Predicción de las ventas de Apple

Beatriz Quevedo

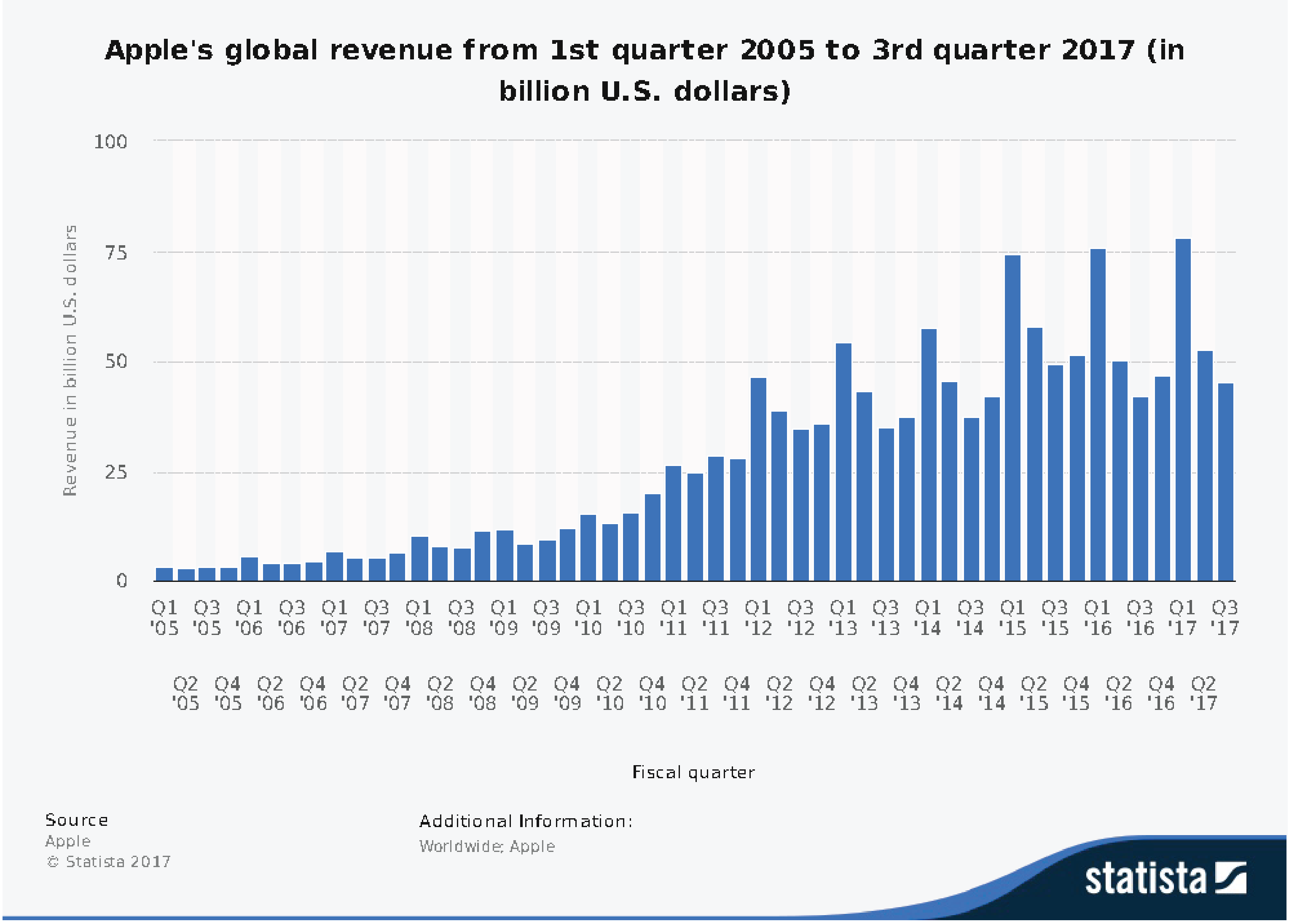
2020-11-16

Introducción

El objetivo del presente trabajo es predecir las ventas de Apple. Para ello, se ha acudido a Bloomberg para obtener los datos trimestrales desde el 2T de 2008 hasta el 3T del 2017.

Se debe elegirá el modelo ETS y el modelo ARIMA que mejor prediga las ventas, habiendo dejado fuera de la estimación los trimestres del 2017.

Una vez seleccionado el modelo, se estimará el modelo con todos los datos y se harán las predicciones del año 2017 y 2018.



Ingresos desde el 1º trimestre del 2005 hasta el 31 del 2017

Importación de librerías y datos

```
library(tidyverse)
library(readr)
library(skimr)
library(janitor)
library(forecast)
library(magrittr)
library(xts)
library(ggplot2)
library(ggfortify)
```

```
apple <-  read.csv("../data/IngresosApple.csv", sep = ";")
head(apple)
```

	Trimestre <chr>	Ingresos <int>
1	Q2 2008	7980
2	Q3 2008	7561
3	Q4 2008	11520

	Trimestre <chr>	Ingresos <int>
4	Q1 2009	11880
5	Q2 2009	9084
6	Q3 2009	9734

6 rows

Transformación e interpretación del DataFrame

Primero, se genera una variable xts para poder transformarla por trimestres y en *zoo data*.

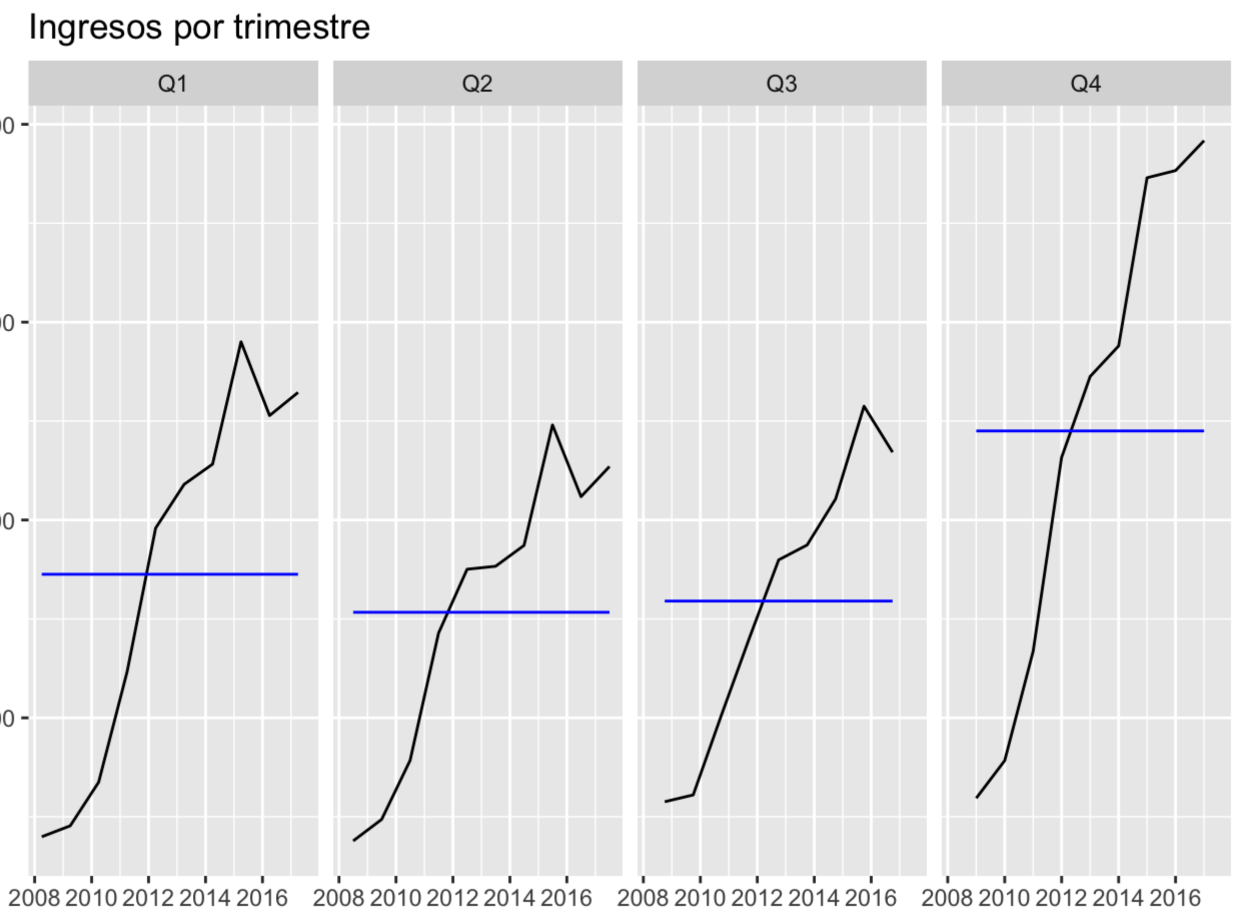
```
rawData <- seq(as.Date("2008/04/01"), as.Date("2017/09/30"), by = "quarter")

xApple <- xts(apple$Ingresos, order.by = rawData)
xApple <- to.quarterly(xApple)

zApple=as.zoo(xApple$xApple.Close)
```

