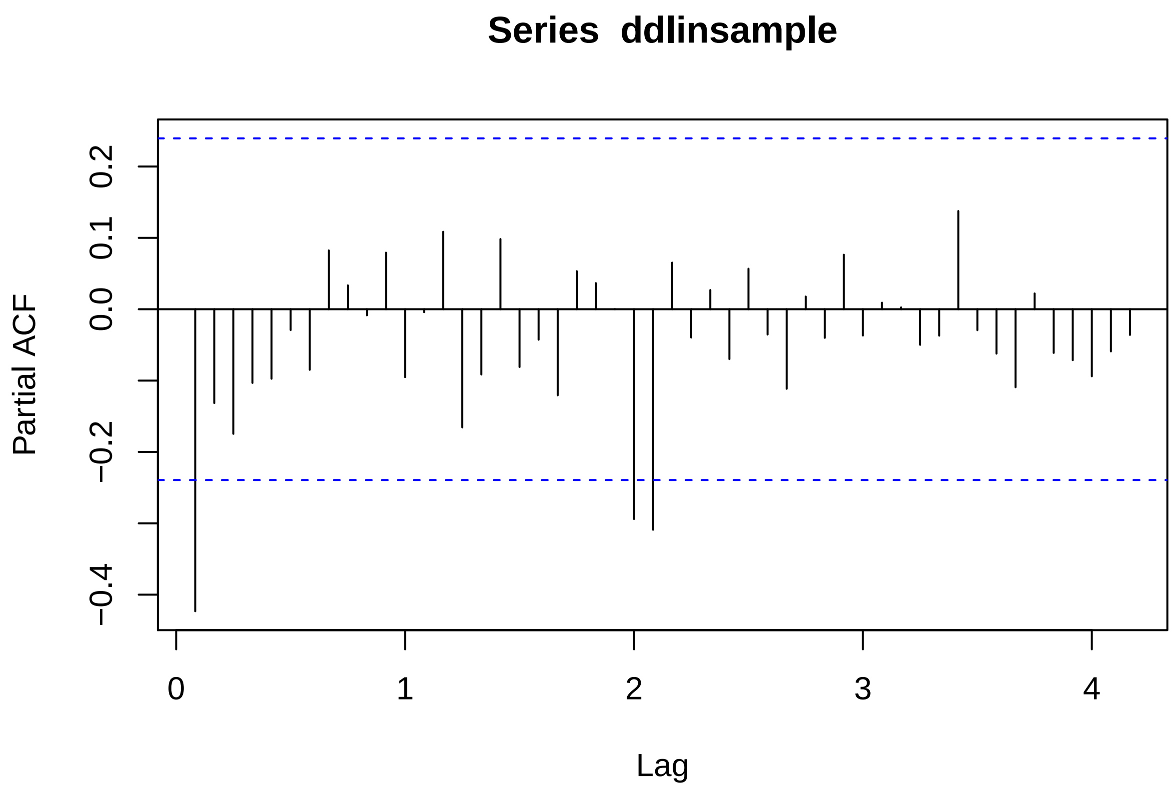


Para elegir los valores p, q, P y Q representamos las funciones de autorcorrelacion y de autocorrelacion parcial:

```
acf(ddlinsample, 50)
```




```
pacf(ddlinsample, 50)
```



Para la parte regular, de los primeros palos de ACF deducimos que $q = 0$, ya que tarda en llegar a 0. No obstante, no es descartable el caso $q = 1$ o incluso $q = 2$. De los primeros palos de PACF obtenemos que $p = 0$. En cuanto a la parte estacionaria, observando los palos correspondientes a s, 2s, 3s, 4s... en ACF y en PACF obtenemos que $P = 0$ y $Q = 0$ ya que en ambos casos decrece lentamente. Por lo tanto, segun nuestro analisis visual tendriamos un modelo ARIMA con $(p, d, q) = (0, 1, 0 - 2)$ y $(P, D, Q) = (0, 1, 0)$.

