Laboratorio No.2

Paul Belches José Pablo Cifuentes Oscar Juárez

3 de agosto, 2020

Obtener datos desde el PDF

```
pages<-extract_tables("C01-Importación-de-combustibles-VOLUMEN-2020-03.pdf") #PDF con datos actualizado
datosImp <- do.call(rbind, pages)
nombresVar<-datosImp[1,]
datosImp<-as.data.frame(datosImp[2:nrow(datosImp),])
nombresVar[c(1,4,5,6,8,10,11,15,16,21,23,24)]<-c("Anio","GasAviacion","GasSuperior","GasRegular","rTurbo
names(datosImp)<-nombresVar
```

Leemos los datos de un CSV

```
#Leer
dataSet = read.csv("datos.csv",stringsAsFactors = FALSE, na.strings = TRUE, strip.white = TRUE,sep = ",
```

Limpiamos la data

```
#Seleccionar
dataset = dataSet[c(1,2,5,6, 9,10)]
dataset = dataset[-c(46, 96, 146, 196),]
```

```
#Limpiar
dataset$Diesel[dataset$Diesel =="-"] <- 0
dataset$DieselLS[dataset$DieselLS =="-"] <- 0
dataset2 = dataset
```

Implicamos el dataset limpio

```
#COnvertir
options(digits=9)
dataset2$Anio = as.numeric(dataset2$Anio)
dataset2$Mes = as.numeric(dataset2$Mes)
dataset2$GasSuperior = as.numeric(gsub(",", "", dataset2$GasSuperior))
dataset2$GasRegular = as.numeric(gsub(",", "", dataset2$GasRegular))
dataset2$Diesel = as.numeric(gsub(",", "", dataset2$Diesel))
```

```r
dataset2$DieselLS = as.numeric(gsub(",", "", dataset2$DieselLS))
#Unir Diesel
dataset2$Diesel = dataset2$Diesel + dataset2$DieselLS
dataset2 = dataset2[-6]
dataSet = dataset2
```

Clasifición de las variables

##ANIO

Año del índice

Variable categórica ordinal

##Mes

Mes del índice

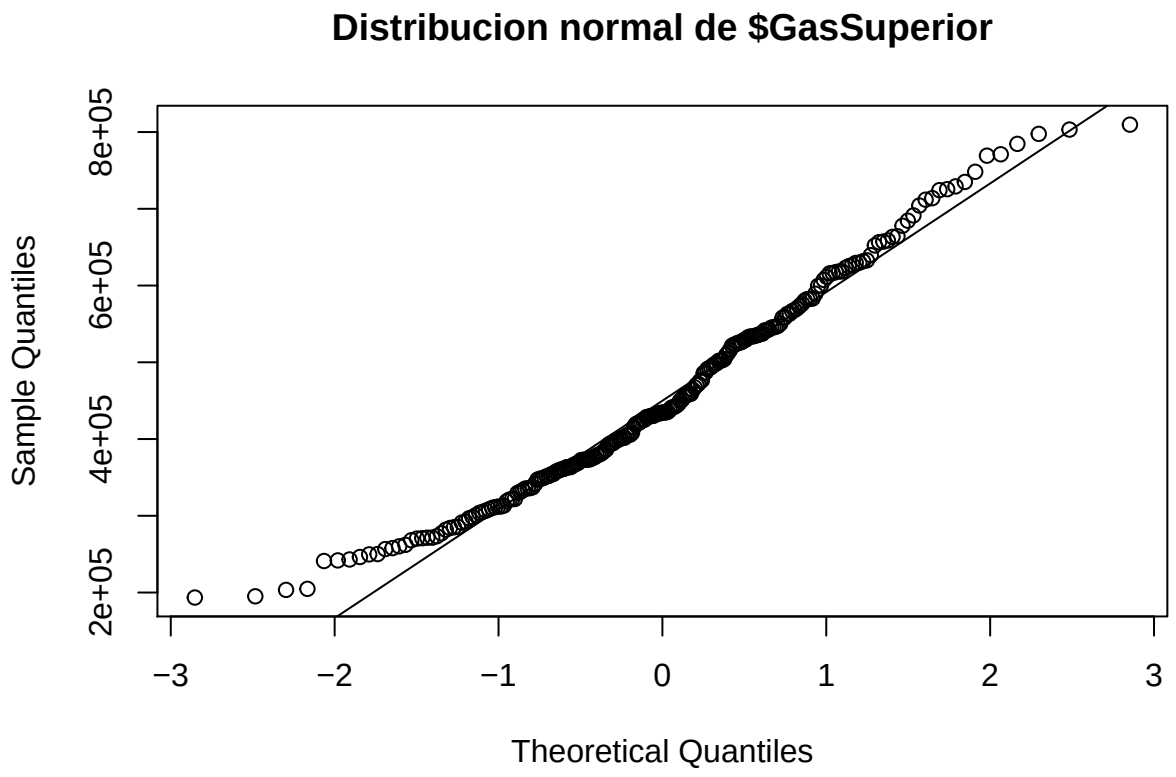
Variable categórica ordinal

##GasSuperior

Volumen de importación gasolina superior

Variable cuantitativa continua

```r
qqnorm(dataSet$GasSuperior, main = "Distribucion normal de $GasSuperior")
qqline(dataSet$GasSuperior)
```



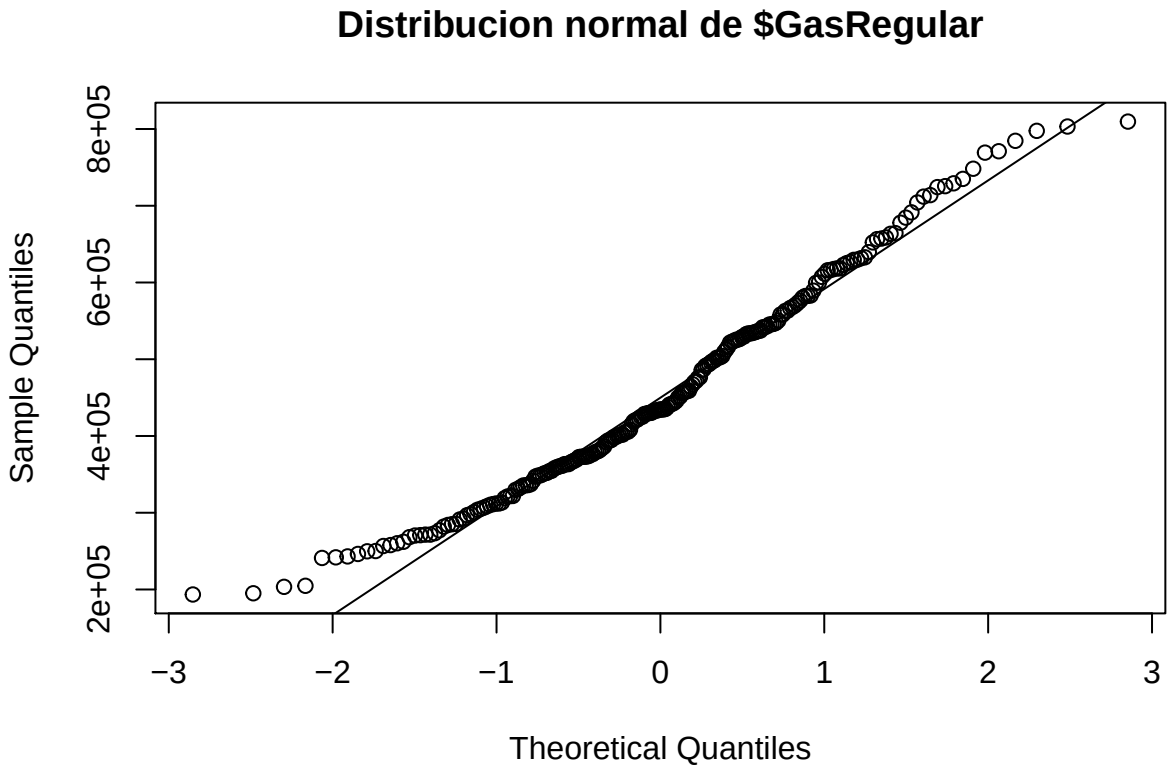
Por la gráfica anterior podemos afirmar que la varaible cuenta con una distribución normal.

##GasRegular

Volumen de importación gasolina regular

Variable cuantitativa continua

```
qqnorm(dataSet$GasSuperior, main = "Distribucion normal de $GasRegular")
qqline(dataSet$GasSuperior)
```



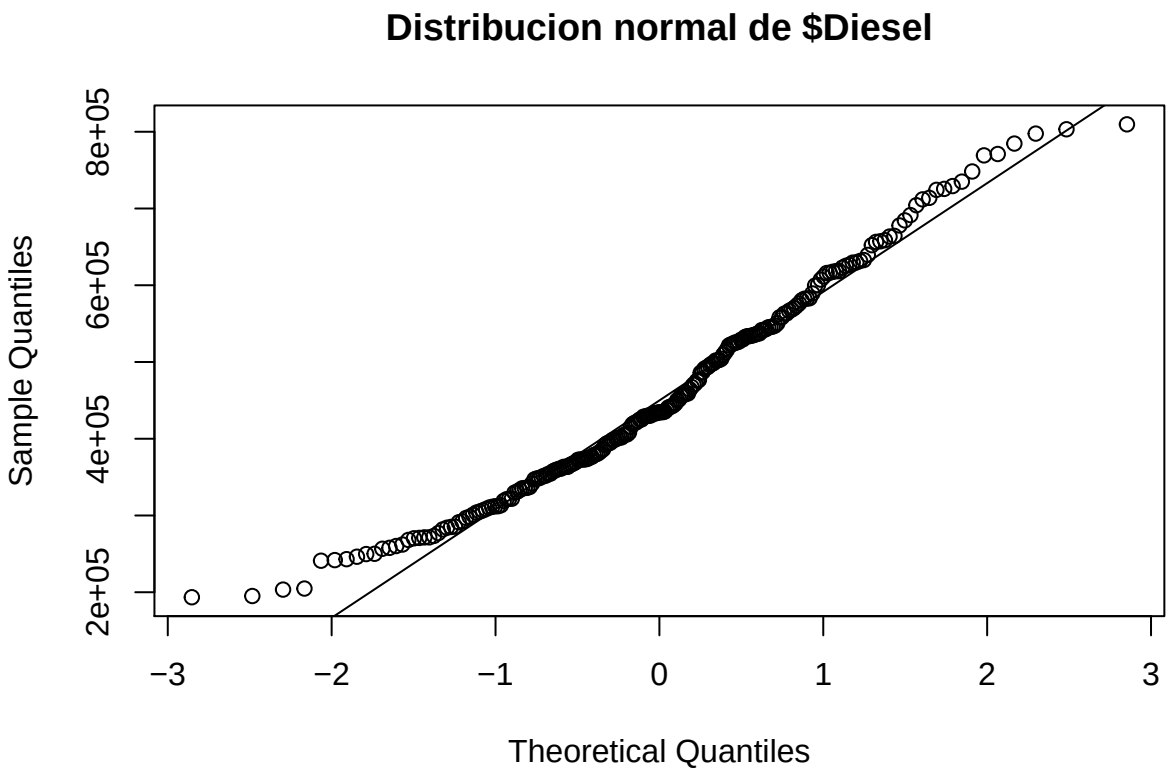
Por la gráfica anterior podemos afirmar que la varaible cuenta con una distribución normal.

##Diesel

Volumen de importación diesel

Variable cuantitativa continua

```
qqnorm(dataSet$GasSuperior, main = "Distribucion normal de $Diesel")
qqline(dataSet$GasSuperior)
```



Por la gráfica anterior podemos afirmar que la varaible cuenta con una distribución normal.

Preguntas

En que año fue mayor la importación de diesel promedio?

```
p <- dataSet %>%
group_by(Anio) %>%
summarise(mean = mean(Diesel))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
p[order(p$mean),]
```

```
## # A tibble: 20 x 2
##     Anio     mean
##    <dbl>    <dbl>
##  1  2001  463923.
##  2  2002  551371.
##  3  2008  658184.
##  4  2004  659549.
##  5  2003  680349.
##  6  2006  744338.
##  7  2010  750928.
```

```
##  8  2005  757132.
##  9  2011  768319.
## 10  2007  792744.
## 11  2012  795824.
## 12  2013  813408.
## 13  2009  844722.
## 14  2014  915396.
## 15  2018  973979.
## 16  2017 1015425.
## 17  2015 1033109.
## 18  2016 1054201.
## 19  2019 1127882.
## 20  2020 1201178.
```

Como podemos observar existe una tendencia de crecimiento en la importacion de diesel. Es interesante ver el decremento que existió en el año 2018 y 2017, gracias a que rompe esta tendencia. De igual manera 2009, es un pico en la importacion que se comporta de manera atípica.

En que año fue mayor la importación de gasolina regular promedio?

```
p <- dataSet %>%
group_by(Anio) %>%
summarise(mean = mean(GasRegular))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
p[order(p$mean),]
```

```
## # A tibble: 20 x 2
##     Anio    mean
##    <dbl>   <dbl>
##  1  2001 156459.
##  2  2002 165775.
##  3  2005 184851.
##  4  2003 189637.
##  5  2004 198150.
##  6  2006 218163.
##  7  2008 225372.
##  8  2007 250146.
##  9  2012 253883.
## 10  2011 272364.
## 11  2010 279867.
## 12  2013 298757.
## 13  2009 301377.
## 14  2014 366927.
## 15  2015 458332.
## 16  2016 495445.
## 17  2017 518473.
## 18  2018 554815.
## 19  2019 684082.
## 20  2020 702138.
```

Como podemos observar existe una tendencia de crecimiento en la importacion de gasolina. Es interesante ver que el 2009, es un pico en la importacion que se comporta de manera atípica.

En que año fue mayor la importación de gasolina superior promedio?

```
p <- dataSet %>%
group_by(Anio) %>%
summarise(mean = mean(GasSuperior))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
p[order(p$mean),]
```

```
## # A tibble: 20 x 2
##     Anio    mean
##    <dbl>   <dbl>
##  1  2001 288240.
##  2  2002 329102.
##  3  2003 370148.
##  4  2004 379443.
##  5  2008 387417.
##  6  2011 394470.
##  7  2005 400707.
##  8  2006 402797.
##  9  2012 406422.
## 10  2013 417940.
## 11  2010 434392.
## 12  2009 445274.
## 13  2007 448246.
## 14  2014 477877.
## 15  2015 576576.
## 16  2018 593199.
## 17  2016 603855.
## 18  2017 615118.
## 19  2019 636148.
## 20  2020 664725.
```

Como podemos observar existe una tendencia de crecimiento en la importacion de gasolina. Es interesante ver el decremento que existio en el ano 2018, gracias a que rompe esta tendencia. De igual manera 2007, es un pico en la importacion que se comporta de manera atípica.