Se representan los trimestres para poder observar su comportamiento individual a lo largo de los años del estudi (2008-2018) Nótese que la línea azul horizontal es la **media de ingreso por trimestre**.

```
tApple <- ts(coredata(zApple), start = c(2008,2), frequency = 4)
ggfreqplot(tApple, freq = 4, nrow = 1, facet.labeller = c('Q1','Q2','Q3','Q4')) + ggtitle('Ingr
esos por trimestre')
```

```
## Warning: `group_by_()` is deprecated as of dplyr 0.7.0.
## Please use `group_by()` instead.
## See vignette('programming') for more help
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

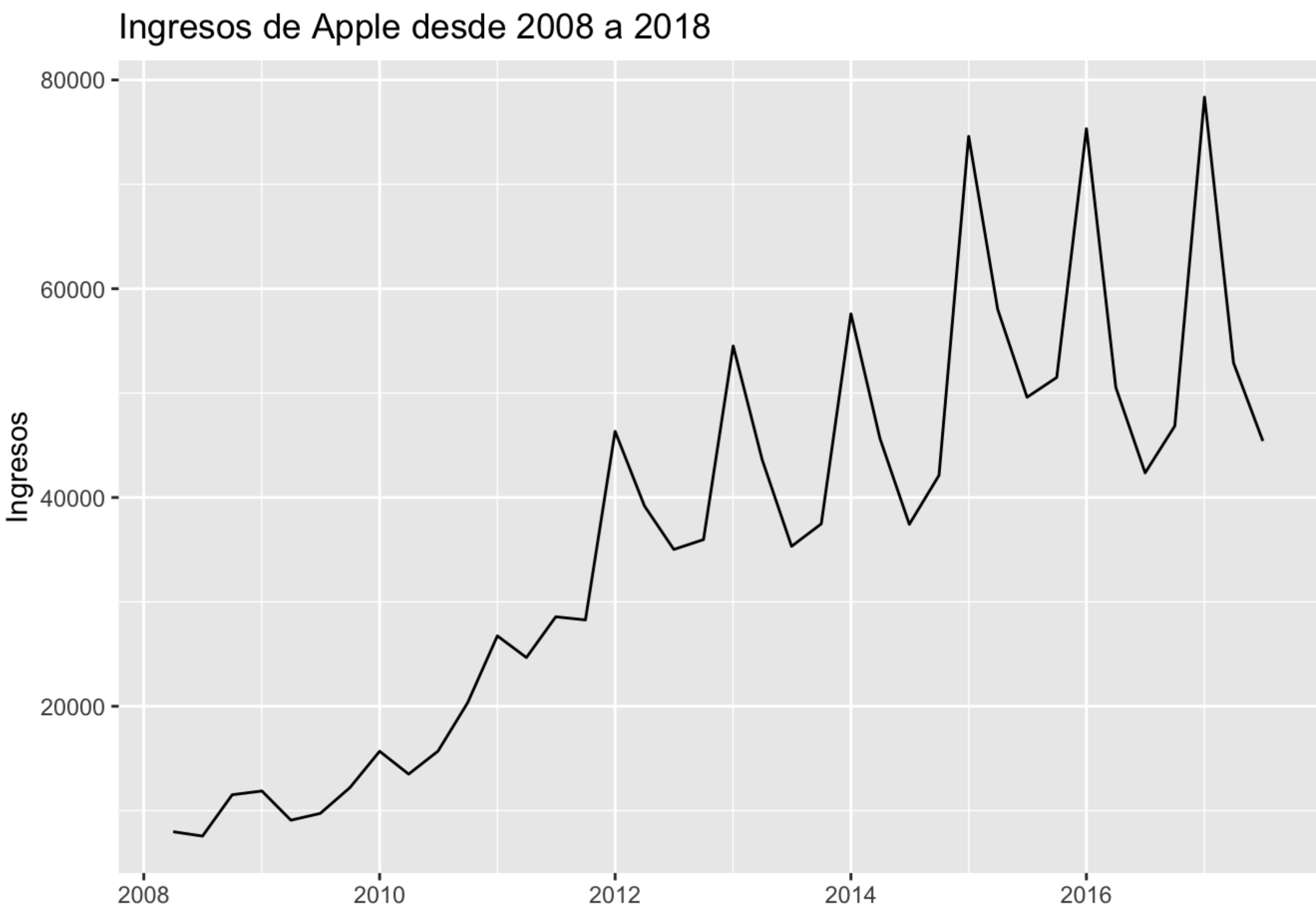
```
## Warning: `summarise_()` is deprecated as of dplyr 0.7.0.
## Please use `summarise()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```



Se aprecia que el cuarto trimesrte tiene los **ingresos más altos del año.**

También se puede observar la evolución en una única línea temporal.

```
autoplot(tApple)+ggtitle('Ingresos de Apple desde 2008 a 2018')+
  xlab('')+ylab('Ingresos')
```



Por lo tanto, estas observaciones probienen de una distribución que es diferente en cada instante del tiempo, es decir, no es estacionaria, ya que como se aprecia en el gráfico de *ingresos de Aplle desde 2008 a 2018*, va en aumento a medida que aumentan los años. La variación del periodo 2008-2012 es diferente de la del periodo 2012-2018, por lo que no es estacionaria en varianza, y de media como se aprecia, tampoco. El modelo de serie temporal parece, por tanto, tener una **tendencia exponencial al alza con estacionalidad multiplicativa**