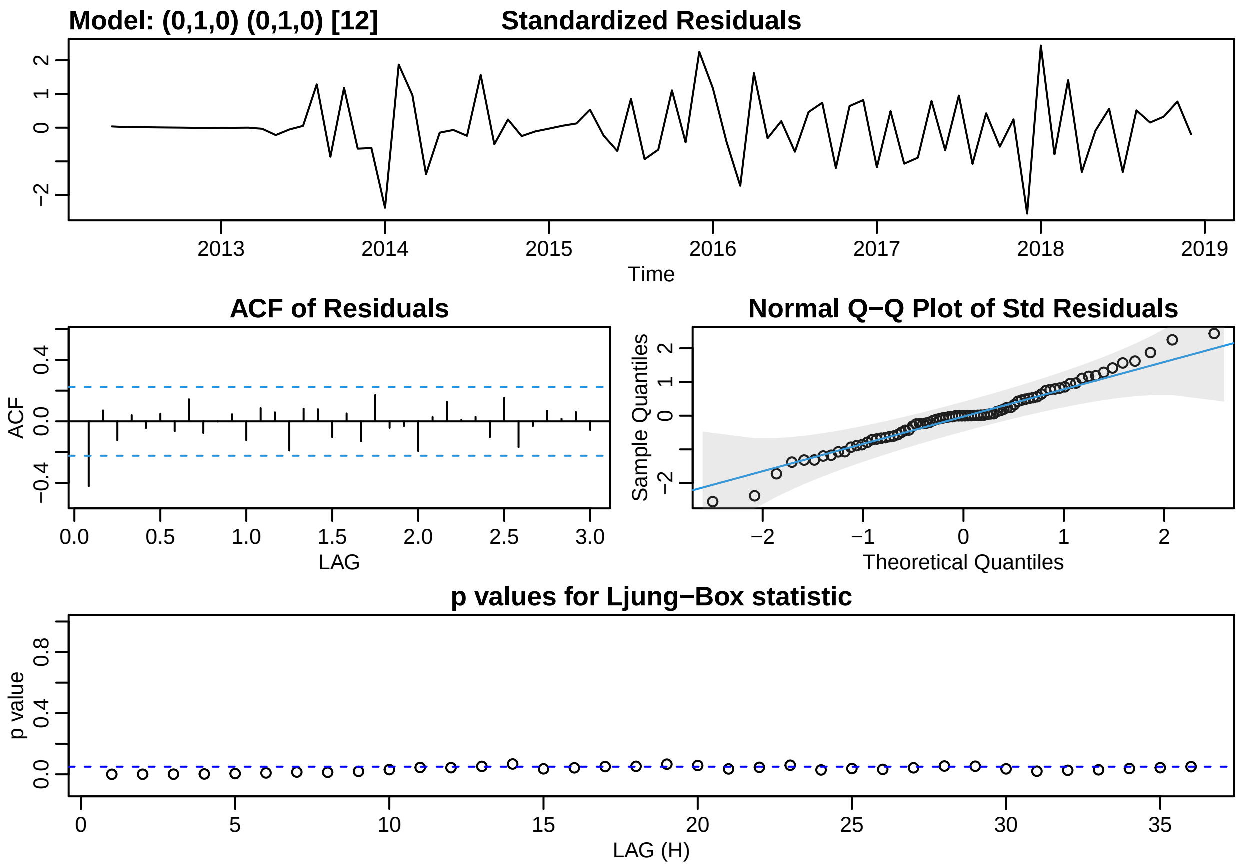
Para calcular que valor de q es el correcto, realizamos el ajuste para los distintos valores y obetenemos la bondad de cada ajuste.

Del modelo 1 (con $q = 0$) obtenemos:

```
#Model 1: p = 0, d = 1, q = 0, P = 0, D = 1, Q = 0
library(astsa)
```

```
##
## Attaching package: 'astsa'

## The following object is masked from 'package:forecast':
##
##     gas
```

```
model1 <- sarima(log(insample), 0, 1, 0, 0, 1, 0, 12)
```