En que mes es mayor la importación de diesel promedio?

```
p <- dataSet %>%
group_by(Mes) %>%
summarise(mean = mean(Diesel))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
p[order(p$mean),]
```

```
## # A tibble: 12 x 2
##      Mes    mean
##    <dbl>   <dbl>
##  1     9 672241.
##  2     6 718116.
##  3     8 749127.
##  4    11 760036.
##  5    10 771444.
##  6     2 796646.
##  7     7 807566.
##  8     4 837208.
##  9     5 877185.
## 10     1 897479.
## 11    12 940674.
## 12     3 949604.
```

Como podemos observar marzo es el mes en donde es mayor el consumo de diesel, seguido por diciembre. Y septiembre es el mes en donde menos se consume.

En que mes es mayor la importación de gasolina regular promedio?

```r
p <- dataSet %>%
group_by(Mes) %>%
summarise(mean = mean(GasRegular))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
p[order(p$mean),]
```

```
## # A tibble: 12 x 2
##      Mes    mean
##    <dbl>   <dbl>
##  1     9 292627.
##  2     6 296547.
##  3     7 298160.
##  4     8 313220.
##  5    11 315190.
##  6     5 324801.
##  7    12 332949.
##  8     4 333128.
##  9     2 335937.
## 10    10 342633.
## 11     1 344456.
## 12     3 361859.
```

Como podemos observar marzo es el mes en donde es mayor el consumo de gasolina, seguido por enero. Y septiembre es el mes en donde menos se consume. Es interesante ver como diciembre se encuentra en la mitad de la tabla, en específico para la gasolina regular.

En que mes es mayor la importación de gasolina superior promedio?

```
p <- dataSet %>%
group_by(Mes) %>%
summarise(mean = mean(GasSuperior))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
p[order(p$mean),]
```

```
## # A tibble: 12 x 2
##      Mes    mean
##    <dbl>   <dbl>
##  1     9 415425.
##  2     8 431703.
##  3     6 439920.
##  4     4 444913.
##  5     2 446619.
##  6    11 451439.
##  7     7 454411.
##  8     5 454746.
##  9    10 459691.
## 10     1 464961.
## 11    12 497202.
## 12     3 505573.
```

Como podemos observar marzo es el mes en donde es mayor el consumo de gasolina, seguido por diciembre. Y septiembre es el mes en donde menos se consume. Es interesante ver los picos en al principio y fin de año, gracias a que en los cuartos de mitad de año se encuentra los puntos más bajos.

Serie de tiempo Gasolina Regular

Primero creamos la serie de tiempo para los datos del Diesel

```
#View(dataSet)

regular <- ts(dataSet$GasRegular, start=c(2001, 1), end=c(2020,3), frequency=12)

start(regular)
```

```
## [1] 2001    1
```

```
end(regular)
```

```
## [1] 2020    3
```

Construcción del modelo ARIMA GAS REGULAR

Identificación

A continuación se exploran las características de la serie:

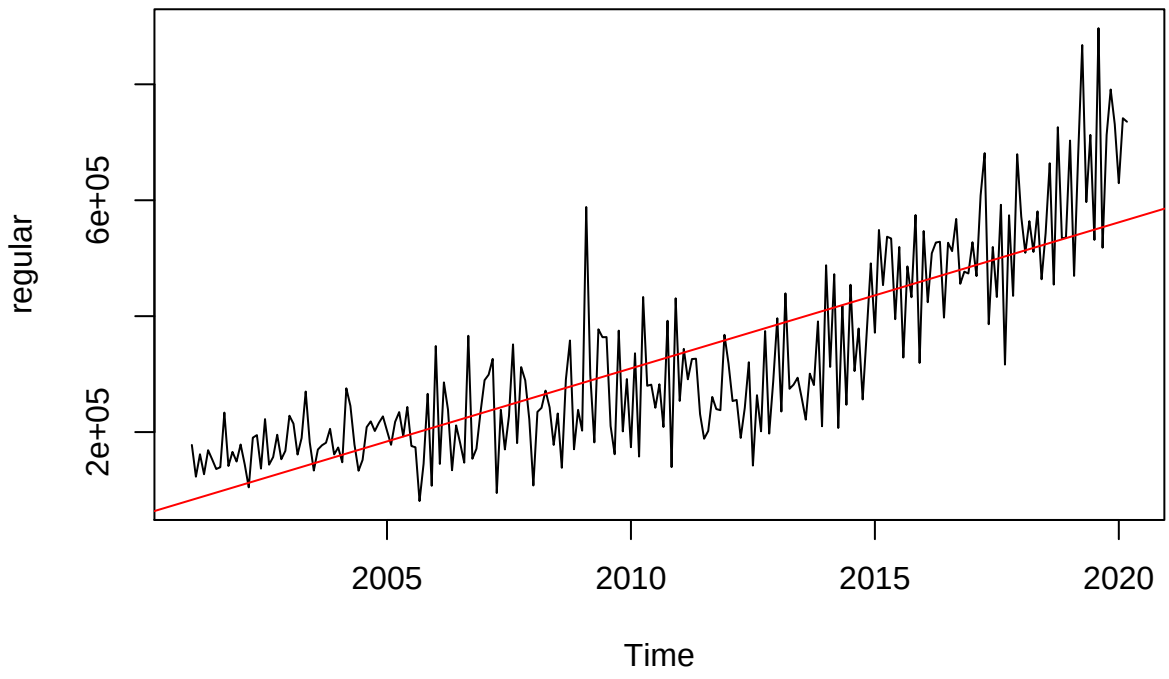
Frecuencia de la serie:

```
frequency(regular)
```

```
## [1] 12
```

Gráfica de la serie de tiempo

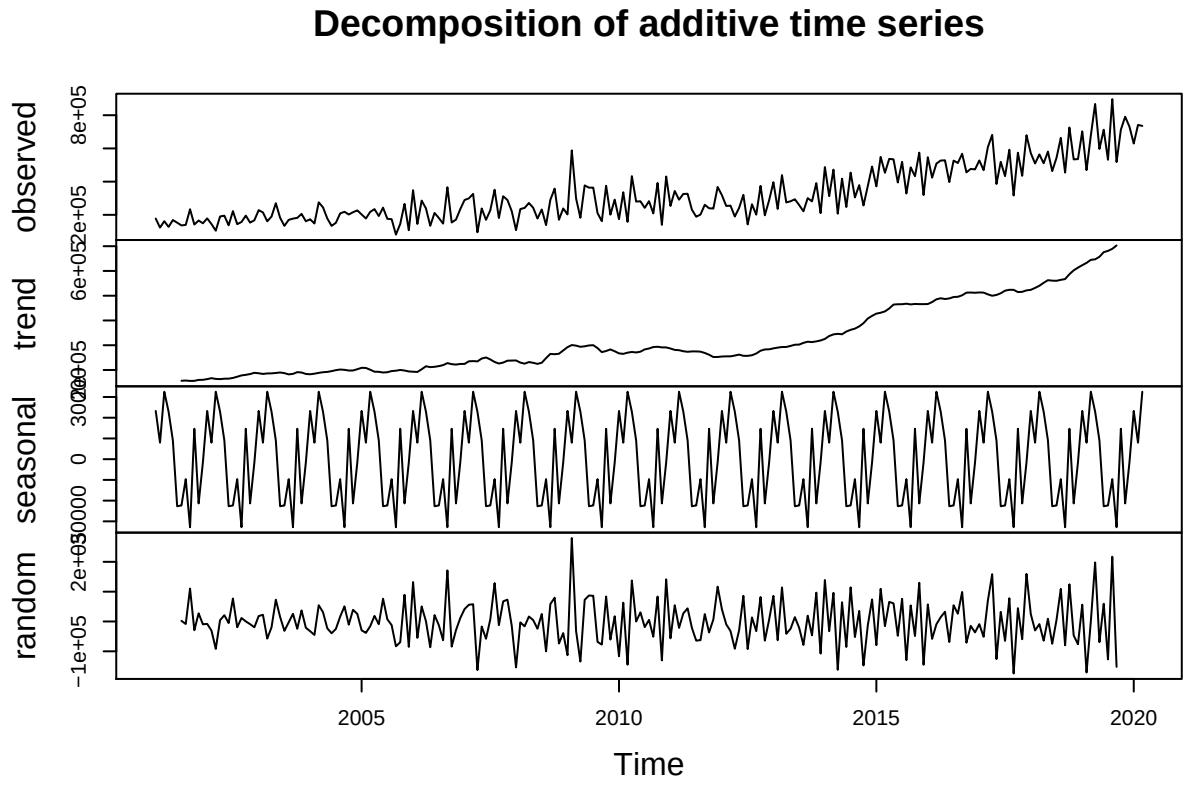
```
plot(regular)
abline(reg=lm(regular~time(regular)), col=c("red"))
```



Podemos observar que es una serie de tiempo continua con tendencia a crecer por lo que es una serie de tiempo no estacionaria.

Descomposición de la serie

```
plot(decompose(regular))
```



Podemos observar una serie con tendencia a aumentar, que no es estacionaria en varianza y también tiene estacionalidad.

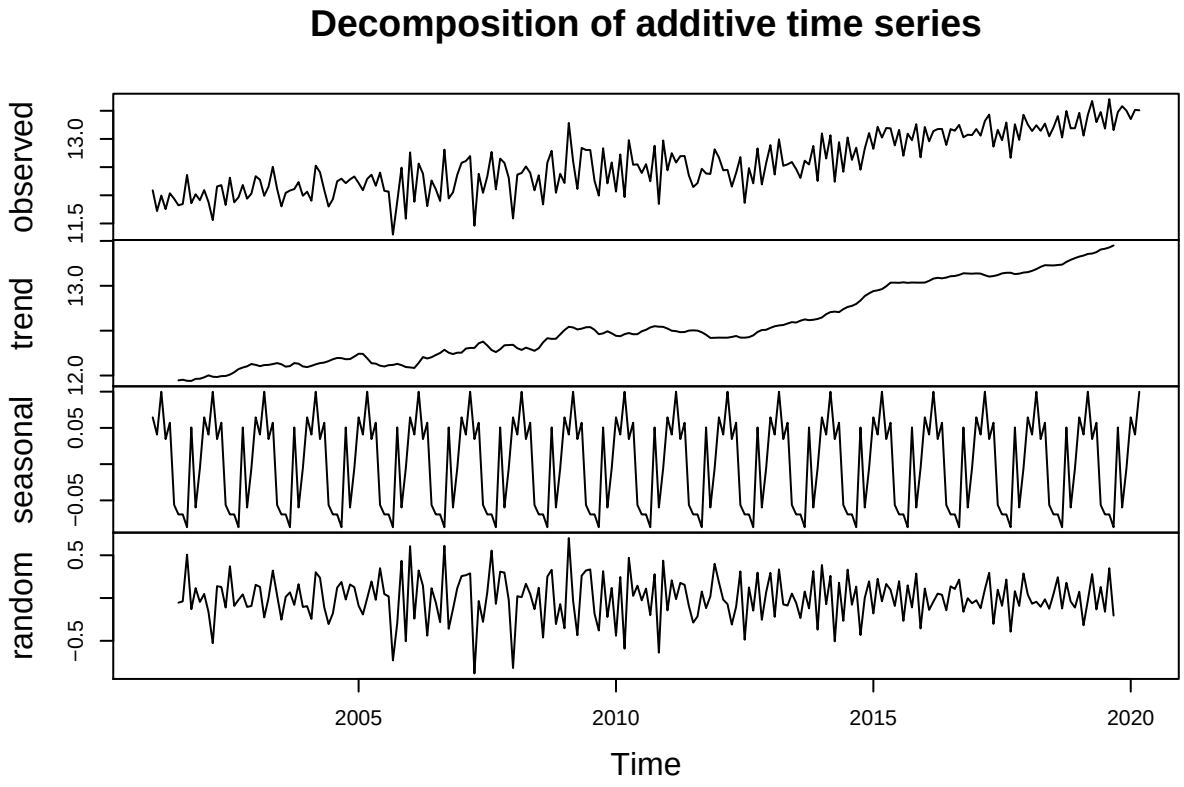
A continuación se separa la serie de tiempo en entrenamiento y prueba

```
train <- head(regular, round(length(regular) * 0.7))
h <- length(regular) - length(train)
test <- tail(regular, h)
```

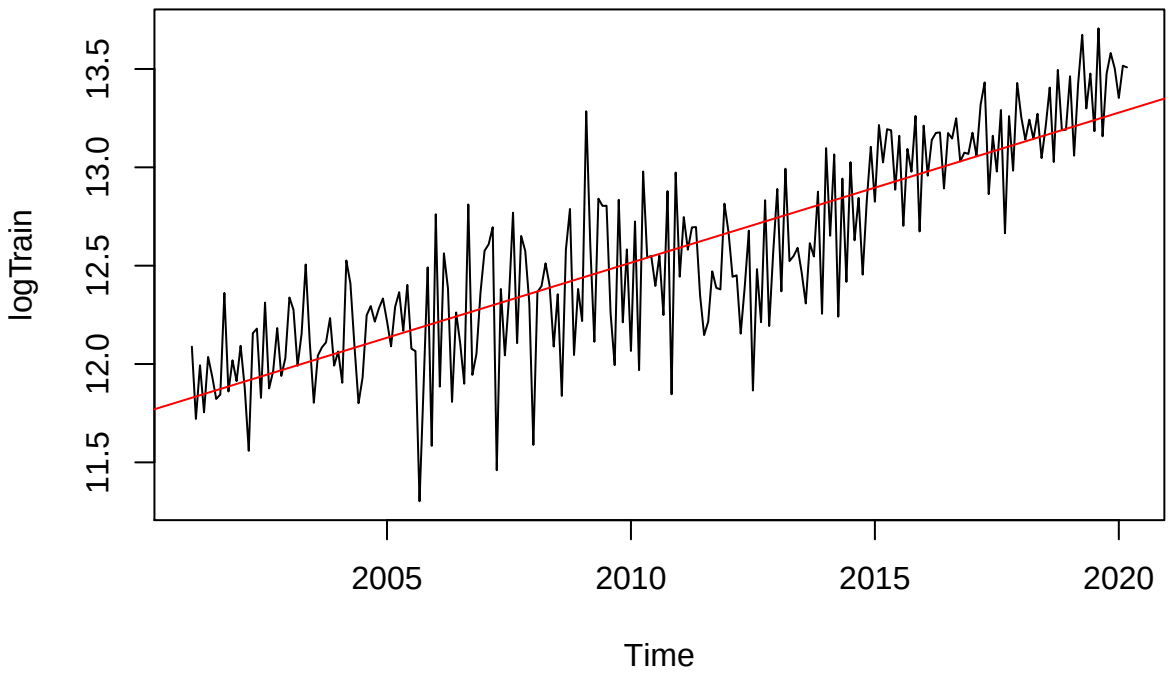
Estimar los parámetros del modelo

A continuación aplicaremos una transformación logaritmica para hacer que la serie sea constante en varianza.

```
logTrain <- log(regular)
plot(decompose(logTrain))
```




```
plot(logTrain)
abline(reg=lm(logTrain~time(logTrain)), col=c("red"))
```



Podemos notar que se logro hacer un poco más constante la varianza de la serie. A continuación verificaremos que es estacionaria en media. Si tiene raices unitarias podemos decir que no es estacionaria en media.

```
adfTest(train)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -1.0843
##   P VALUE:
##     0.2689
##
## Description:
##  Mon Aug 03 21:43:20 2020 by user: Oscar
```

```
unitrootTest(train)
```

```
##
## Title:
```

```
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     DF: -1.0843
##   P VALUE:
##     t: 0.2512
##     n: 0.4567
##
## Description:
##  Mon Aug 03 21:43:20 2020 by user: Oscar
```

Como podemos notar, en ambas pruebas el valor de p es mayor a 0.05 por lo que no se puede rechazar la hipótesis nula de ausencia de raíces unitarias. Entonces por ende no es estacionaria en media.