Componentes de la serie temporal

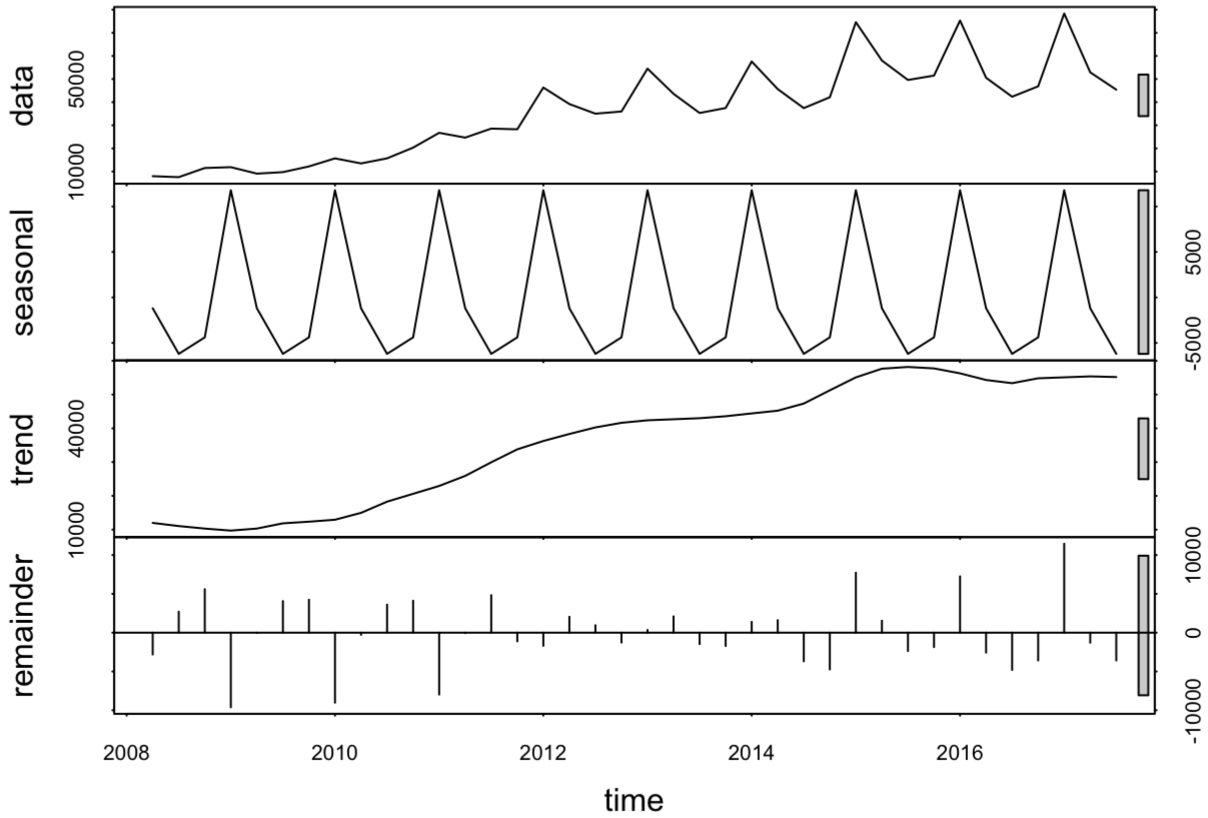
Se supone que la serie temporal es la suma de varias componentes = **tendencia** (*trend*), **estacionalidad** (*seasonal*) e **irregular** (*remainer*).

En la siguiente tabla se puede ver cada componente del DataFrame del estudio.

```
stl(tApple[, 1], s.window = "periodic")
```

```
##  Call:
##  stl(x = tApple[, 1], s.window = "periodic")
##
## Components
##          seasonal     trend   remainder
## 2008 Q2 -1191.515 11987.713 -2816.19786
## 2008 Q3 -6202.795 11054.647  2709.14756
## 2008 Q4 -4388.396 10306.954  5601.44193
## 2009 Q1 11782.720  9712.795 -9615.51515
## 2009 Q2 -1191.515 10313.378   -37.86329
## 2009 Q3 -6202.795 11865.051  4071.74373
## 2009 Q4 -4388.396 12348.763  4246.63276
## 2010 Q1 11782.720 12927.130 -9026.85026
## 2010 Q2 -1191.515 14968.422  -277.90712
## 2010 Q3 -6202.795 18282.820  3619.97453
## 2010 Q4 -4388.396 20608.629  4122.76624
## 2011 Q1 11782.720 22936.090 -7977.81022
## 2011 Q2 -1191.515 25917.203   -58.68777
## 2011 Q3 -6202.795 29940.698  4833.09640
## 2011 Q4 -4388.396 33777.770 -1119.37412
## 2012 Q1 11782.720 36249.530 -1699.24976
## 2012 Q2 -1191.515 38326.357  2051.15766
## 2012 Q3 -6202.795 40267.415   958.37982
## 2012 Q4 -4388.396 41633.253 -1278.85770
## 2013 Q1 11782.720 42379.047   350.23228
## 2013 Q2 -1191.515 42689.157  2105.35767
## 2013 Q3 -6202.795 42998.839 -1473.04409
## 2013 Q4 -4388.396 43579.104 -1718.70816
## 2014 Q1 11782.720 44414.890  1396.39005
## 2014 Q2 -1191.515 45222.672  1614.84282
## 2014 Q3 -6202.795 47330.414 -3695.61937
## 2014 Q4 -4388.396 51256.536 -4745.14062
## 2015 Q1 11782.720 55088.745  7727.53443
## 2015 Q2 -1191.515 57657.107  1544.40748
## 2015 Q3 -6202.795 58173.837 -2366.04237
## 2015 Q4 -4388.396 57747.384 -1857.98841
## 2016 Q1 11782.720 56283.309  7257.97053
## 2016 Q2 -1191.515 54318.201 -2569.68577
## 2016 Q3 -6202.795 53365.704 -4804.90925
## 2016 Q4 -4388.396 54818.937 -3578.54150
## 2017 Q1 11782.720 55104.114 11464.16602
## 2017 Q2 -1191.515 55382.281 -1294.76586
## 2017 Q3 -6202.795 55190.924 -3580.12898
```

```
plot(stl(tApple[, 1], s.window = "periodic"))
```



Selección del modelo ETS

Previamente se crará una muestra sin los últimos tres trimestres para poder probar el modelo predictivo más adelante.

```
cOmit=3
nObs = length(zApple)
oApple <- window(zApple,start=index(zApple[1]),end=index(zApple[nObs-cOmit]))
```

Se selecciona de manera automática el ETS y se crea un modelo de predicción con la muestra:

```
etsfit <- ets(oApple)
etsfit
```