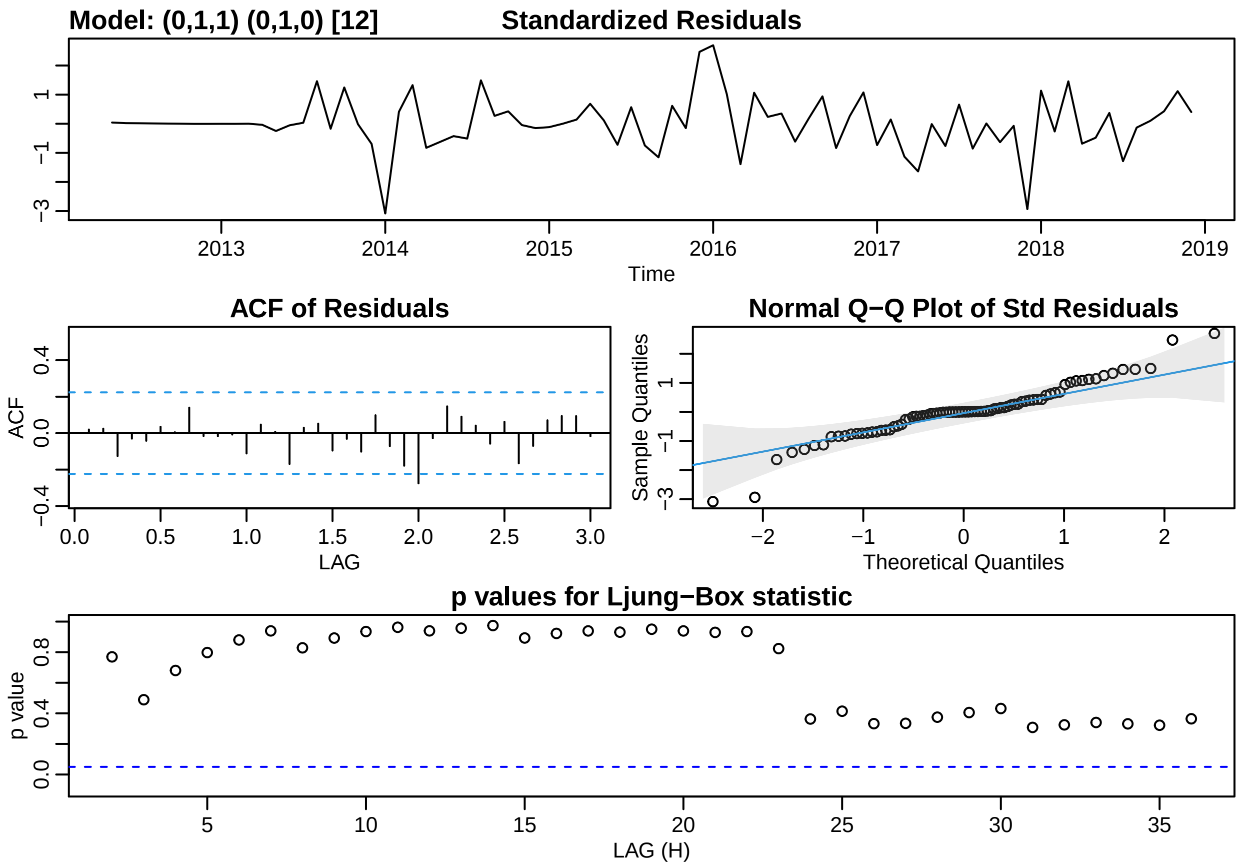

```
model1$AIC
```

```
## [1] -1.607557
```

Del modelo 2 (con $q = 1$) obtenemos:

```
#Model 2: p = 0, d = 1, q = 1, P = 0, D = 1, Q = 0
model2 <- sarima(log(insample), 0, 1, 1, 0, 1, 0, 12)
```

```
## initial  value -2.369606
## iter   2 value -2.485574
## iter   3 value -2.493962
## iter   4 value -2.495200
## iter   5 value -2.495319
## iter   6 value -2.495322
## iter   6 value -2.495322
## iter   6 value -2.495322
## final  value -2.495322
## converged
## initial  value -2.492811
## iter   2 value -2.492841
## iter   3 value -2.492842
## iter   3 value -2.492842
## iter   3 value -2.492842
## final  value -2.492842
## converged
```




```r
model2$ttable
```

```
##      Estimate    SE t.value p.value
## ma1   -0.5558 0.113 -4.9164       0
```

```r
model2$AIC
```