A coninuación hacemos lo mismo pero con una diferenciación:

```
adfTest(diff(train))
```

```
## Warning in adfTest(diff(train)): p-value smaller than printed p-value
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -15.1922
##   P VALUE:
##     0.01
##
## Description:
##  Mon Aug 03 21:43:20 2020 by user: Oscar
```

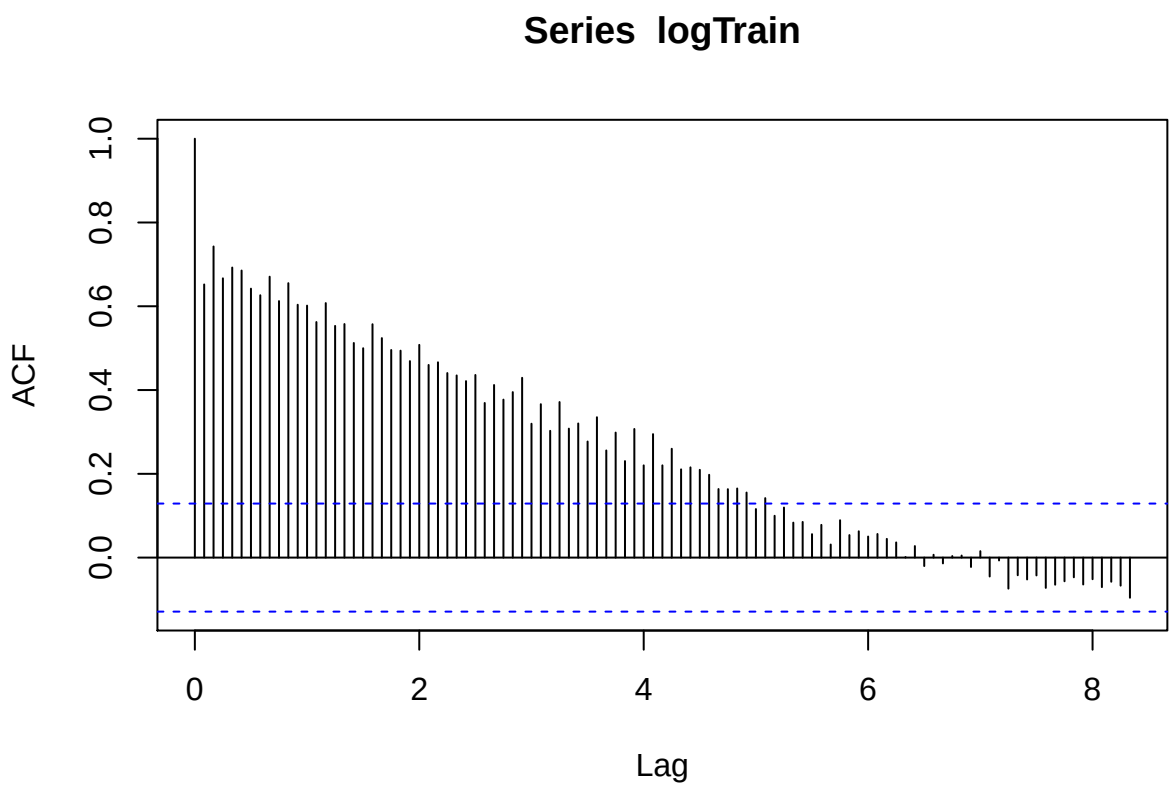
```
unitrootTest(diff(train))
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     DF: -15.1922
##   P VALUE:
##     t: < 2.2e-16
##     n: 0.005762
##
## Description:
##  Mon Aug 03 21:43:20 2020 by user: Oscar
```

Como podemos observar en esta ocasión el valor de p está por debajo de 0.05 por lo que ahora si se puede descartar la hipótesis nula de que existen raíces unitarias. Podemos notar entonces que solo es necesaria una diferenciación (d=1).

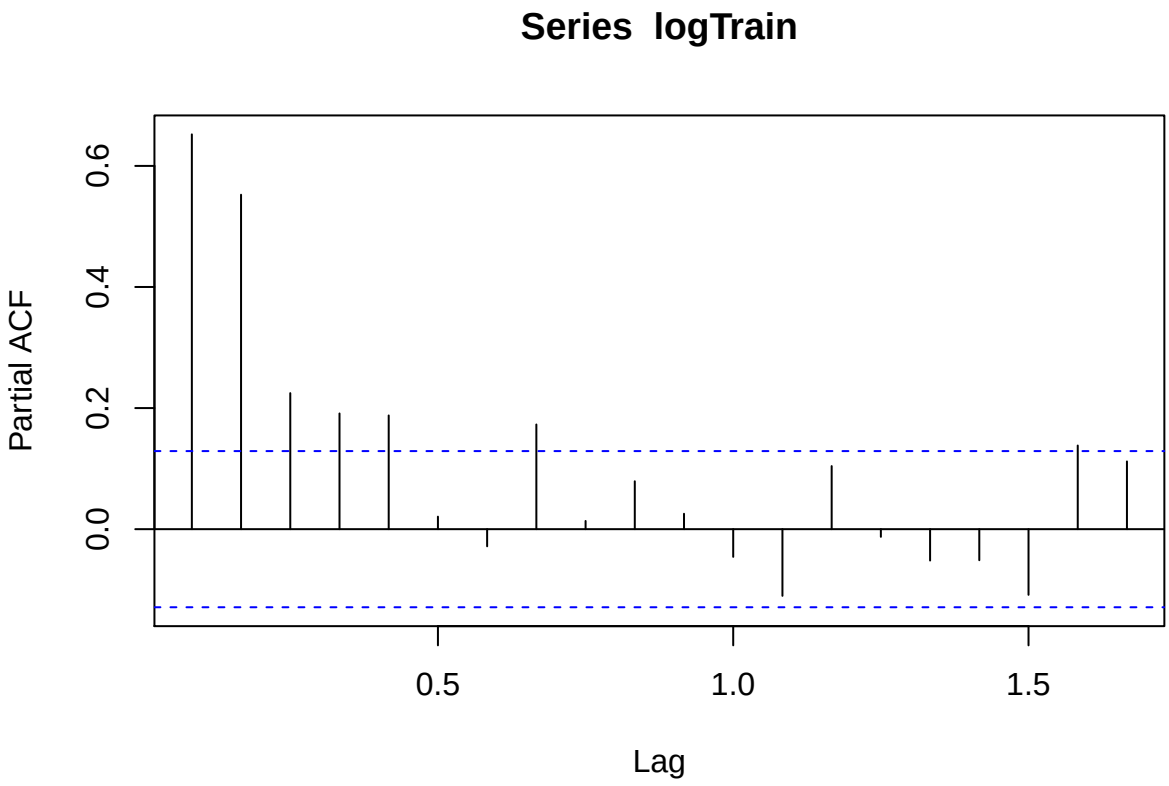
El siguiente paso es intentar identificar los parámetros p y q usando los gráficos de autocorrelación y autocorrelación parcial.

```
acf(logTrain,100)
```



En el gráfico de la función podemos notar que se anula después del quinto retardo por lo que se puede sugerir q=5.

```
pacf(logTrain,20)
```



En el gráfico de la función podemos notar que se anula después del retardo 0 por lo que podríamos sugerir un coeficiente p=0.

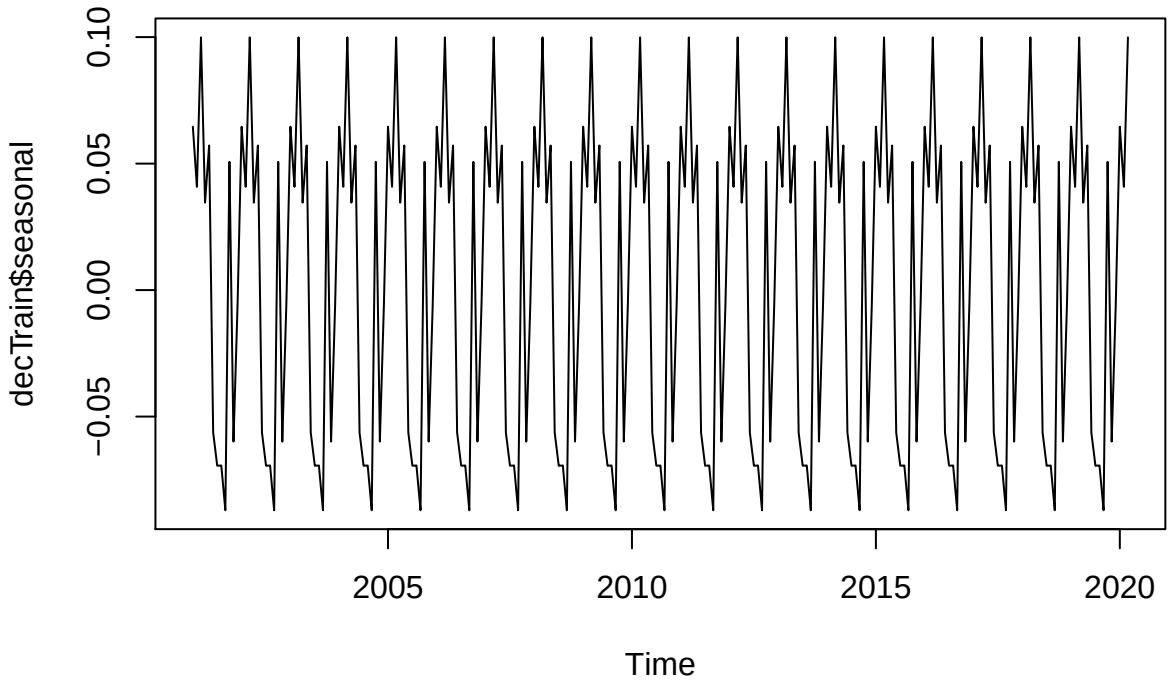
Podemos sugerir un modelo con los siguientes parámetros:

- p=0.
- q=5.
- d=1.

O en otras palabras ARIMA(p,d,q).

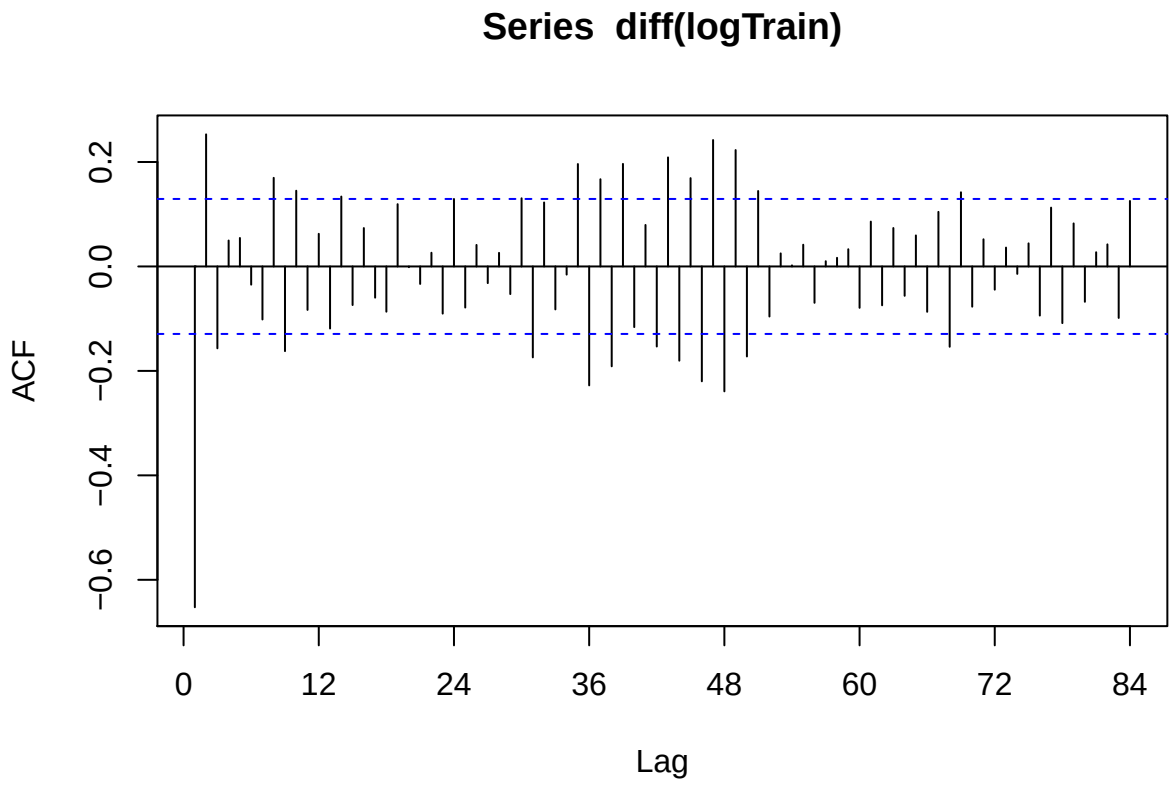
Volvamos a ver la descomposición de la serie paran determinar si hay estacionalidad

```
decTrain <- decompose(logTrain)
plot(decTrain$seasonal)
```