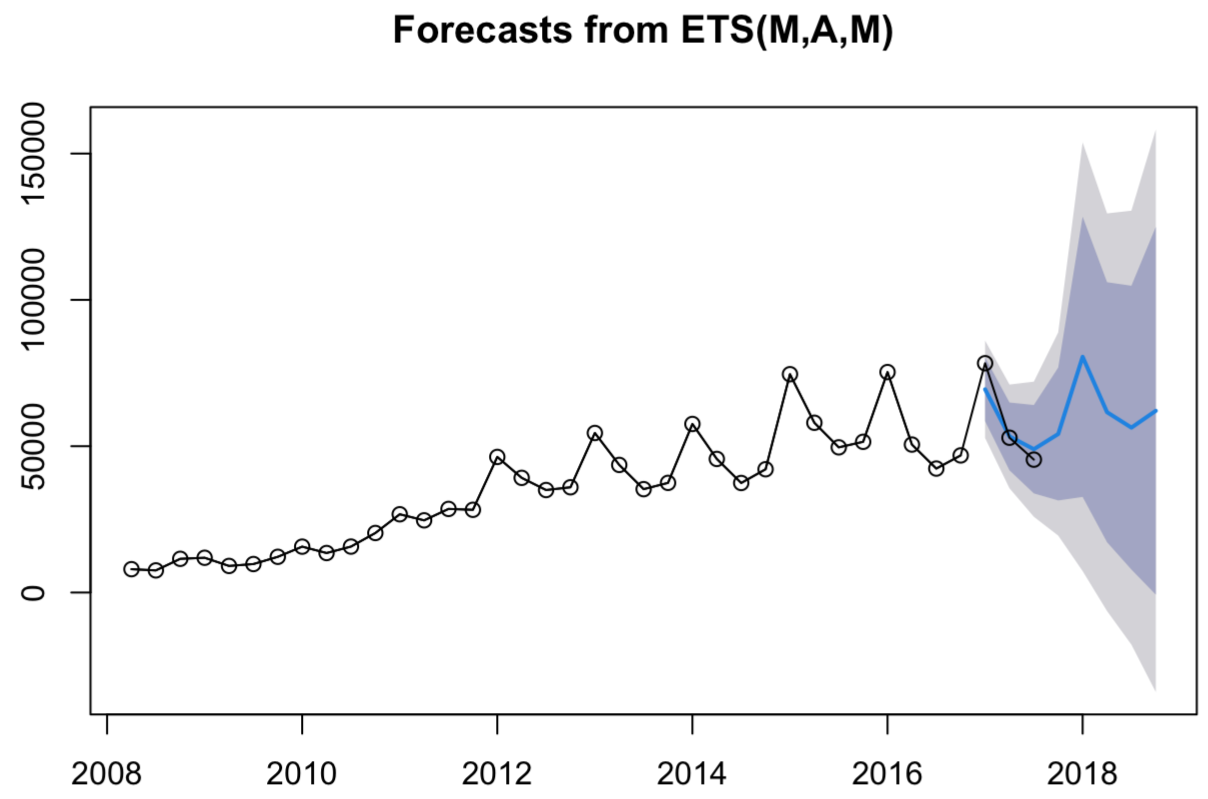
```
## ETS(M,A,M)
##
## Call:
##  ets(y = oApple)
##
##   Smoothing parameters:
##     alpha = 0.493
##     beta  = 0.493
##     gamma = 0.507
##
##   Initial states:
##     l = 7125.3462
##     b = 1485.7975
##     s = 1.1511 1.1163 0.8322 0.9004
##
##   sigma:  0.1222
##
##      AIC     AICc      BIC
## 703.9538 711.1538 717.9519
```

```
fApple.ets <- forecast(etsfit)
summary(fApple.ets)
```

```
##
## Forecast method: ETS(M,A,M)
##
## Model Information:
## ETS(M,A,M)
##
## Call:
##  ets(y = oApple)
##
##   Smoothing parameters:
##     alpha = 0.493
##     beta  = 0.493
##     gamma = 0.507
##
##   Initial states:
##     l = 7125.3462
##     b = 1485.7975
##     s = 1.1511 1.1163 0.8322 0.9004
##
##   sigma:  0.1222
##
##      AIC     AICc      BIC
## 703.9538 711.1538 717.9519
##
## Error measures:
##                   ME     RMSE      MAE       MPE     MAPE      MASE      ACF1
## Training set -41.934 4120.155 2883.262 -0.297759 8.677434 0.4160202 0.1438481
##
## Forecasts:
##         Point Forecast     Lo 80     Hi 80      Lo 95     Hi 95
## 2017 Q1       69439.83 58568.387  80311.27  52813.394  86066.26
## 2017 Q2       53347.98 41773.016  64922.95  35645.598  71050.37
## 2017 Q3       48972.04 33884.613  64059.47  25897.811  72046.27
## 2017 Q4       54176.09 31475.035  76877.14  19457.824  88894.35
## 2018 Q1       80540.07 32680.293 128399.85   7344.857 153735.28
## 2018 Q2       61619.03 17211.212 106026.85  -6296.866 129534.93
## 2018 Q3       56344.41  7869.773 104819.05 -17791.151 130479.97
## 2018 Q4       62103.80  -718.363 124925.97 -33974.410 158182.02
```

Que graficado por trimestres, tiene la siguiente forma:

```
plot(fApple.ets)
lines(window(zApple),type="o")
```



Diferencias entre la predicción y los valores reales

Se aprecia cómo predice con un ligero error, sobretodo en el primer trimestre.

```
matrix(c(fApple.ets$mean[1:cOmit],zApple[(nObs-cOmit+1):nObs]),ncol=2)
```

```
##          [,1]  [,2]
## [1,] 69439.83 78351
## [2,] 53347.98 52896
## [3,] 48972.04 45408
```

```
etsfit<-ets(window(tApple,end=2016+3/4))
fventas.ets=forecast(etsfit,h=cOmit)
forecast:::testaccuracy(fApple.ets$mean,window(tApple,start=2017),test = NULL, d = NULL, D = NU
LL)
```

```
##            ME          RMSE           MAE           MPE          MAPE
## 1631.71485476 5547.24116365 4309.06585239    0.88999690    6.69226842
##          ACF1     Theil's U
##   -0.05148699    0.19082258
```

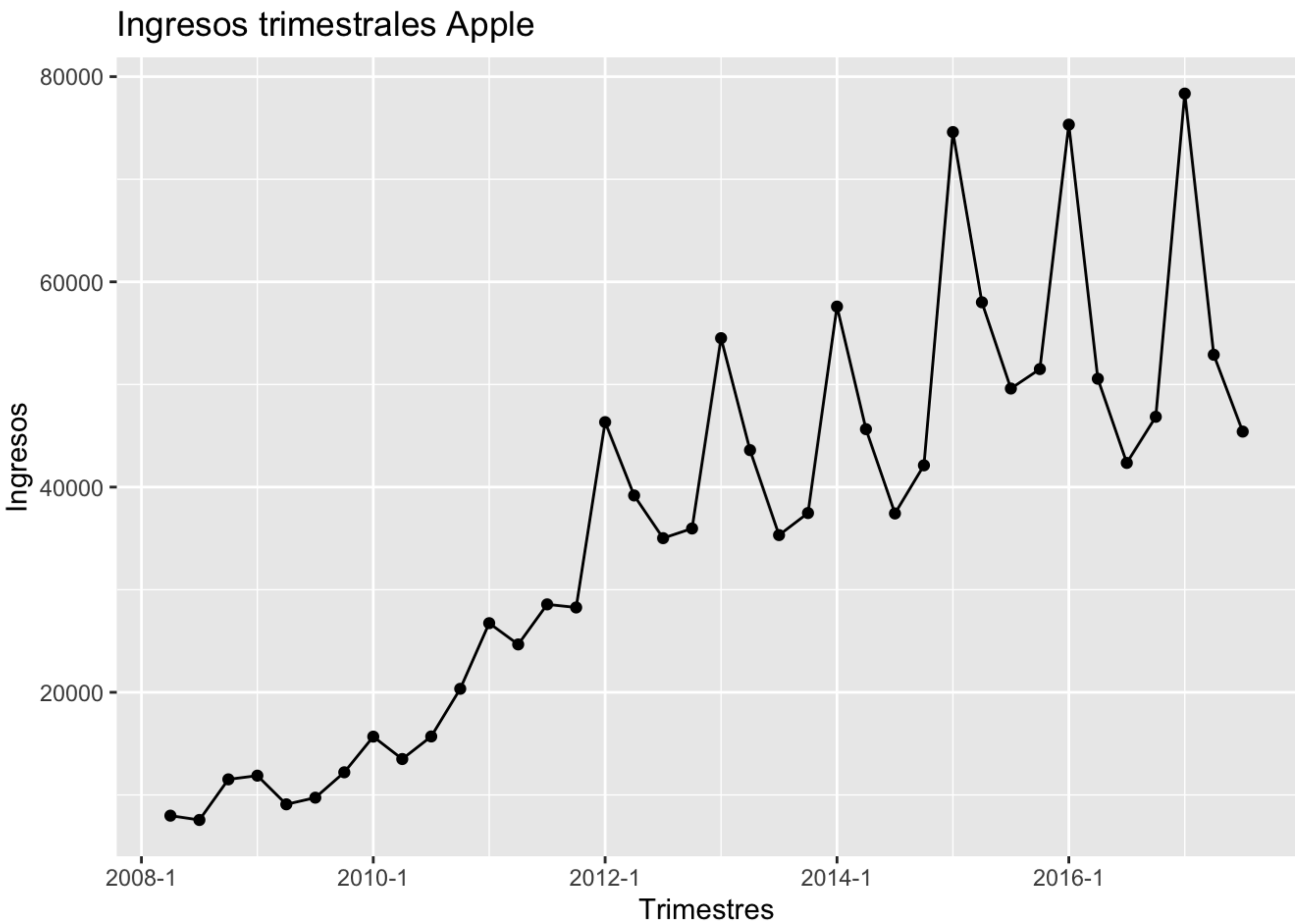
Tranformación serie temporal para que sea estacionaria