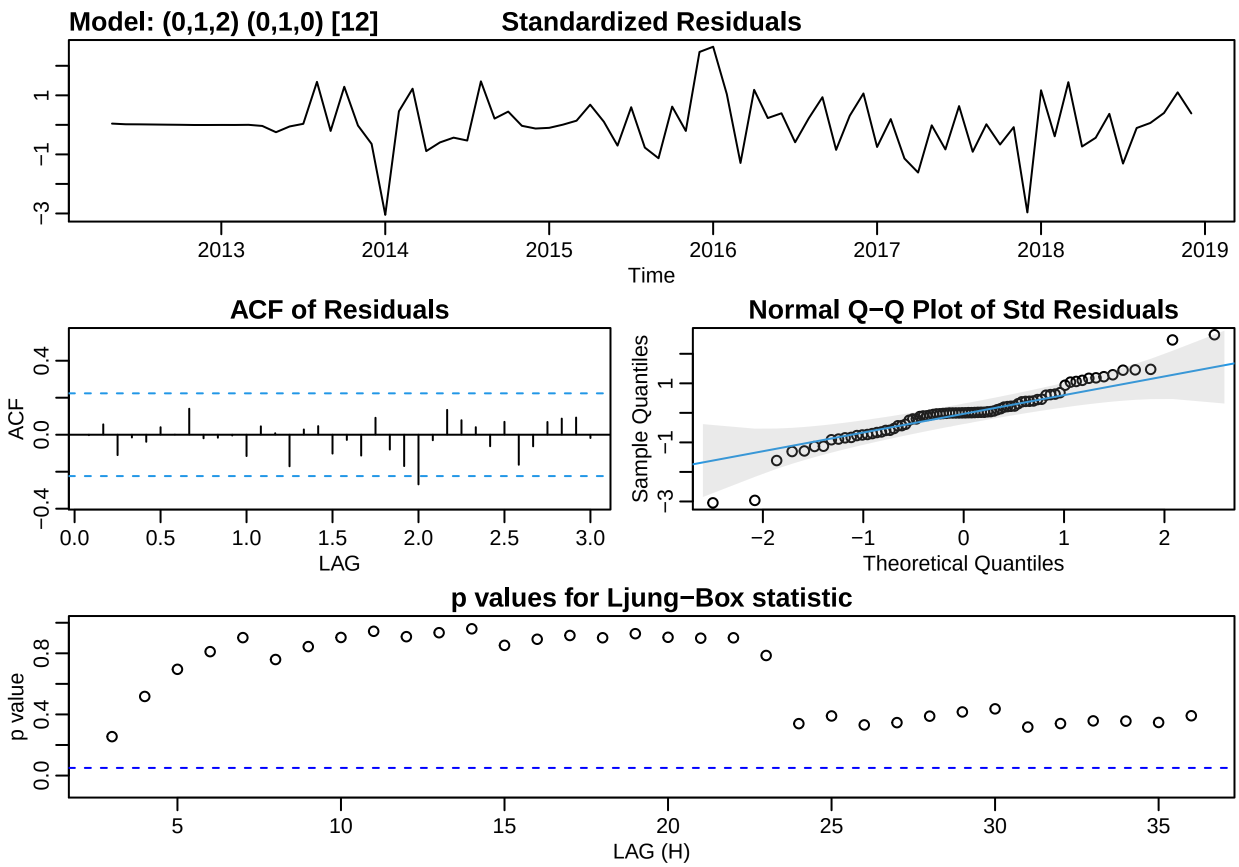
```
## [1] -1.79363
```

Del modelo 3 (con $q = 2$) obtenemos:

```r
#Model 3: p = 0, d = 1, q = 2, P = 0, D = 1, Q = 0
model3 <- sarima(log(insample), 0, 1, 2, 0, 1, 0, 12)
```

```
## initial  value -2.369606
## iter   2 value -2.477738
## iter   3 value -2.487750
## iter   4 value -2.492433
## iter   5 value -2.494852
## iter   6 value -2.496170
## iter   7 value -2.496170
## iter   8 value -2.496285
## iter   9 value -2.496285
## iter   9 value -2.496285
```

```
## iter   9 value -2.496285
## final  value -2.496285
## converged
## initial  value -2.493800
## iter   2 value -2.493807
## iter   3 value -2.493830
## iter   4 value -2.493830
## iter   4 value -2.493830
## iter   4 value -2.493830
## final  value -2.493830
## converged
```




```r
model3$ttable
```

```
##     Estimate     SE t.value p.value
## ma1  -0.5382 0.1167 -4.6113  0.0000
## ma2  -0.0401 0.1099 -0.3646  0.7166
```

```r
model3$AIC
```