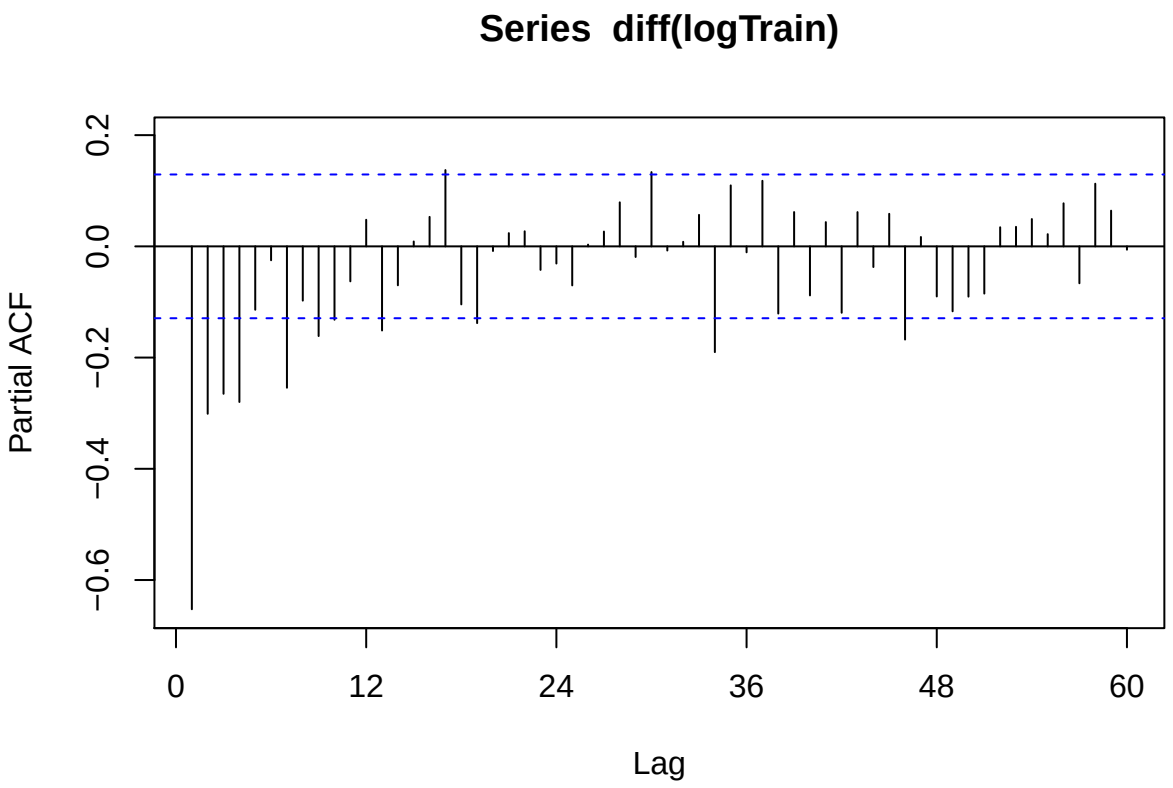
Podemos notar que si existe estacionalidad en la serie. A continuación, se buscarán los parámetros estacionales usando las funciones de autocorrelación y autocorrelación parcial.

```
Acf(diff(logTrain),84)
```



Podemos notar que en casi todos los períodos hay dos decaimientos estacionales por lo que podemos sugerir P=2.

```
Pacf(diff(logTrain),60)
```



Podemos darnos cuenta que los pico significativos están al inicio de la gráfica por lo que podemos sugerir D=1. Finalmente Q=0.

Con los parámetros determinados generamos un modelo:

```
fitArima <- arima(logTrain,order=c(0,1,5),seasonal = c(2,1,0))
```

R tiene la capacidad de generar un modelo automáticamente, entonces también lo tomaremos en cuenta:

```
fitAutoArima <- auto.arima(train)
```

Significación de los coeficientes:

A continuación, analizaremos que tan significativos son los coeficientes de cada modelo

```
coeftest(fitArima)
```

```
##
## z test of coefficients:
##
##          Estimate   Std. Error   z value   Pr(>|z|)
## ma1  -1.149050254  0.068498160 -16.77491 < 2.22e-16 ***
## ma2   0.344195857  0.106848232   3.22135  0.0012759 **
## ma3  -0.271126271  0.121191736  -2.23717  0.0252754 *
## ma4   0.189132817  0.117590289   1.60841  0.1077465
```

```
## ma5  -0.006775789  0.070339966  -0.09633  0.9232592
## sar1 -0.659578707  0.068126852  -9.68163 < 2.22e-16 ***
## sar2 -0.169097870  0.068135767  -2.48178  0.0130729 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

En este caso podemos notar no todos los coeficientes son significativos.

```
coeftest(fitAutoArima)
```

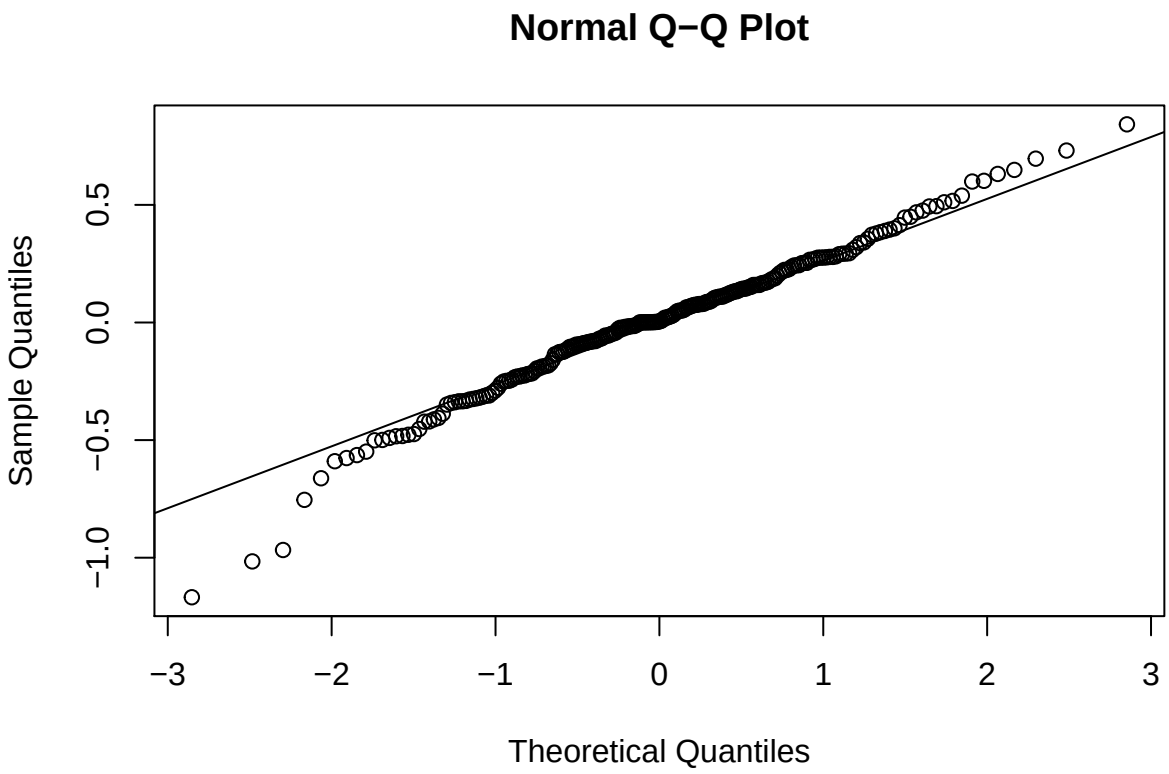
```
##
## z test of coefficients:
##
##         Estimate  Std. Error  z value Pr(>|z|)
## ar1  -1.08757649  0.12598956 -8.63227  < 2e-16 ***
## ar2  -0.14060995  0.09642382 -1.45825  0.14477
## ma1  -0.02683433  0.09812491 -0.27347  0.78449
## ma2  -0.77705945  0.08363777 -9.29077  < 2e-16 ***
## sar1  0.06336379  0.08253527  0.76772  0.44265
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Podemos notar que todos los coeficientes son significativo ya que todos son menores a 0.05.

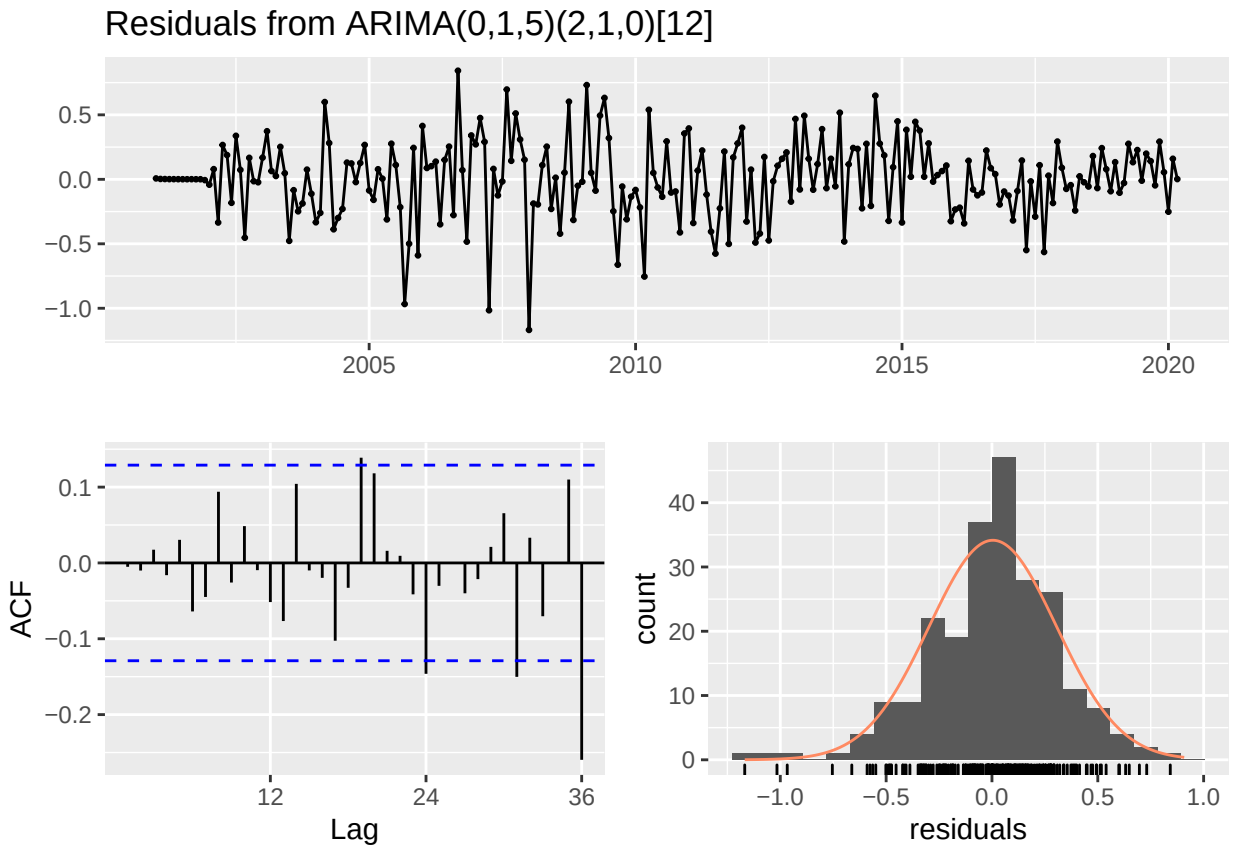
Análisis de Residuales

A continuación, se analizarán los residuales de ambos modelos. Estos deben estar distribuidos normalmente y se deben de parecer al ruido blanco.

```
qqnorm(fitArima$residuals)
qqline(fitArima$residuals)
```




```
checkresiduals(fitArima)
```