Bajo el supuesto que una serie no es estacionaria en media y varianza, debemos realizar en primer lugar la transformación de la serie para conseguir que sea *estacionaria en varianza* y luego la transformación para obtener la *estacionariedad en media*.

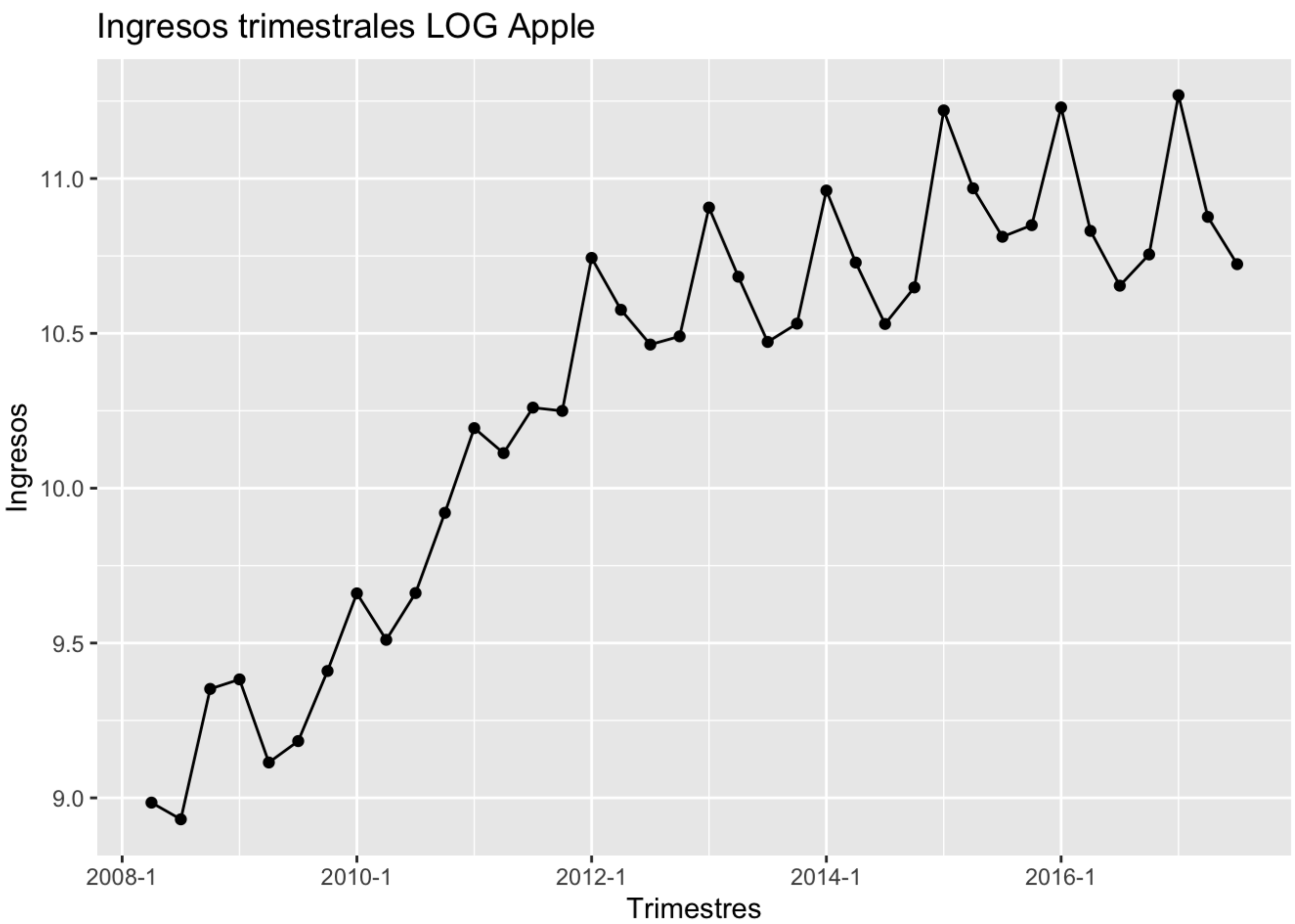
Estacionariedad en varianza

Para ello se realizará una transformación logarítmica. Así se puede apreciar a continuación el primer gráfico, con la variable sin transformar, y el segundo, donde la serie es estacionaria en varianza después de la transformación logarítmica.

```
df_new <- data.frame(value = as.vector(zApple),
                     time = time(zApple))
ggplot(df_new)+geom_point(aes(x=time,y=value))+geom_line(aes(x=time,y=value))+ylab("Ingresos")+
ggtitle("Ingresos trimestrales Apple")+xlab("Trimestres")
```




```
zlApple=log(zApple)
df_newl <- data.frame(value = as.vector(zlApple),
                     time = time(zlApple))
ggplot(df_newl)+geom_point(aes(x=time,y=value))+geom_line(aes(x=time,y=value))+ylab("Ingresos")
+ggtitle("Ingresos trimestrales LOG Apple")+xlab("Trimestres")
```