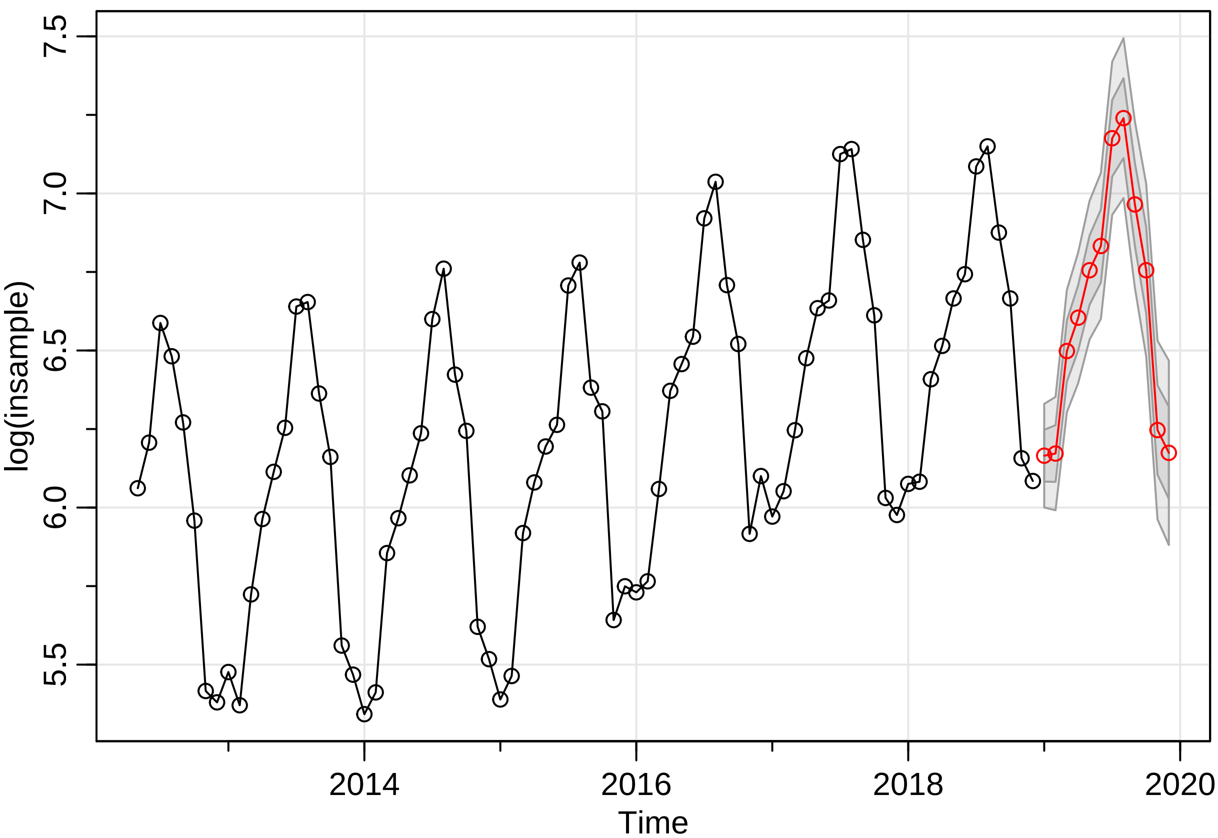
```
## [1] -1.769685
```

Como para los modelos 1 y 3 obtenemos un AIC mayor que para el modelo 2 directamente los descartamos y nos quedamos con el modelo 2, que ademas tiene un p-valor de 0. Dicho modelo tiene los siguientes parametros:$(p, d, q) = (0, 1, 1)$ y $(P, D, Q) = (0, 1, 0)$

Prediccion 2019

A continuacion realizamos la prediccion para el año 2019 mediante el modelo que acabamos de calcular, el cual corresponde con ARIMA(0,1,1)x(0,1,0)s=12:

```
logpred <- sarima.for(log(insample), 12, 0, 1, 1, 0, 1, 0, 12)
```




```
logpred$pred
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2019 6.165084 6.171957 6.498267 6.604451 6.755422 6.832618 7.175639 7.239655
##           Sep      Oct      Nov      Dec
## 2019 6.964970 6.755422 6.246717 6.174237
```