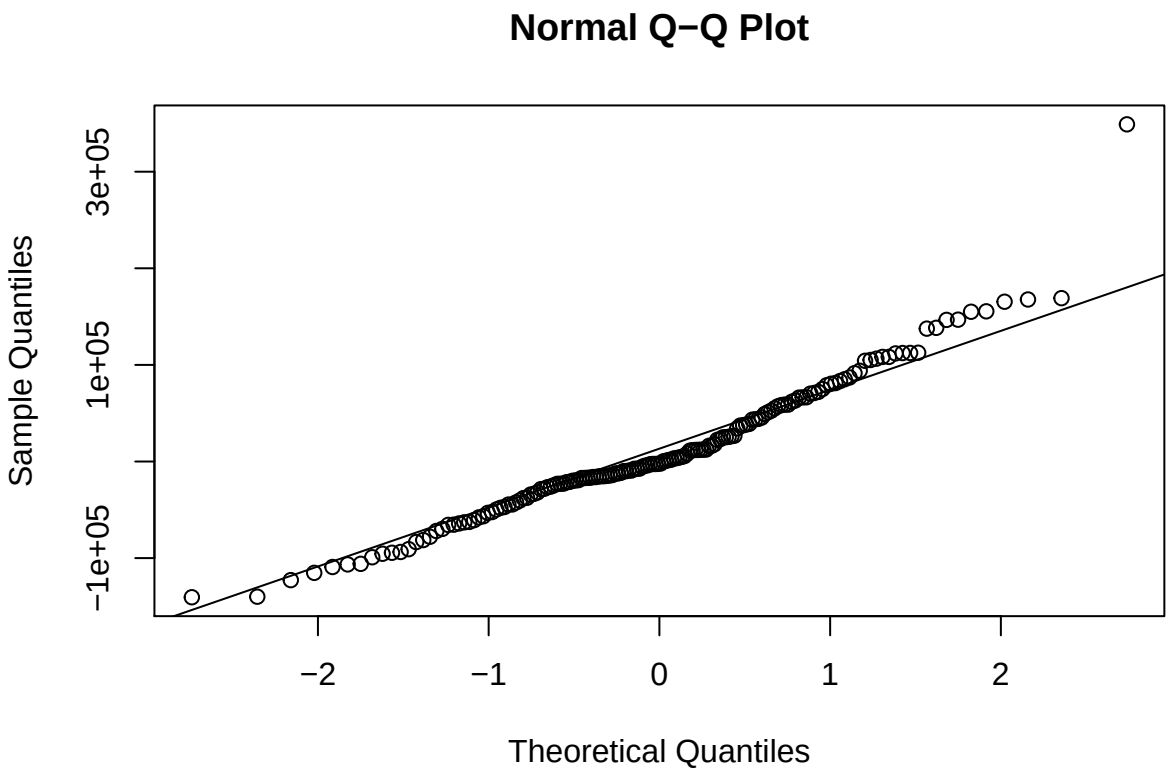



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,5)(2,1,0)[12]
## Q* = 27.10699, df = 17, p-value = 0.0565173
##
## Model df: 7.   Total lags used: 24
```

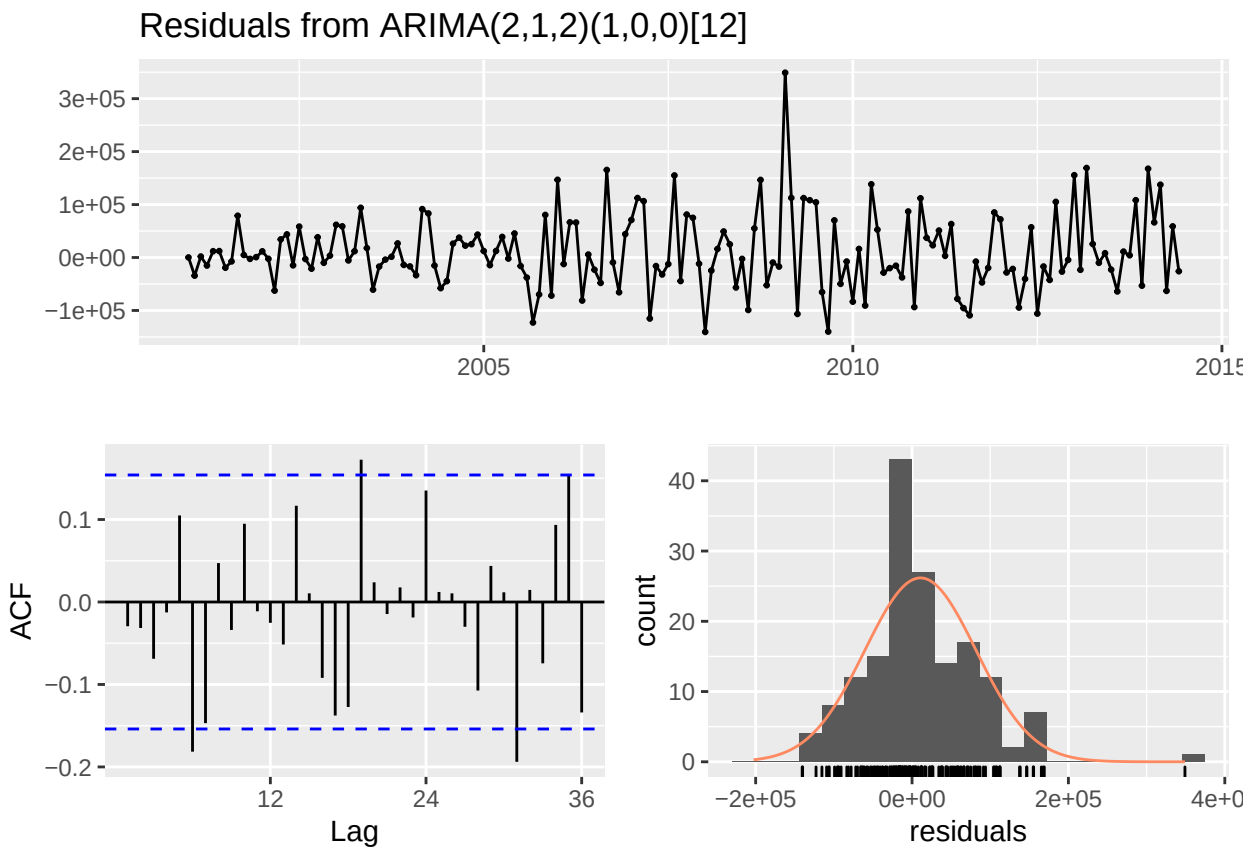
Podemos notar que los datos tienen una distribución normal. El test de Ljung-Box muestra que los datos se distribuyen de forma independiente puesto que el p-value es mayor a 0.05 y por lo que no se puede rechazar la hipótesis nula. Esto quiere decir que el modelo es aceptable para predecir.

Analicemos también el modelo generado automáticamente por R:

```
qqnorm(fitAutoArima$residuals)
qqline(fitAutoArima$residuals)
```




```
checkresiduals(fitAutoArima)
```



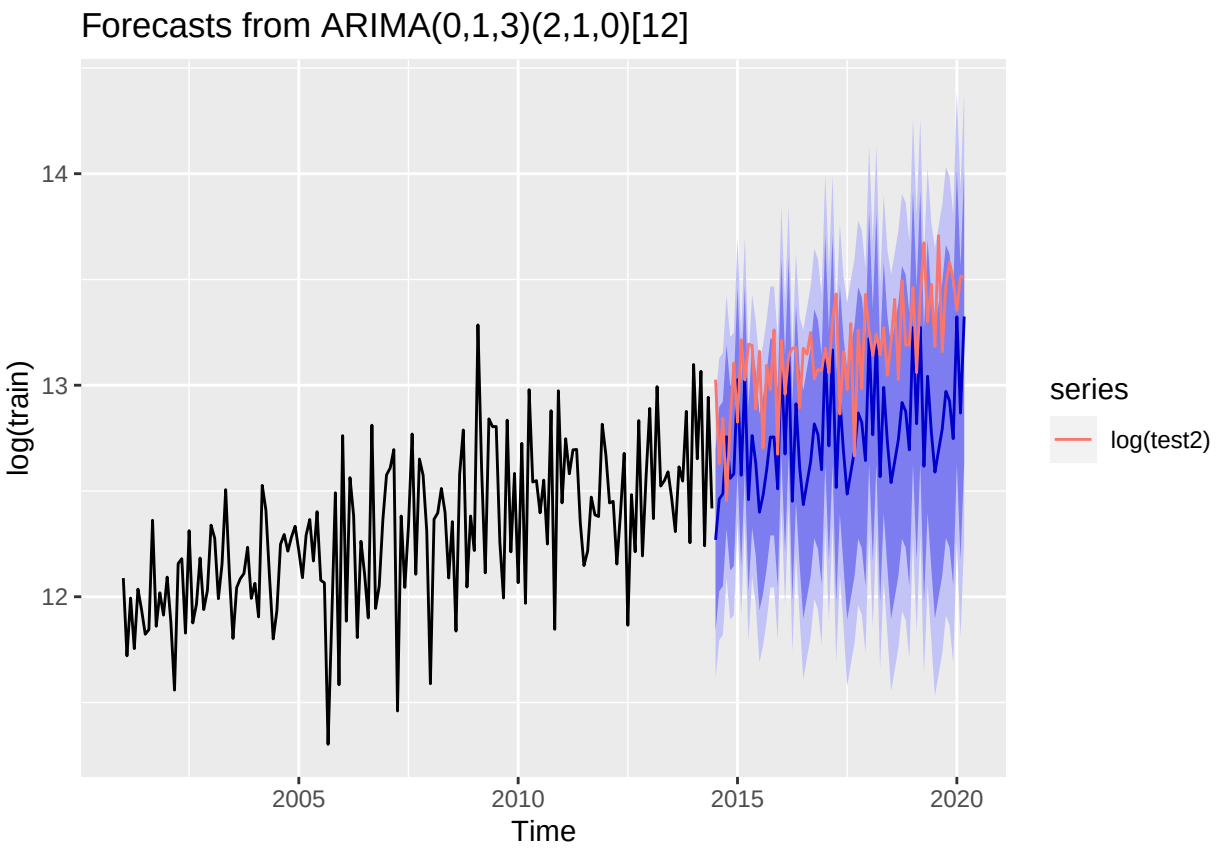

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,2)(1,0,0)[12]
## Q* = 34.84239, df = 19, p-value = 0.0145897
##
## Model df: 5.   Total lags used: 24
```

Podemos notar que los datos tienen una distribución normal. Sin embargo, según el test de Ljung-Box los datos se distribuyen de forma dependiente puesto que el p-value es menor a 0.05 y se puede rechazar la hipótesis nula. Esto quiere decir que el modelo no es aceptable para predecir.

Predicción con el modelo generado

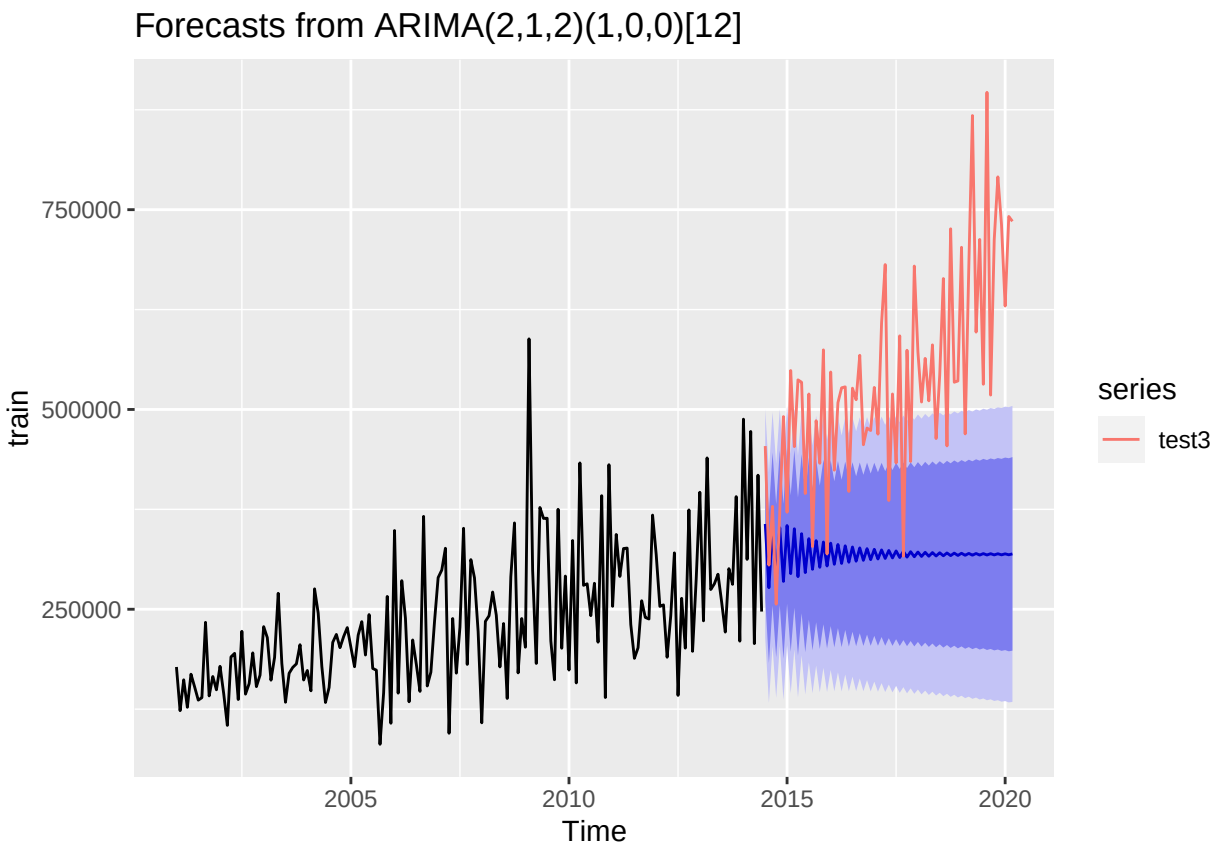
Pasaremos a predecir con la función forecast la misma cantidad de meses que hay en test y compararemos con los valores reales.

```
fitArima <- arima(log(train),c(0,1,3),seasonal =list( order=c(2,1,0),period=12))
test2<-ts(test,start = c(2014,7),end = c(2020,3),frequency = 12)
fitArima %>%
  forecast(h) %>%
  autoplot() + autolayer(log(test2))
```



Podemos notar que nuestro modelo es bueno prediciendo, ya que las predicciones están dentro del intervalo de confianza.

```
fitAutoArima <- auto.arima(train)
test3<-ts(test,start = c(2014,7),end = c(2020,3),frequency = 12)
fitAutoArima %>%
  forecast(h) %>%
  autoplot() + autolayer(test3)
```



Podemos notar que el modelo realizado automáticamente por R no predice muy bien, ya que no se parece en nada a los datos reales. Entonces podemos decir que nosotros hemos construido un mejor modelo al que genera automáticamente R.

Construcción del modelo Prophet

Preparamos los datos como los necesita este algoritmo:

```
df<-data.frame(ds=as.Date(as.yearmon(time(train))),y=as.matrix(train) )
testdf<-data.frame(ds=as.Date(as.yearmon(time(test))),y=as.matrix(test) )
```