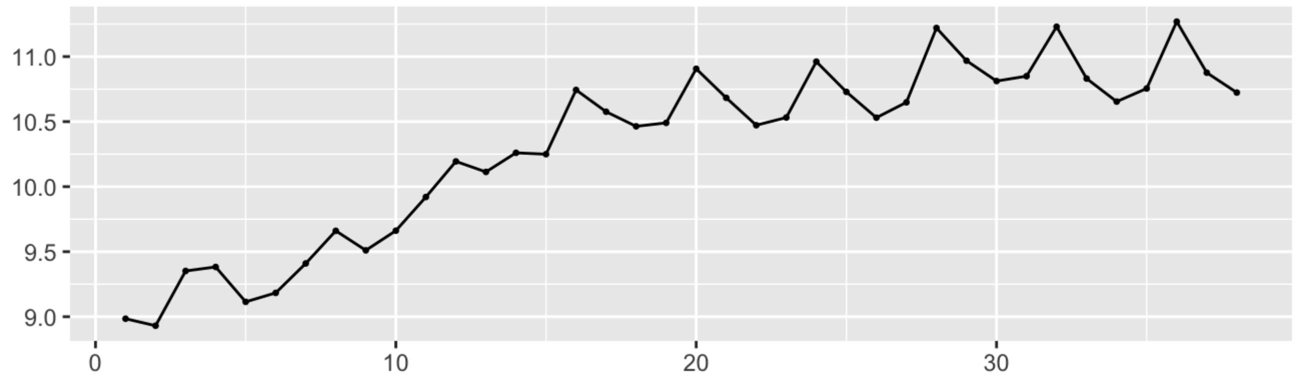
Estacionariedad en media

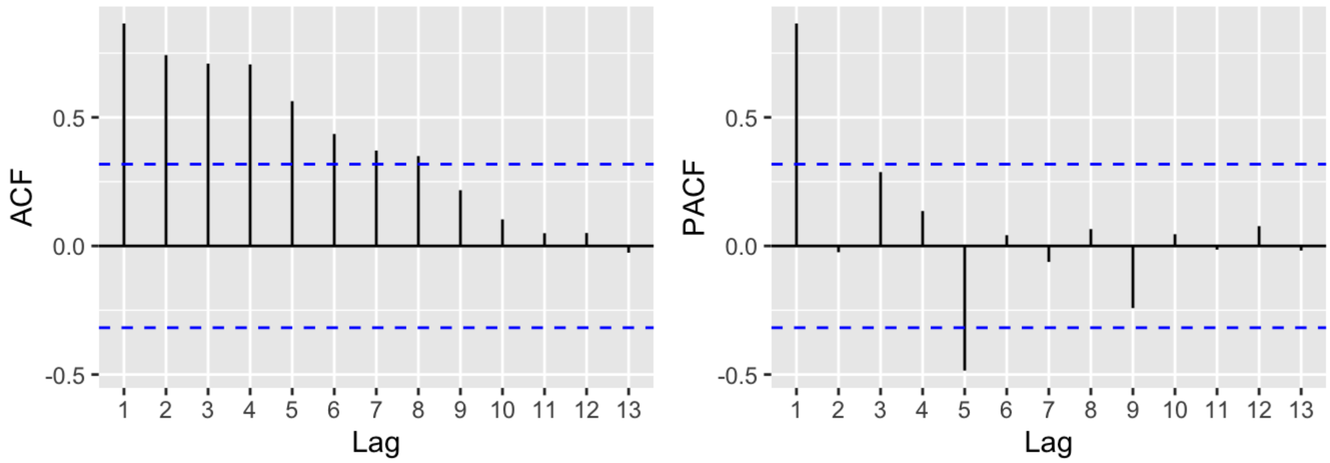
La transformación consiste en la aplicación del operador diferencia (realizar diferencias) hasta obtener una serie ya estacionaria. Además, la función de autocorrelación simple (**ACF**) se representa gráficamente dando valor a la autocorrelación para cada retardo y graficando las dos intervalos de confianza que permiten determinar si el coeficiente de autocorrelación es cero o no. Esto es importante porque cuando una serie temporal es estacionaria, la información histórica relevante es la mas cercana, así cuando se calcula la autocorrelación de retardos alejados en el tiempo se debe esperar obtener un valor pequeño o cero.

Adicionalmente de calcula la función de autocorrelación parcial (**PACF**), mediante una regresión donde cada valor de la función es el parámetro estimado del último retardo incluido en la regresión. La diferencia entre la *acf* y la *pacf*, es que en la *pacf* elimina los efectos indirectos.

Se define primero el **operador retardo**, que permite retardar una serie temporal:

```
ggtsdisplay(zlApple)
```

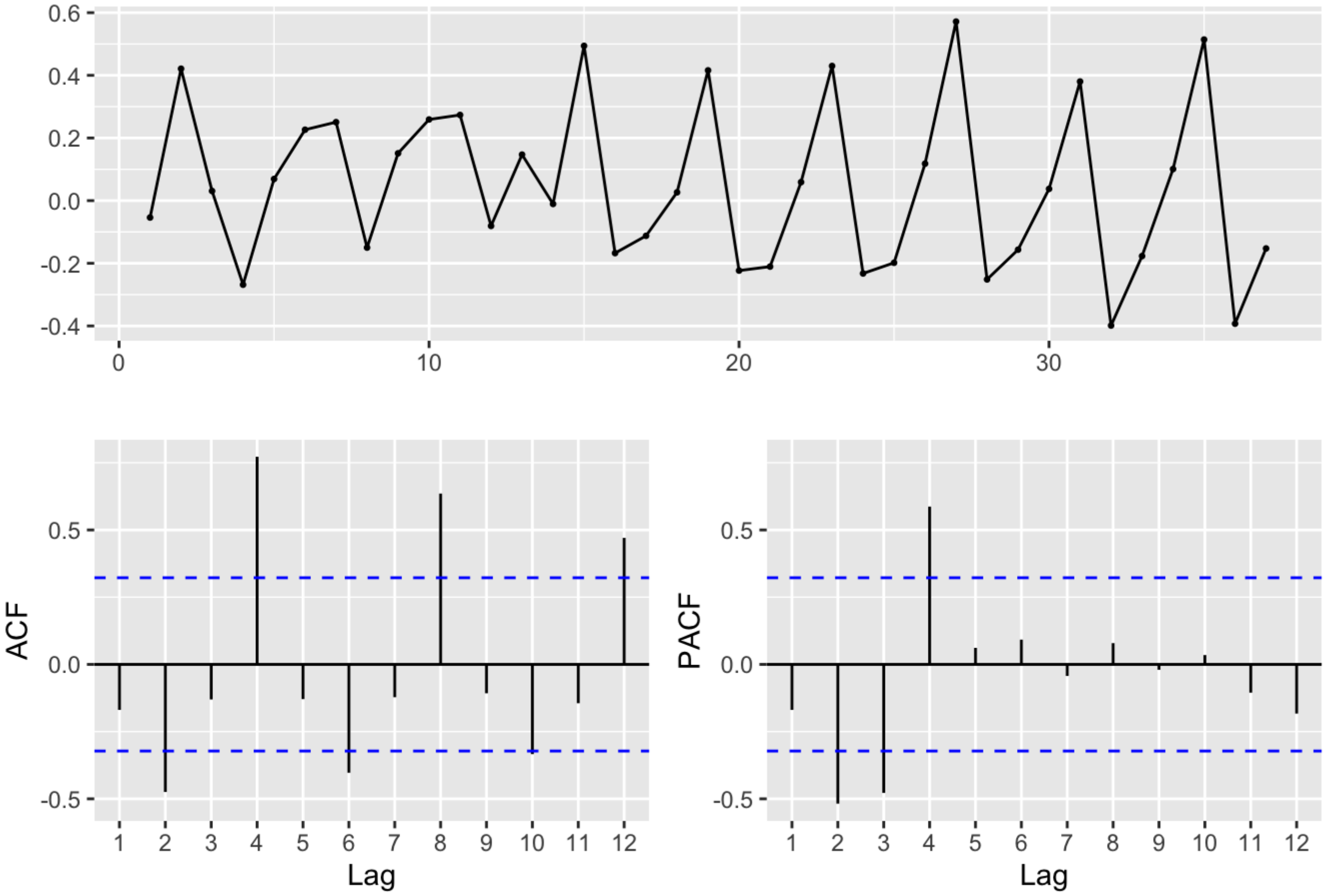




En este caso, el **ACF** de los retardos alejados se sale de las bandas de los intervalos de confianza, por lo que la autocorrelación no es cero, lo que significa que la serie temporal **no es estacionaria.**