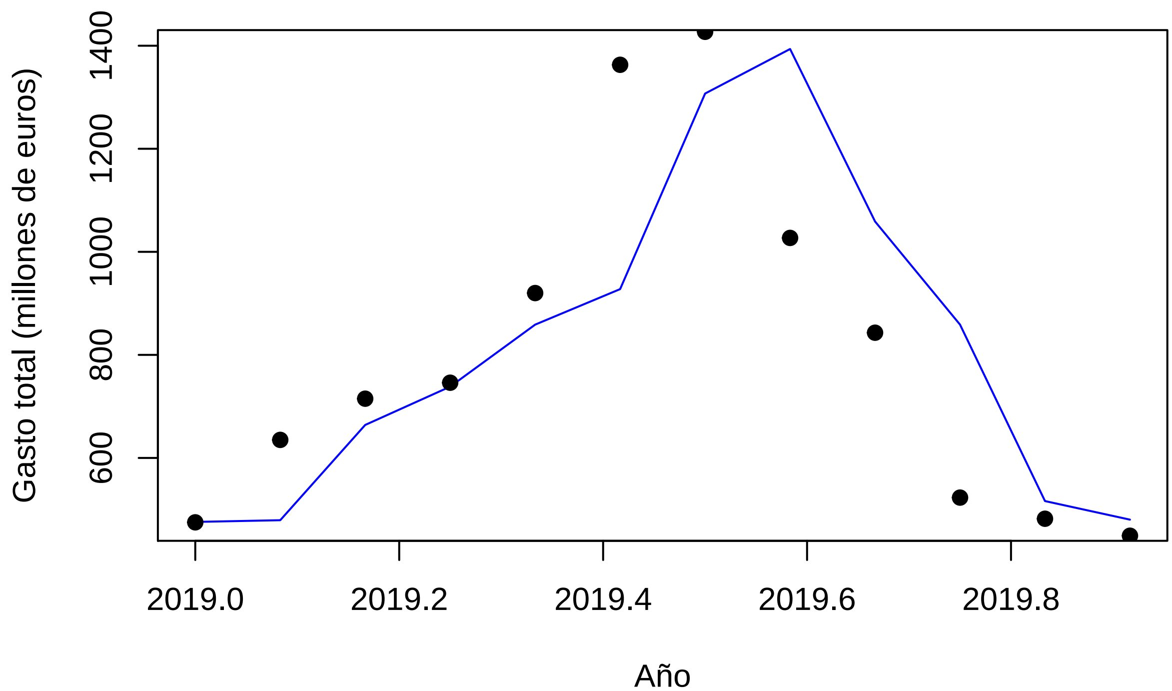
Para obtener los valores reales y no el log de ellos calculamos la exponencial:

```
predict <- exp(logpred$pred)
predict
```

```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2019  475.8410  479.1227  663.9897  738.3740  858.7017  927.6166 1307.1955
##            Aug       Sep       Oct       Nov       Dec
## 2019 1393.6126 1058.8831  858.7017  516.3149  480.2166
```

Representamos graficamente los datos reales de 2019 (puntos negros) junto con las predicciones realizadas por el ajuste (linea azul):

```r
plot(predict,type="l",col="blue",xlab="Año", ylab = 'Gasto total (millones de euros)')
points(outsample_2019,pch=19)
```




```r
diferencia_2_2019 <- outsample_2019 - predict
diferencia_2_2019
```

```
##              Jan        Feb        Mar        Apr        May        Jun
## 2019   -0.841048 155.877290  51.010308   7.625960  61.298339 435.383428
##              Jul        Aug        Sep        Oct        Nov        Dec
## 2019  119.804477 -366.612633 -215.883068 -335.701661  -34.314884  -31.216598
```

```r
print(sum(diferencia_2_2019))
```