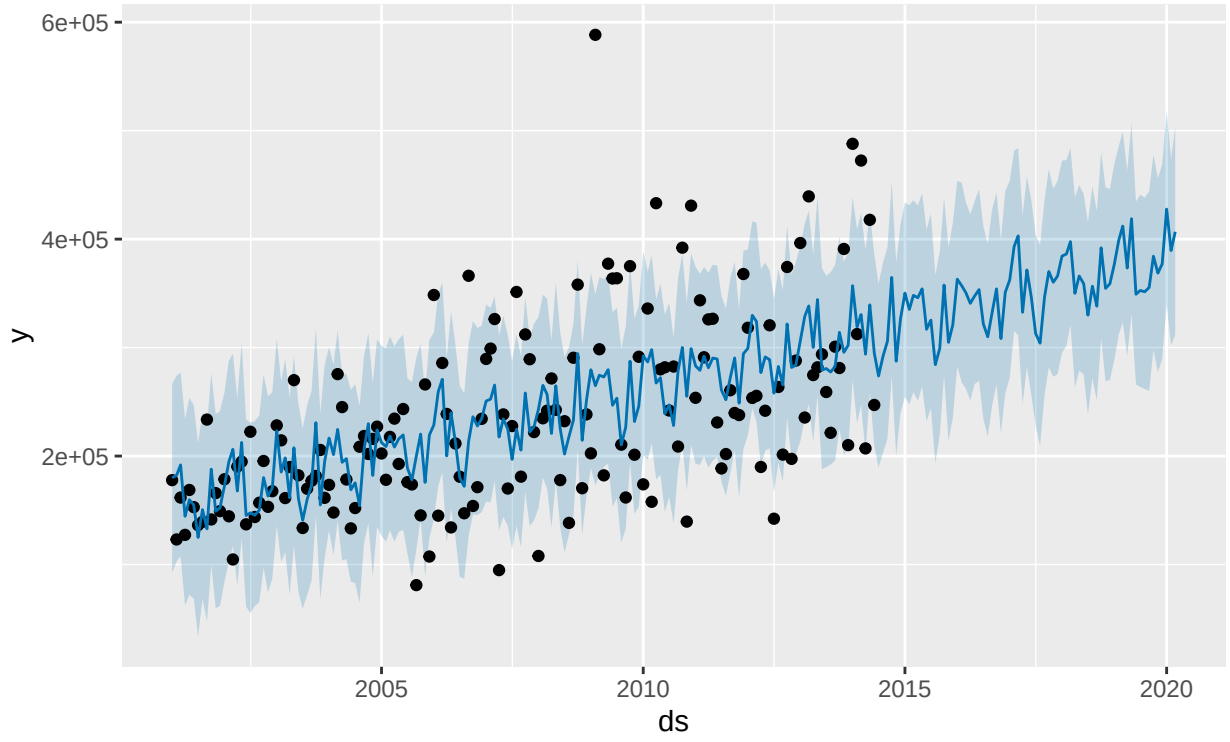
A continuación, elaboramos el modelo:

```
fitProphet<-prophet(df,yearly.seasonality = T,weekly.seasonality = T)
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

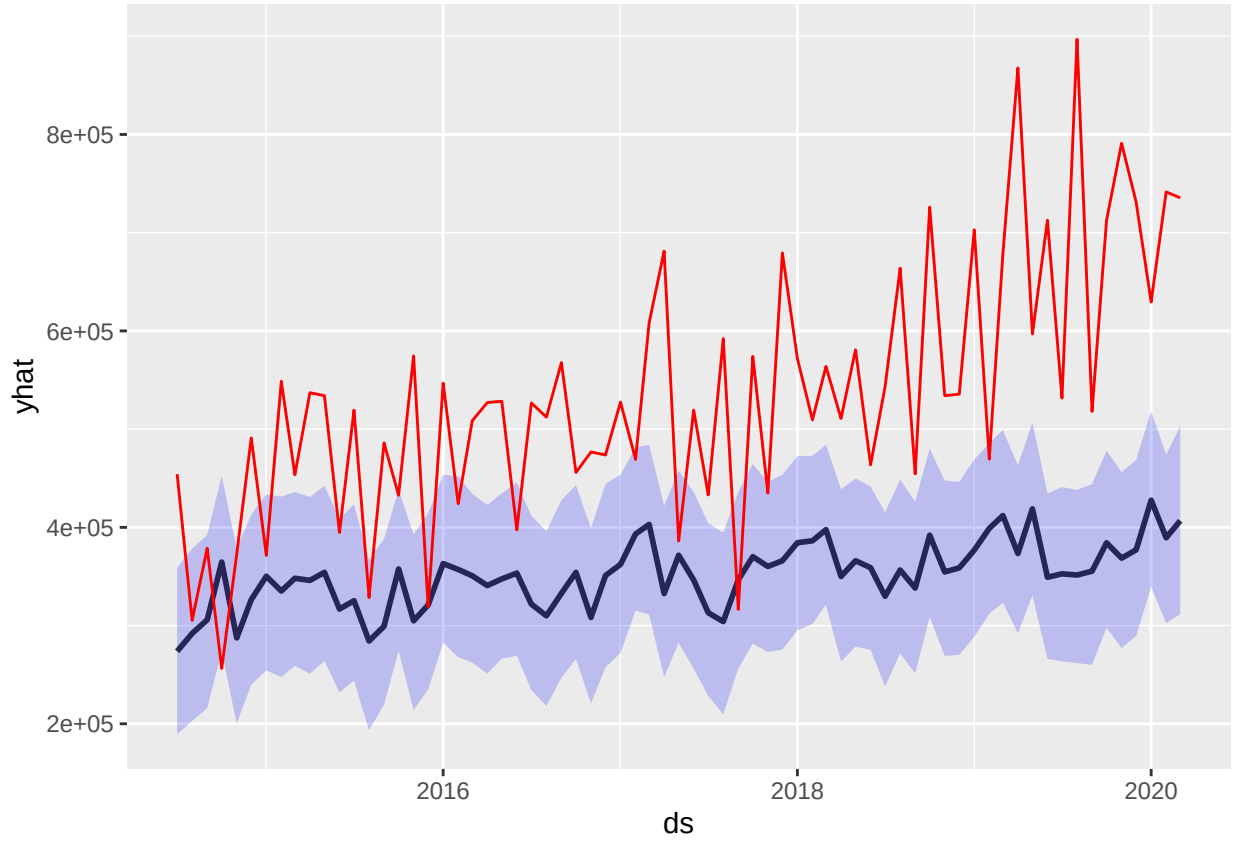
Usaremos el modelo para predecir la misma cantidad de meses que el modelo ARIMA:

```
future <- make_future_dataframe(fitProphet,periods = h,freq = "month", include_history = T)
p <- predict(fitProphet,future)
p<-p[,c("ds","yhat","yhat_lower","yhat_upper")]
plot(fitProphet,p)
```




```
pred<-tail(p,h)
pred$y<-testdf$y

ggplot(pred, aes(x=ds, y=yhat)) +
   geom_line(size=1, alpha=0.8) +
   geom_ribbon(aes(ymin=yhat_lower, ymax=yhat_upper), fill="blue", alpha=0.2) +
   geom_line(data=pred, aes(x=ds, y=y),color="red")
```



Podemos notar que el modelo Prophet no es muy bueno, por que no se encuentra en los intervalos de confianza. En comparación con el modelo que realizamos ARIMA , nuestro predicción se parece más a los datos reales por lo que podemos decir que el modelo ARIMA es mejor.

Construcción del modelo ARIMA GAS REGULAR

Creamos la serie de tiempo

```
superior <- ts(dataSet$GasSuperior, start=c(2001, 1), end=c(2020, 3), frequency=12)
```

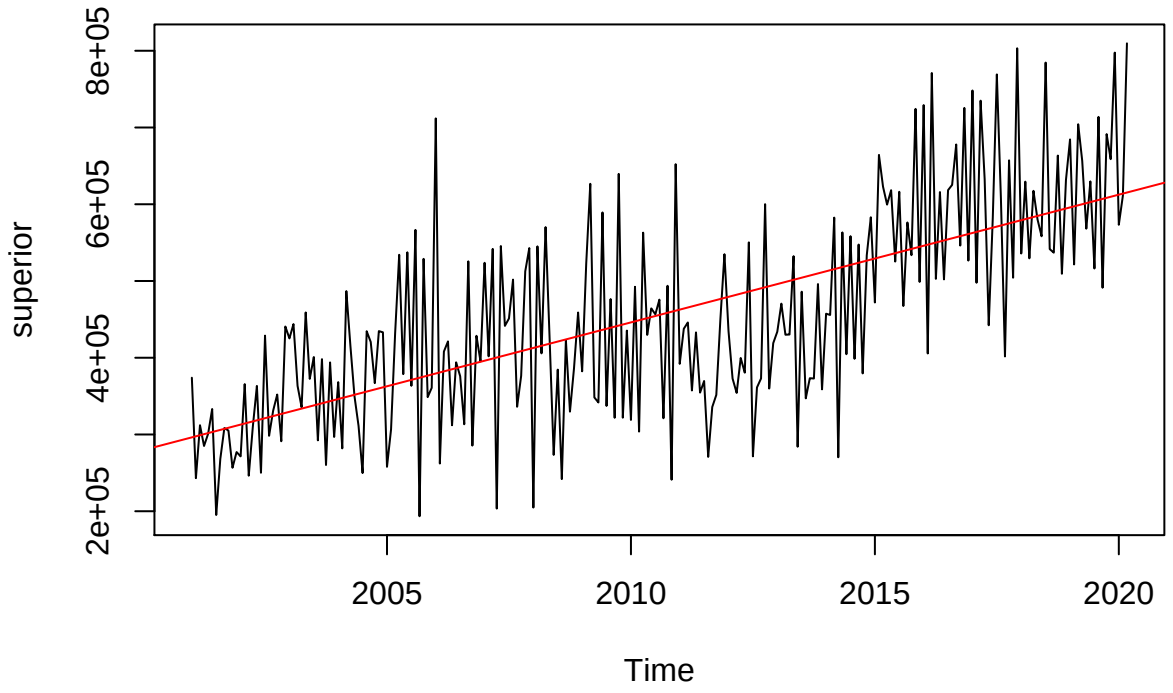
Establecemos datos a usar

```
train <- head(superior, round(length(superior) * 0.7))
h <- length(superior) - length(train)
test <- tail(superior, h)
```

Iniciamos la construcción del modelo

Primero, analizaremos el gráfico de la serie

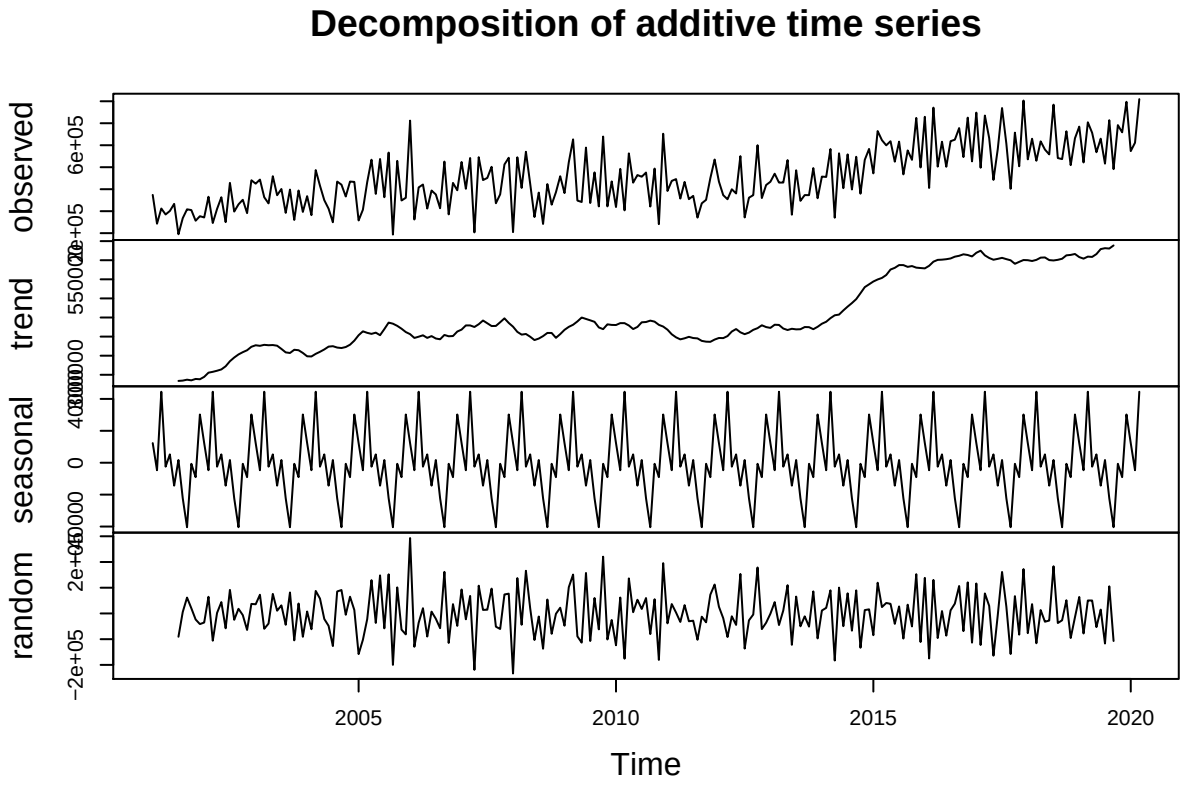
```
plot(superior)
abline(reg=lm(superior~time(superior)), col=c("red"))
```



Es una serie con frecuencia anual desde enero 2001 hasta marzo 2020. La línea de la media nos indica que la serie tiene una tendencia a crecer. Cabe destacar que, durante los años 2011 y 2015 el precio se mantuvo estable, es hasta después del 2015 que el precio creció de nuevo.

Descomposición de la serie

```
dec.superior<-decompose(superior)
plot(dec.superior)
```



La tendencia no es estacionaria en varianza, pues los rangos varían bastante con el tiempo. La serie tiene estacionalidad.

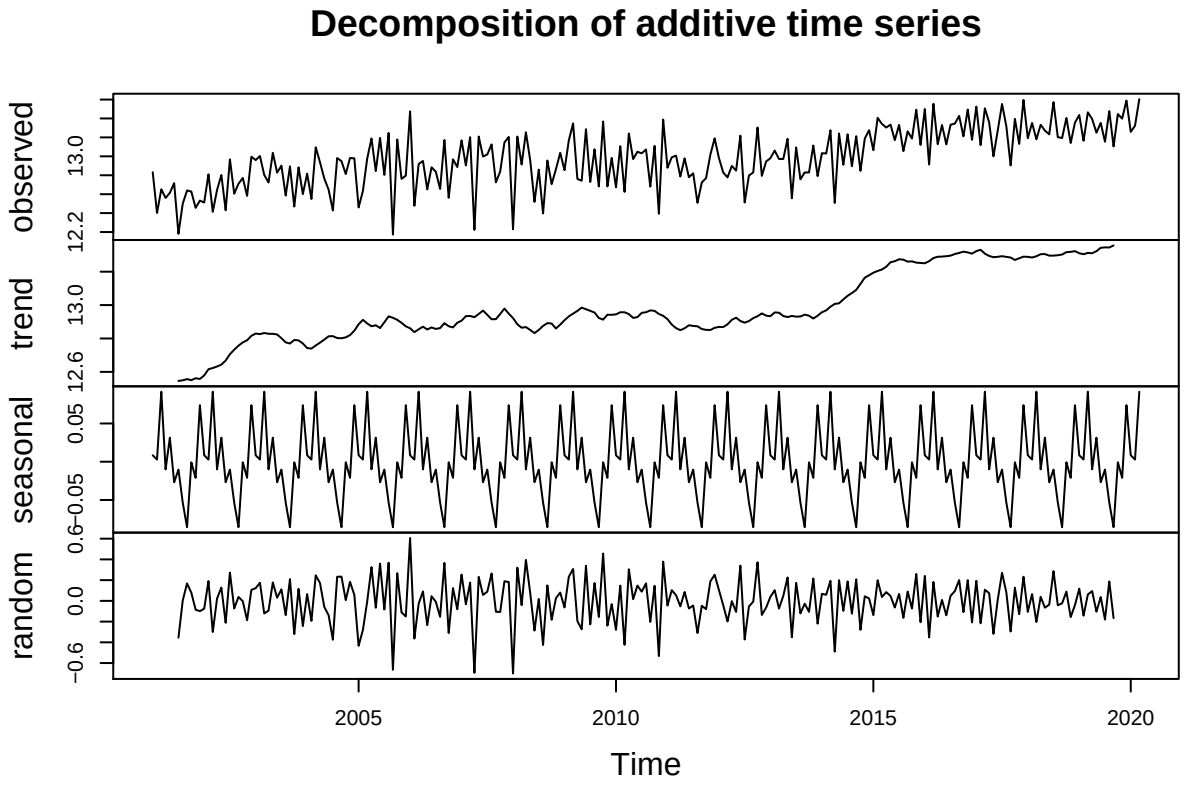
Estimar parámetros del modelo

Dado que no es estacionaria en varianza le aplicaremos una transformación logaritmica para hacerla constante en varianza. Lo haremos con la serie de entrenamiento que es la que nos ayudará a predecir

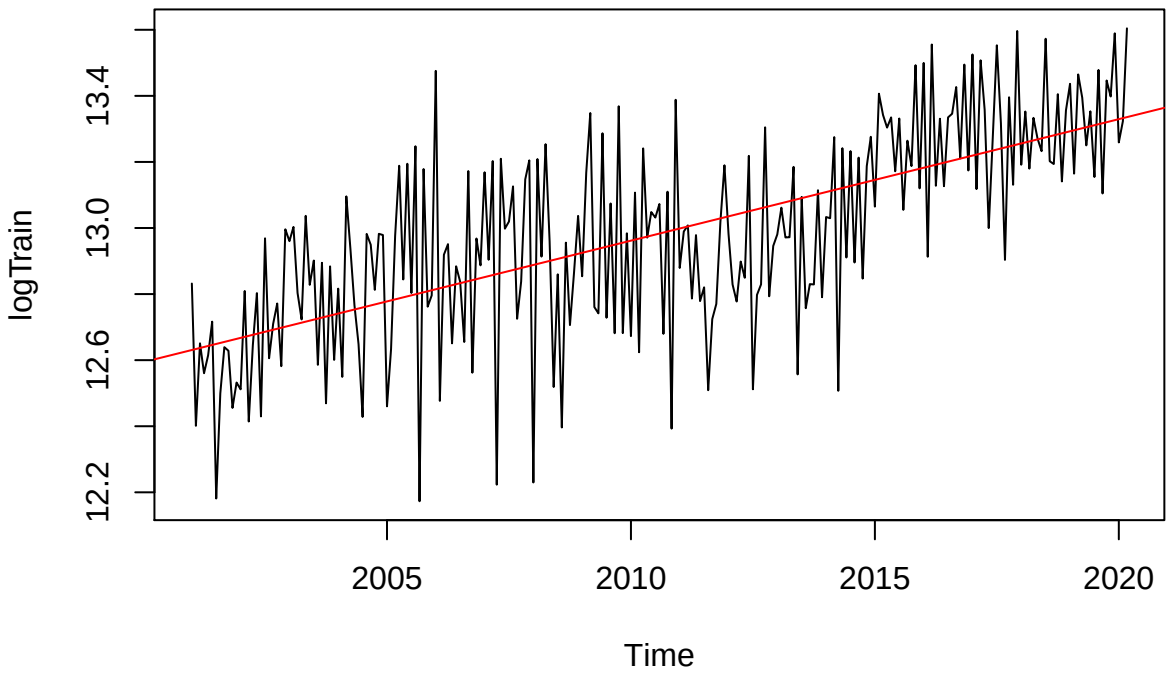
```
logTrain <- log(superior)
```

Intento de estacionalizar la serie con una transformación logarítmica.