La **diferencia de primer orden** consiste en restar a la serie original la misma serie pero retardar un periodo:

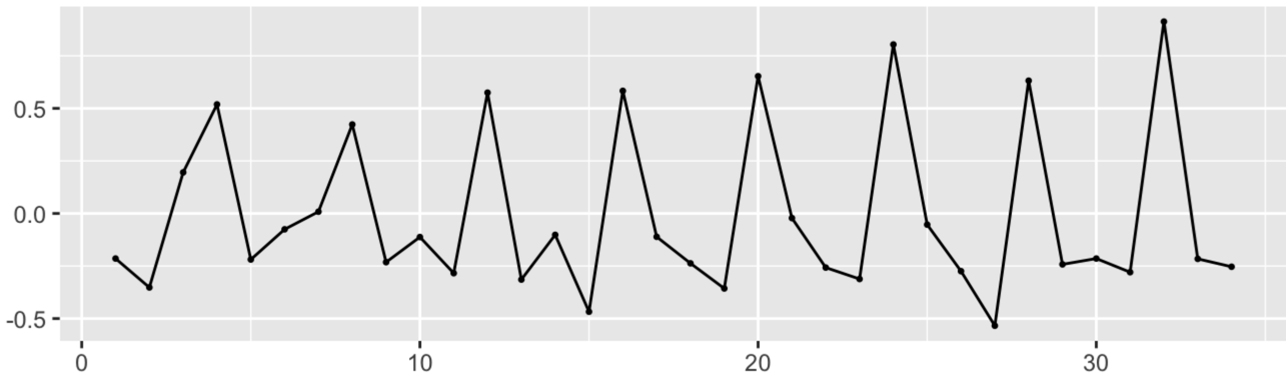
```
ggtsdisplay(diff(zlApple))
```

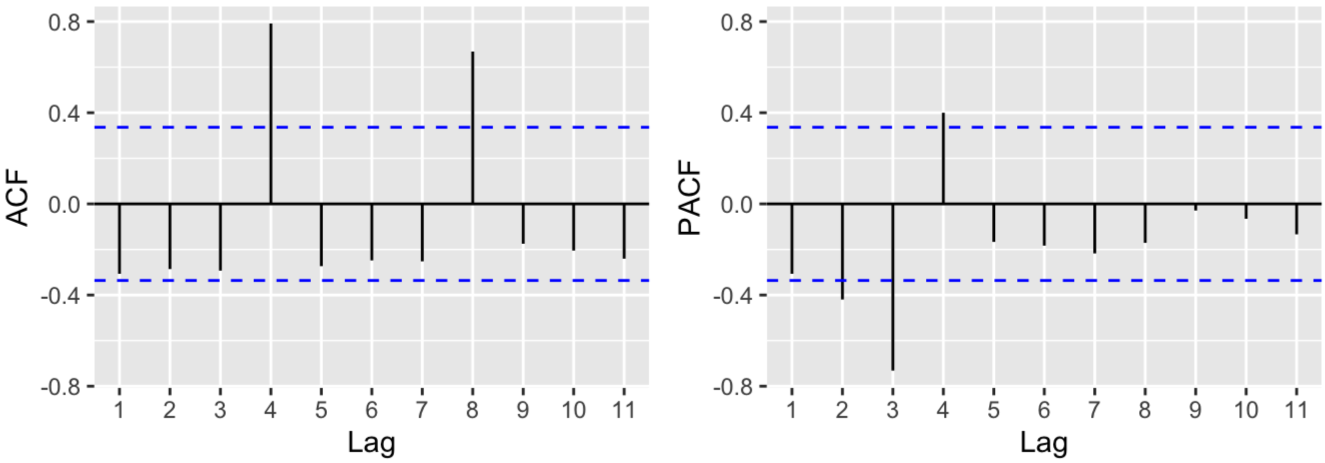


En este caso la autocorrelación de orden elevado (12) tiene un valor significativo distinto de cero, por lo que la serie temporal **sigue sin ser estacionaria.**

La **diferencia de segundo orden** consiste en aplicar dos veces la diferencia a la serie original, o una diferencia a la serie ya diferenciada:

```
ggtsdisplay(diff(diff(zlApple,3),1))
```