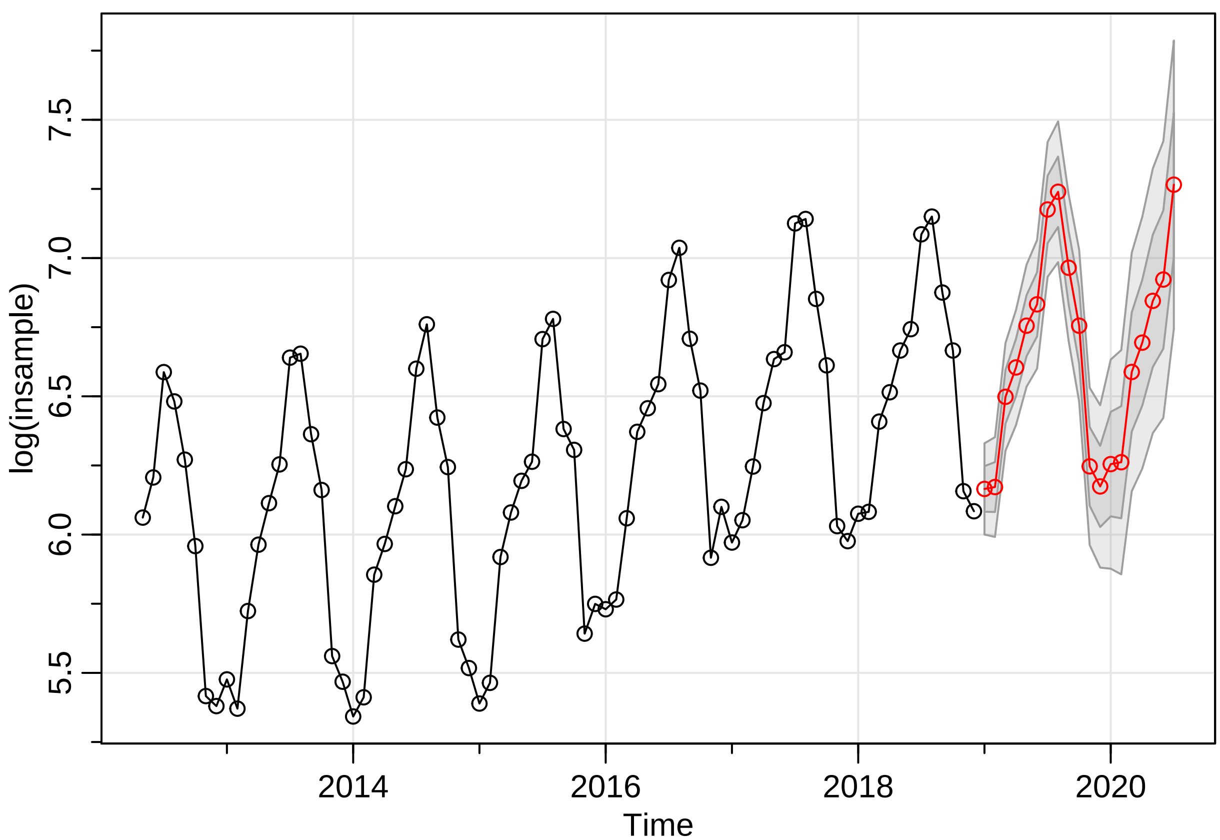
```
## [1] -153.5701
```

Por lo tanto, vemos que nuestro modelo ha predecido que en 2019 se iban a gasta 153.57 millones mas de los que se gastaron en realidad. Gracias a la grafica de las predicciones y los valores reales de 2019 observamos que nuestro ajuste predice que un gasto menor del realizado durante la primera mitad del año y mayor del realizado durante la segunda mitad. Ademas, el pico del maximo se prevee para un mes mas tarde de cuando sucede en realidad.

Prediccion 2019-2020

A continuacion realizamos la prediccion para el año 2019-2020 mediante el modelo que acabamos de calcular, el cual corresponde con ARIMA(0,1,1)x(0,1,0)s=12:

```
logpred_2020 <- sarima.for(log(insample), 19, 0, 1, 1, 0, 1, 0, 12)
```




```
logpred_2020$pred
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2019 6.165084 6.171957 6.498267 6.604451 6.755422 6.832618 7.175639 7.239655
## 2020 6.254822 6.261695 6.588004 6.694188 6.845159 6.922356 7.265377
##           Sep      Oct      Nov      Dec
## 2019 6.964970 6.755422 6.246717 6.174237
## 2020
```

Para obtener los valores reales y no el log de ellos calculamos la exponencial:

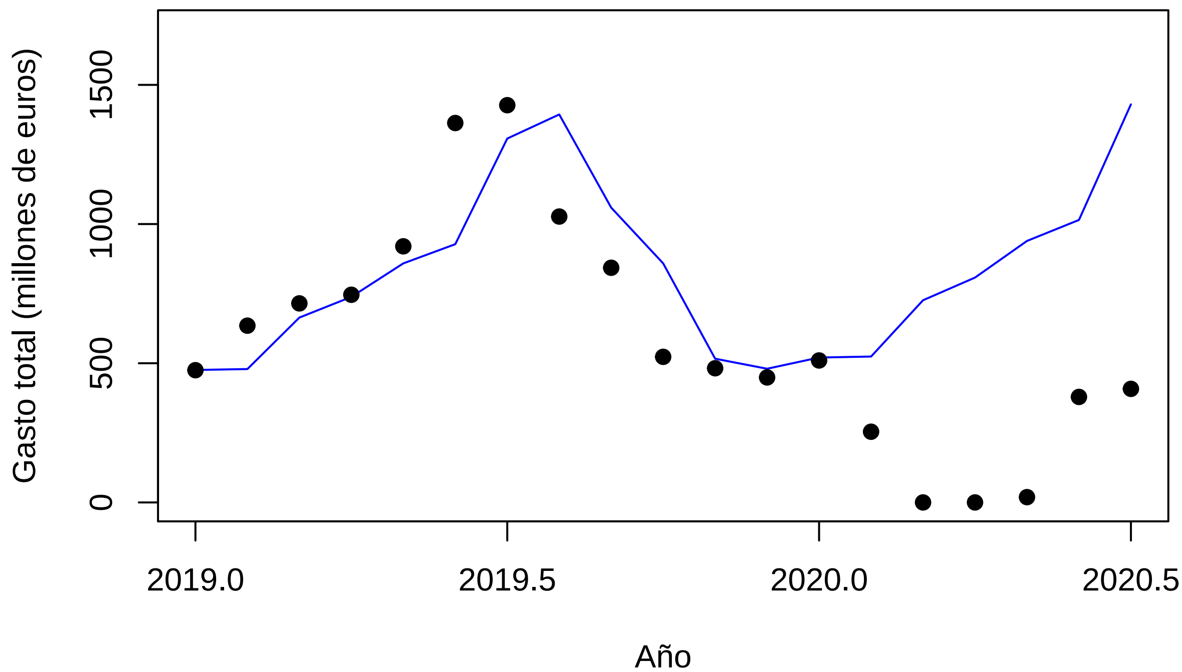
```
predict_2020 <- exp(logpred_2020$pred)
predict_2020
```

```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2019  475.8410  479.1227  663.9897  738.3740  858.7017  927.6166 1307.1955
## 2020  520.5166  524.1063  726.3300  807.6981  939.3230 1014.7081 1429.9248
```

```
##            Aug       Sep       Oct       Nov       Dec
## 2019 1393.6126 1058.8831  858.7017  516.3149  480.2166
## 2020
```

Representamos graficamente los datos reales de 2019 (puntos negros) junto con las predicciones realizadas por el ajuste (linea azul):

```r
plot(predict_2020,type="l",col="blue",xlab="Año", ylab = 'Gasto total (millones de euros)', ylim=c(0,17
points(outsample_20192020,pch=19)
```




```r
diferencia_2_2020 <- outsample_20192020 - predict_2020
diferencia_2_2020
```

```
##              Jan          Feb          Mar          Apr          May
## 2019   -0.841048   155.877290    51.010308     7.625960    61.298339
## 2020  -10.516559  -270.106328  -726.330002  -807.698108  -920.322985
##              Jun          Jul          Aug          Sep          Oct
## 2019  435.383428   119.804477  -366.612633  -215.883068  -335.701661
## 2020 -635.708142 -1021.924799
##              Nov          Dec
## 2019  -34.314884   -31.216598
## 2020
```

```r
print(sum(diferencia_2_2020))
```

```
## [1] -4546.177
```

Como vemos en este caso el modelo ha predicho un gasto entre 2019 - Julio 2020 de 4546.18 millones mayor del que se ha producido en realidad segun los datos. Esto es debido al Covid-19, ya que la pandemia y el confinamiento durante los meses de Marzo, Abril y Mayo ha supuesto que el gasto realizado durante estos meses sea practicamente 0, cosa que el modelo obviamente no es capaz de predecir.

Conclusiones

Como podemos observar, el ajuste mediante el modelo ARIMA es mejor que mediante el suavizado exponencial. Para la prediccion del año 2019, el modelo ARIMA predice un gasto 154 millones mayor que el que se produjo en realidad, mientras que mediante el suavizado exponencial esa diferencia en la prediccion aumenta hasta los 349 millones. Para el caso de 2019-Julio 2020 es obvio que ambos modelos van a fallar debido a la situacion mundial provocada por la pandemia del Covid-19 que desde Marzo redujo practicamente a cero el numero de turistas. No obstante, tambien en este caso la prediccion del modelo ARIMA se aleja menos de las cifras reales que la prediccion del modelo realizado con el suavizado exponencial.