```
plot(decompose(logTrain))
```




```
plot(logTrain)
abline(reg=lm(logTrain~time(logTrain)), col=c("red"))
```



Al parecer se logra mejorar que la varianza sea un poco más constante. Debemos verificar si es estacionaria en media. Si tiene raices unitarias podemos decir que no es estacionaria en media y hay que aplicar procesos de diferenciación. Uno de los factores que nos llevan a pensar que en efecto la varianza es un poco más constante es que la gráfica posee menos oscilación.

Usando la prueba de Dickey-Fuller

```
adfTest(train)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -0.7879
##   P VALUE:
##     0.3633
##
## Description:
##  Mon Aug 03 21:43:27 2020 by user: Oscar
```

El valor p es mayor a 0.05, por lo que no se puede rechazar la hipótesis nula de las raices unitarias.

Segunda prueba

```
unitrootTest(train)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     DF: -0.7879
##   P VALUE:
##     t: 0.3734
##     n: 0.5102
##
## Description:
##  Mon Aug 03 21:43:27 2020 by user: Oscar
```

El valor p tampoco es mayor a 0.05. Esto significa que la serie no es estacionaria en media.

Aplicando diferenciación

Se aplicará una diferenciación con el fin de hacer la serie estacionario en media.

```
adfTest(diff(train))
```

```
## Warning in adfTest(diff(train)): p-value smaller than printed p-value
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -16.4935
##   P VALUE:
##     0.01
##
## Description:
##  Mon Aug 03 21:43:27 2020 by user: Oscar
```

```
unitrootTest(diff(train))
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
```

```
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     DF: -16.4935
##   P VALUE:
##     t: < 2.2e-16
##     n: 0.003944
##
## Description:
##  Mon Aug 03 21:43:27 2020 by user: Oscar
```

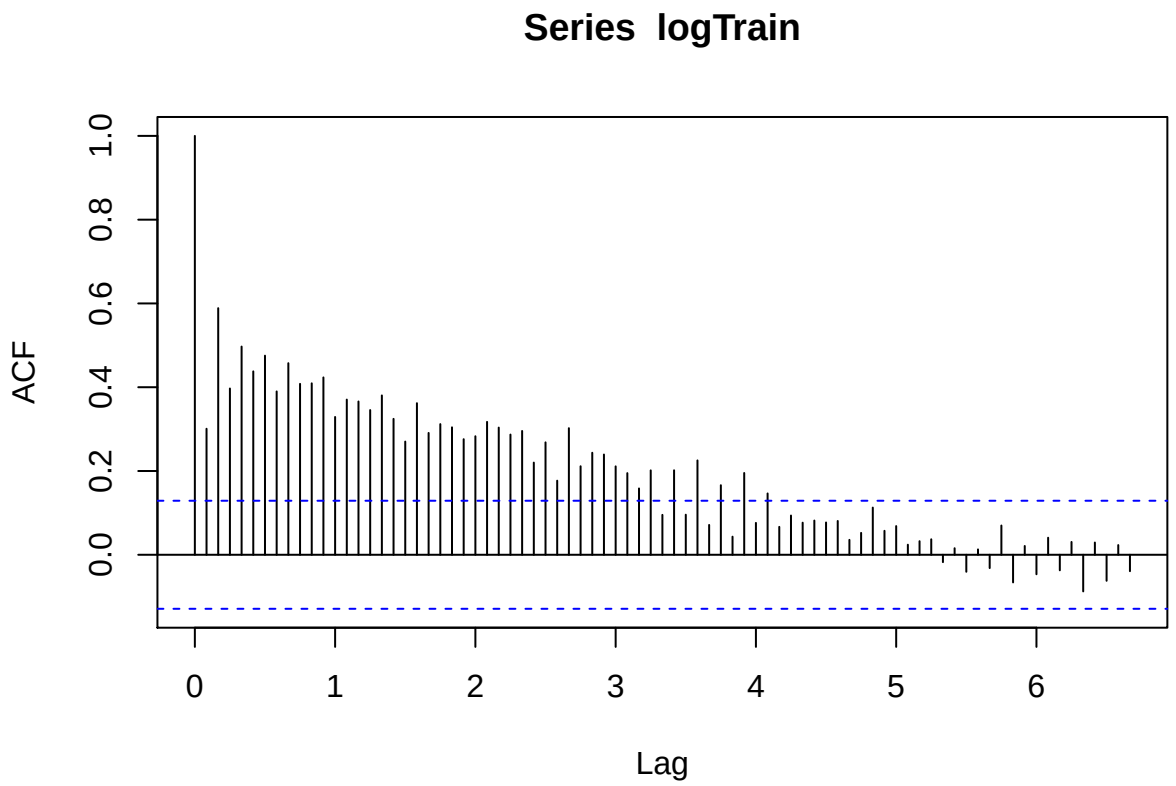
En ambas pruebas el valor p es menor a 0.05. Con una sola diferenciación se puede rechazar la hipótesis nula de las raíces unitarias. Por lo tanto:

- d = 1

Identificando parámetros p y q

Función de autocorrelación:

```
acf(logTrain,80)
```

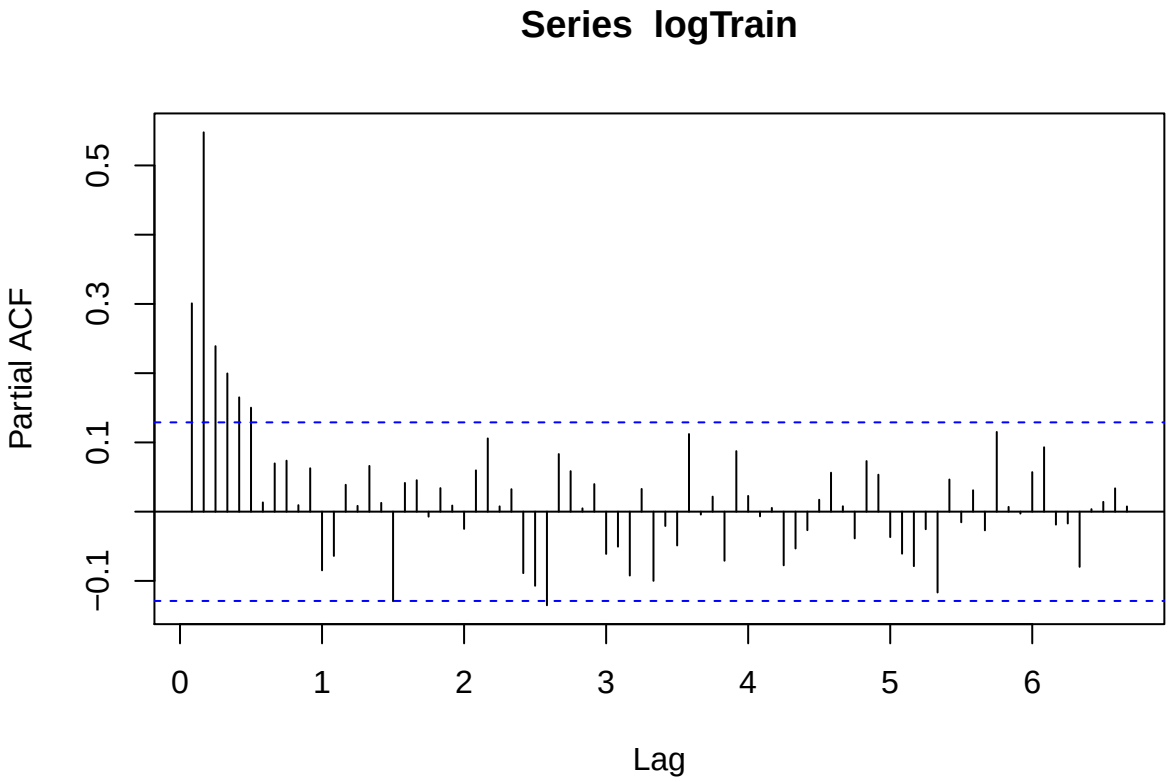


Con respecto al presente gráfico, podemos ver que se anula el valor j se anula luego del tercer retardo.

- Se propone un valor q = 3

Función de correlación parcial

```
pacf(logTrain,80)
```



Se anula antes del primer retardo, por ello:

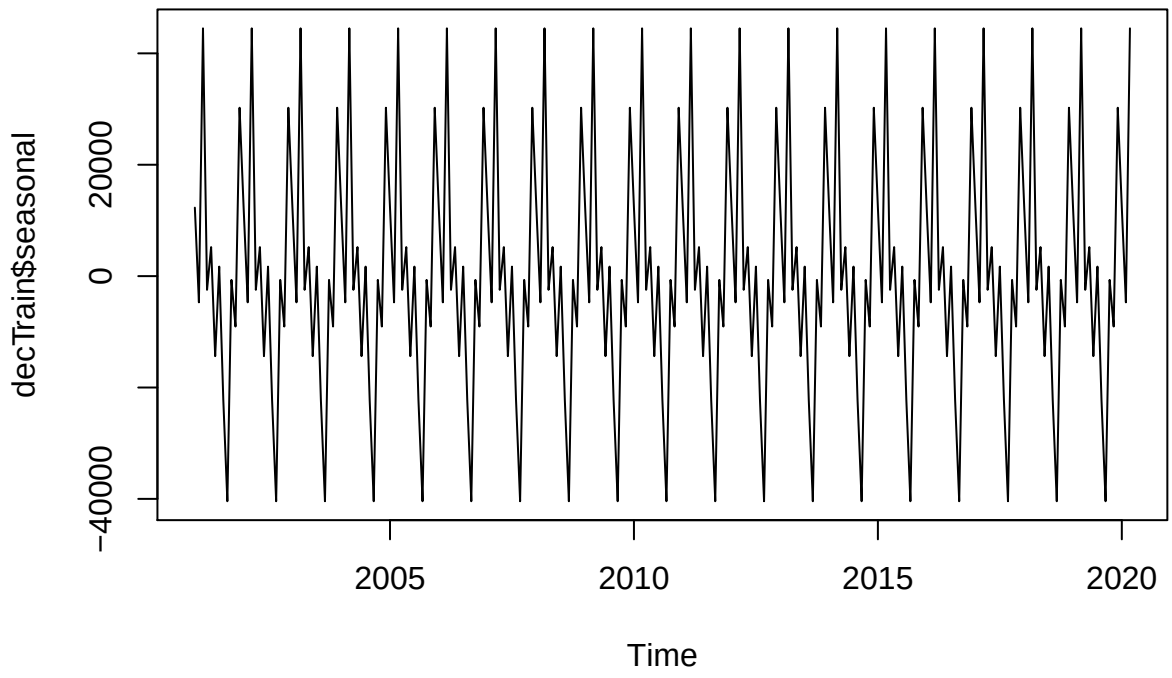
- Se propone un valor p = 0

Resumen de parámetros:

- d = 1
- p = 0
- q = 3
- ARIMA(0,1,3)

¿Estacionalidad en la serie?