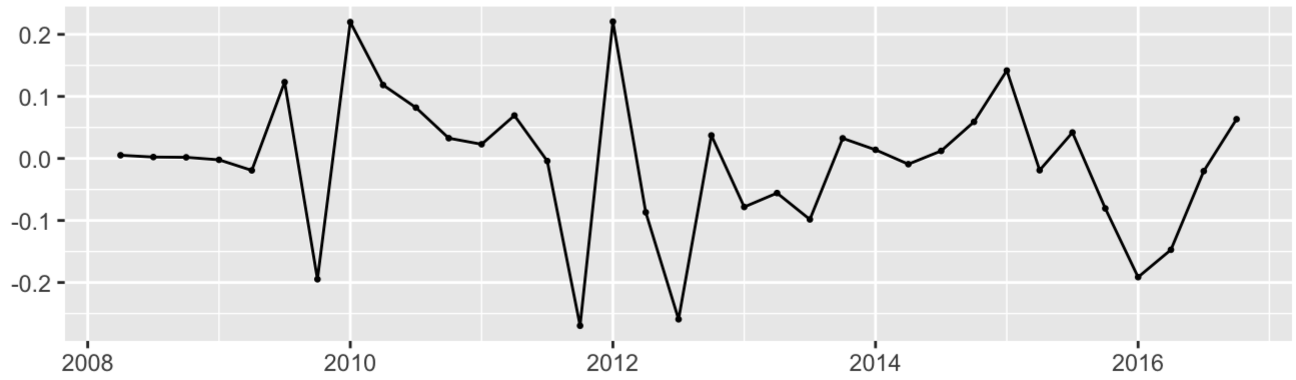
Modelo ARIMA

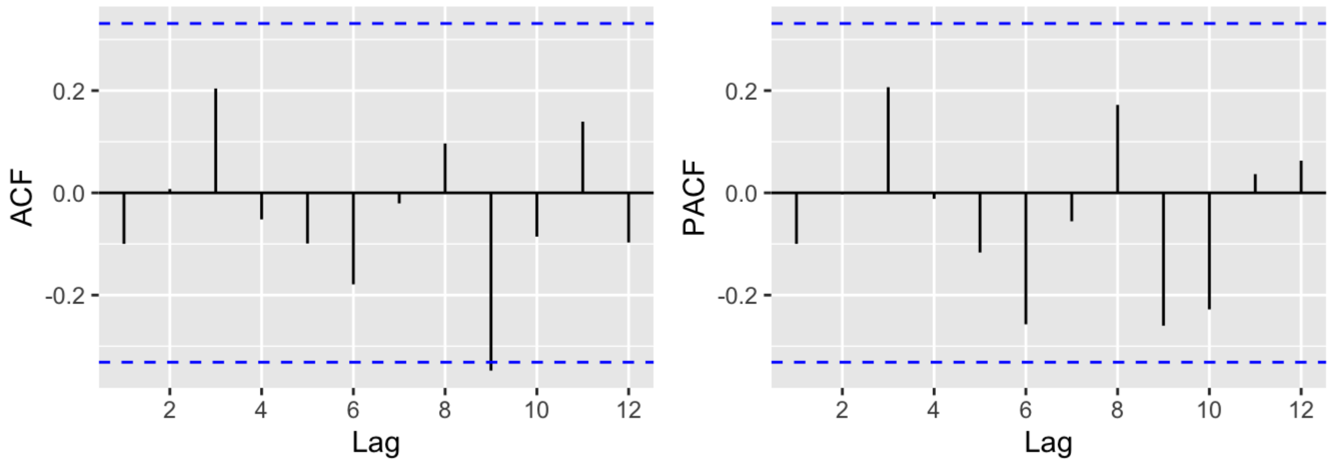
Cuando la serie temporal es estacional, se tiene que modelizar el componente estacional con un modelo ARIMA adicional.

```
#ARIMA MODEL
fit1=auto.arima(oApple,lambda=0)
summary(fit1)
```

```
## Series: oApple
## ARIMA(0,1,0)(0,1,0)[4]
## Box Cox transformation: lambda= 0
##
## sigma^2 estimated as 0.01472:  log likelihood=20.72
## AIC=-39.45   AICc=-39.3   BIC=-38.04
##
## Training set error measures:
##                     ME     RMSE      MAE       MPE     MAPE      MASE      ACF1
## Training set -764.5058 4786.405 3054.054 -1.321616 8.284962 0.4406634 0.1269135
```

```
#residual analysis
ggtsdisplay(fit1$residuals)
```





Test Box-Ljung

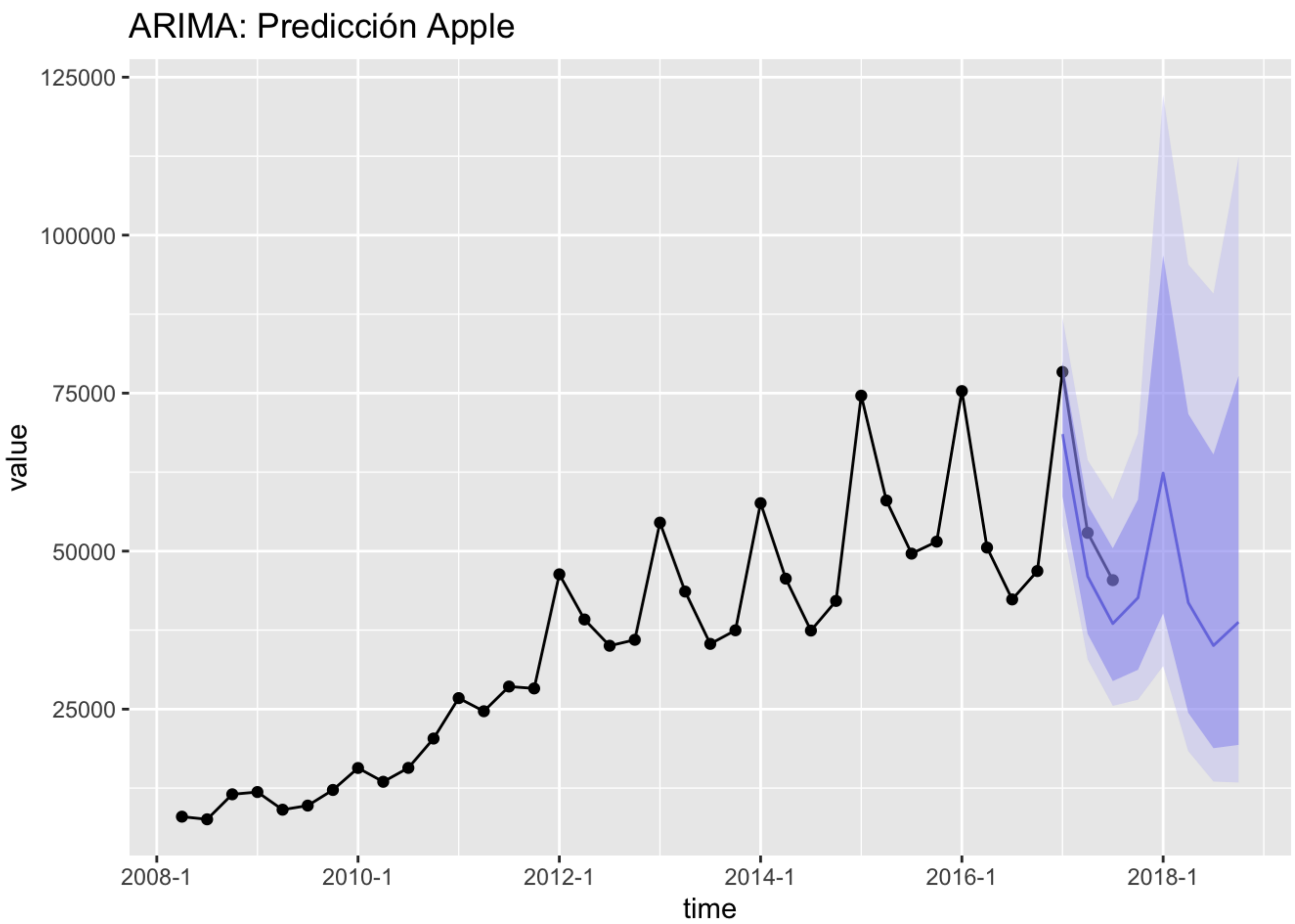
El test Box-Ljung consiste en contrastar si los retardos de la acf son cero a la vez. Si el valor-p es inferior a los valores del nivel de significación rechazamos que sea ruido blanco y el modelo no sería correcto.

```
#box-Ljung Test
Box.test(fit1$residuals,lag=4, fitdf=3, type="Lj")
```

```
##
##  Box-Ljung test
##
## data:  fit1$residuals
## X-squared = 2.1794, df = 1, p-value = 0.1399
```

```
fApple.arima=forecast(fit1)

ggplot(df_new)+geom_point(aes(x=time,y=value))+geom_line(aes(x=time,y=value))+ geom_forecast(fA
pple.arima,alpha=0.4)+ggtitle("ARIMA: Predicción Apple")
```

```
## Warning in geom_forecast(fApple.arima, alpha = 0.4): Use autolayer instead of
## geom_forecast to add a forecast layer to your ggplot object.
```




```
fApple.arima
```

```
##         Point Forecast    Lo 80    Hi 80    Lo 95     Hi 95
## 2017 Q1       68524.50 58656.57 80052.53 54021.77  86920.63
## 2017 Q2       45993.21 36914.17 57305.26 32857.77  64379.78
## 2017 Q3       38534.34 29436.35 50444.28 25525.07  58174.00
## 2017 Q4       42622.67 31230.74 58169.99 26490.28  68579.55
## 2018 Q1       62338.78 40156.60 96774.23 31816.09 122143.34
## 2018 Q2       41841.40 24416.21 71702.49 18358.80  95360.44
## 2018 Q3       35055.84 18821.04 65294.59 13541.05  90754.54
## 2018 Q4       38775.11 19344.31 77723.59 13387.03 112310.89
```