```
decTrain <- decompose(superior)
plot(decTrain$seasonal)
```

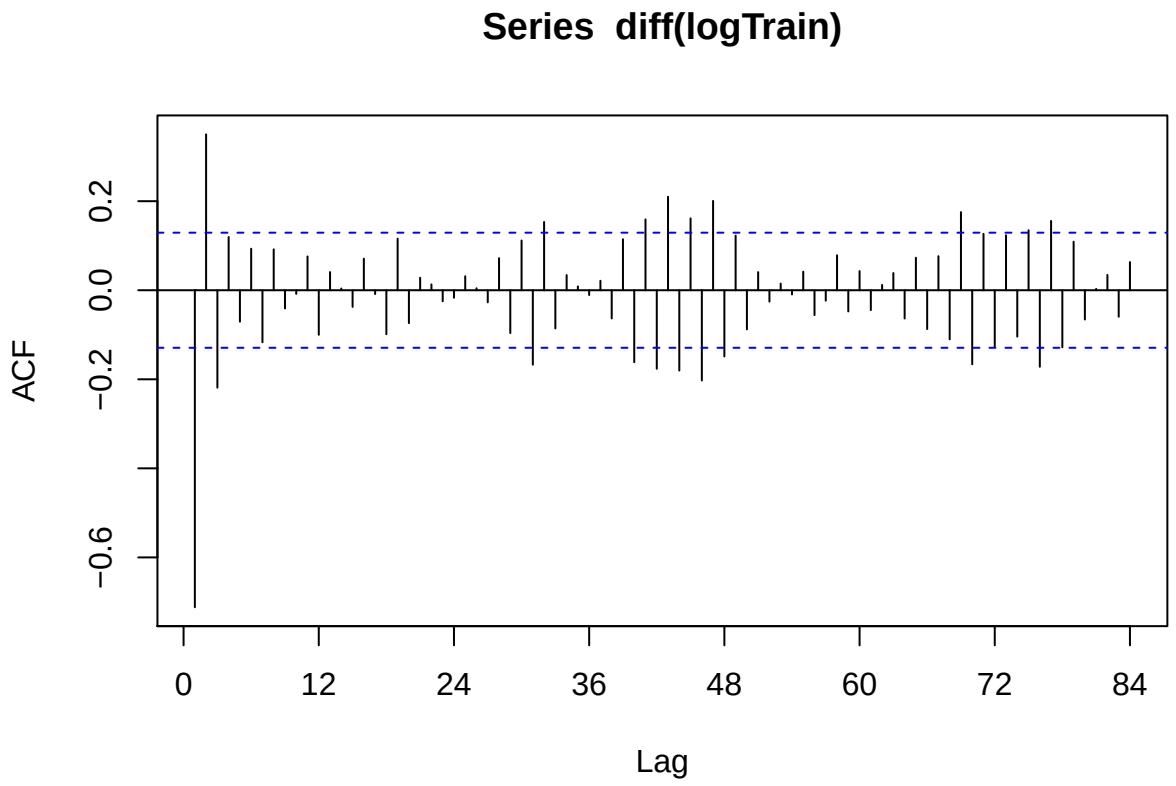


A simple vista, parece que la serie sí tiene estacionalidad.

Uso de la función de autocorrelación

Se usará la serie estacionarizada.

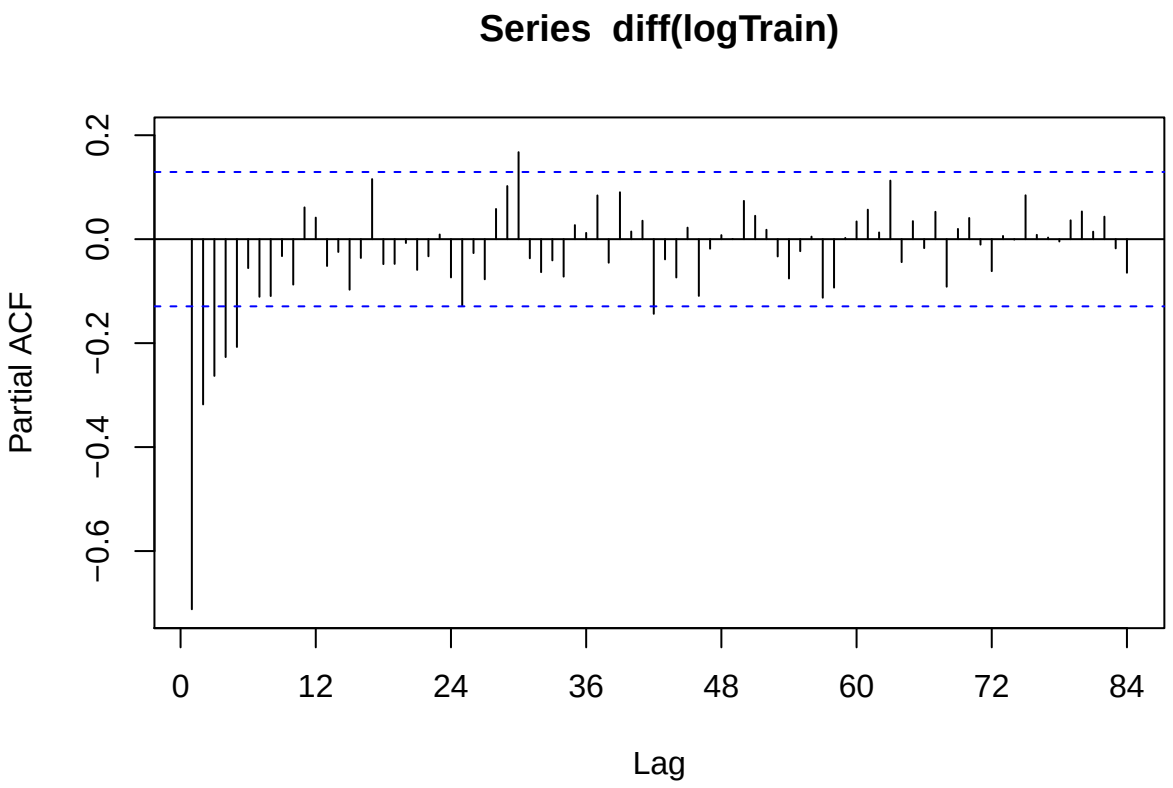
```
Acf(diff(logTrain),84)
```



Los períodos poseen 2 decaimientos estacionales. Por ello:

- P = 2

```
Pacf(diff(logTrain),84)
```



Los picos significativos están durante el primer período. Por ello:

- Q = 1
- D = 0

Generación del modelo

Modelo con parámetros obtenidos y modelo automático, respectivamente:

```
fitArima <- arima(log(train),c(0,1,3),seasonal =list( order=c(2,1,0),period=12))
fitAutoArima <- auto.arima(train)
```

Validamos si los coeficientes son significativos:

```
coeftest(fitArima)
```

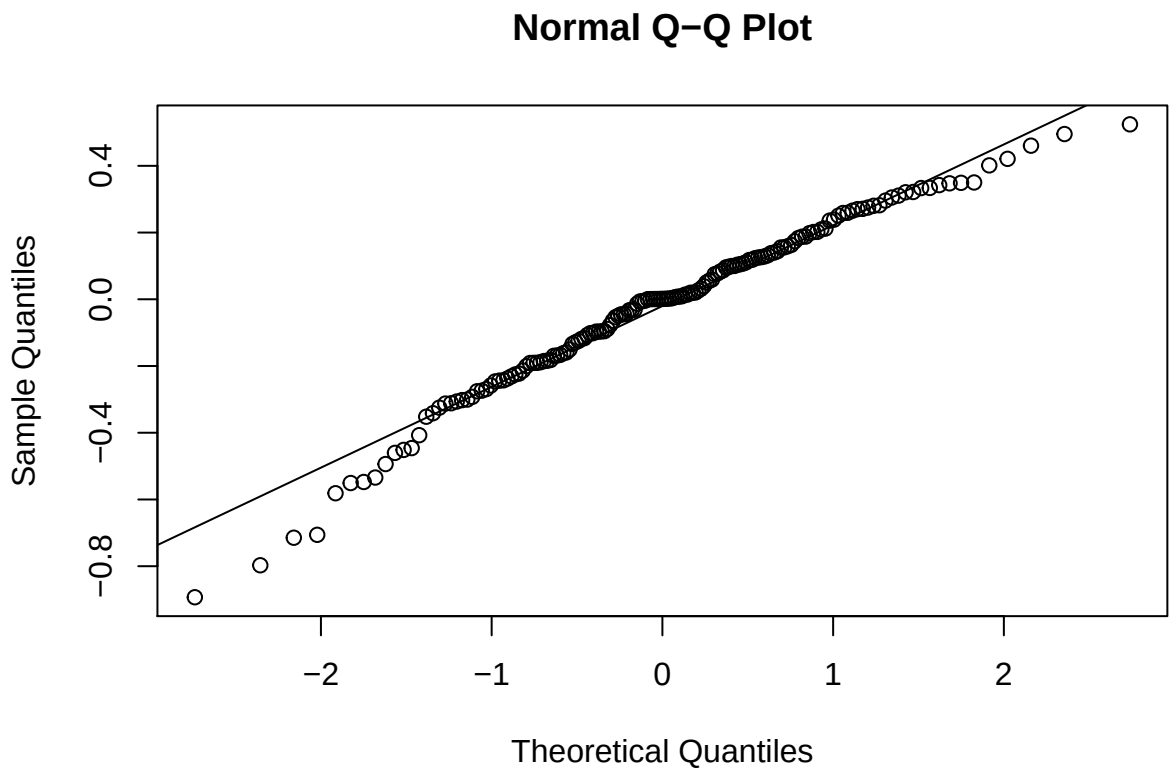
```
##
## z test of coefficients:
##
##         Estimate  Std. Error   z value   Pr(>|z|)
## ma1  -1.31861284  0.08178088 -16.12373 < 2.22e-16 ***
## ma2   0.44399595  0.11157682   3.97928 6.9123e-05 ***
## ma3  -0.07743204  0.07334852  -1.05567    0.29112
## sar1 -0.78981619  0.07699064 -10.25860 < 2.22e-16 ***
## sar2 -0.39232767  0.07567892  -5.18411 2.1705e-07 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

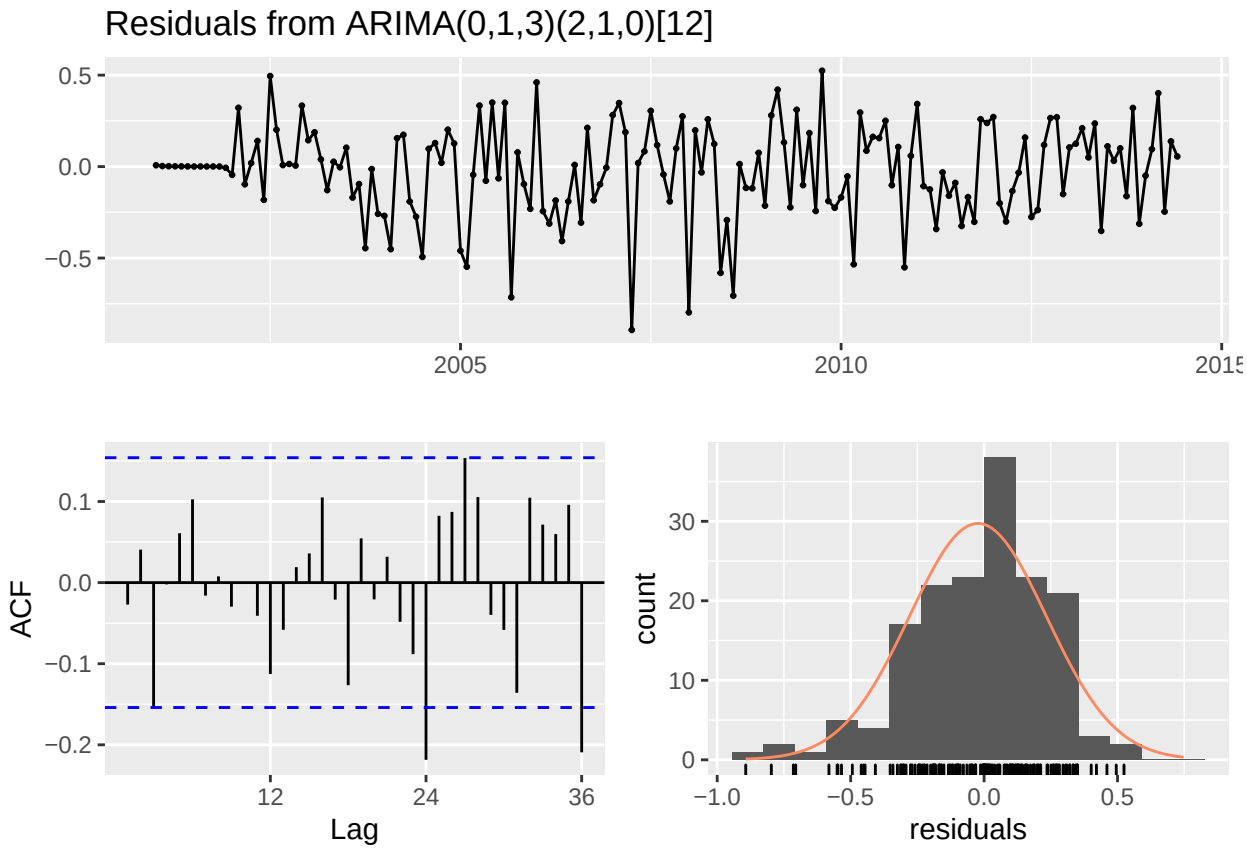
No todos son menores a 0.05, por lo que no son significativos

Análisis de residuales

```
qqnorm(fitArima$residuals)
qqline(fitArima$residuals)
```




```
checkresiduals(fitArima)
```

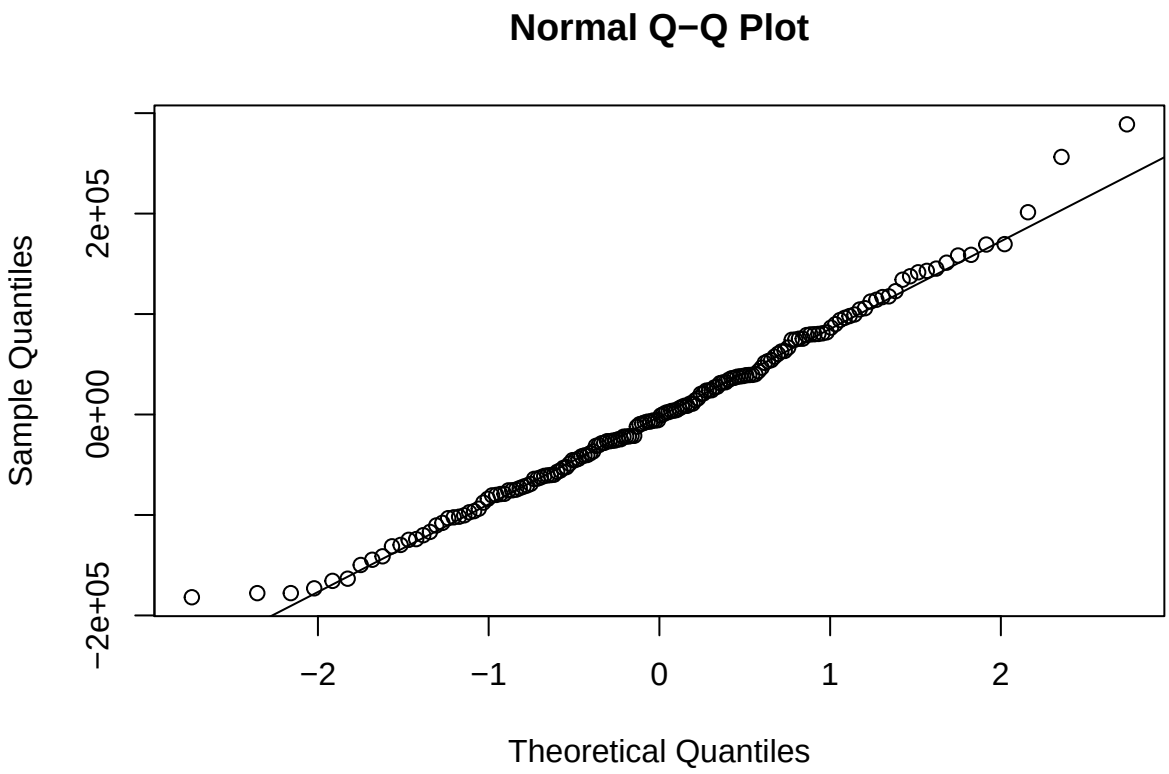



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,3)(2,1,0)[12]
## Q* = 27.48372, df = 19, p-value = 0.0938807
##
## Model df: 5.   Total lags used: 24
```

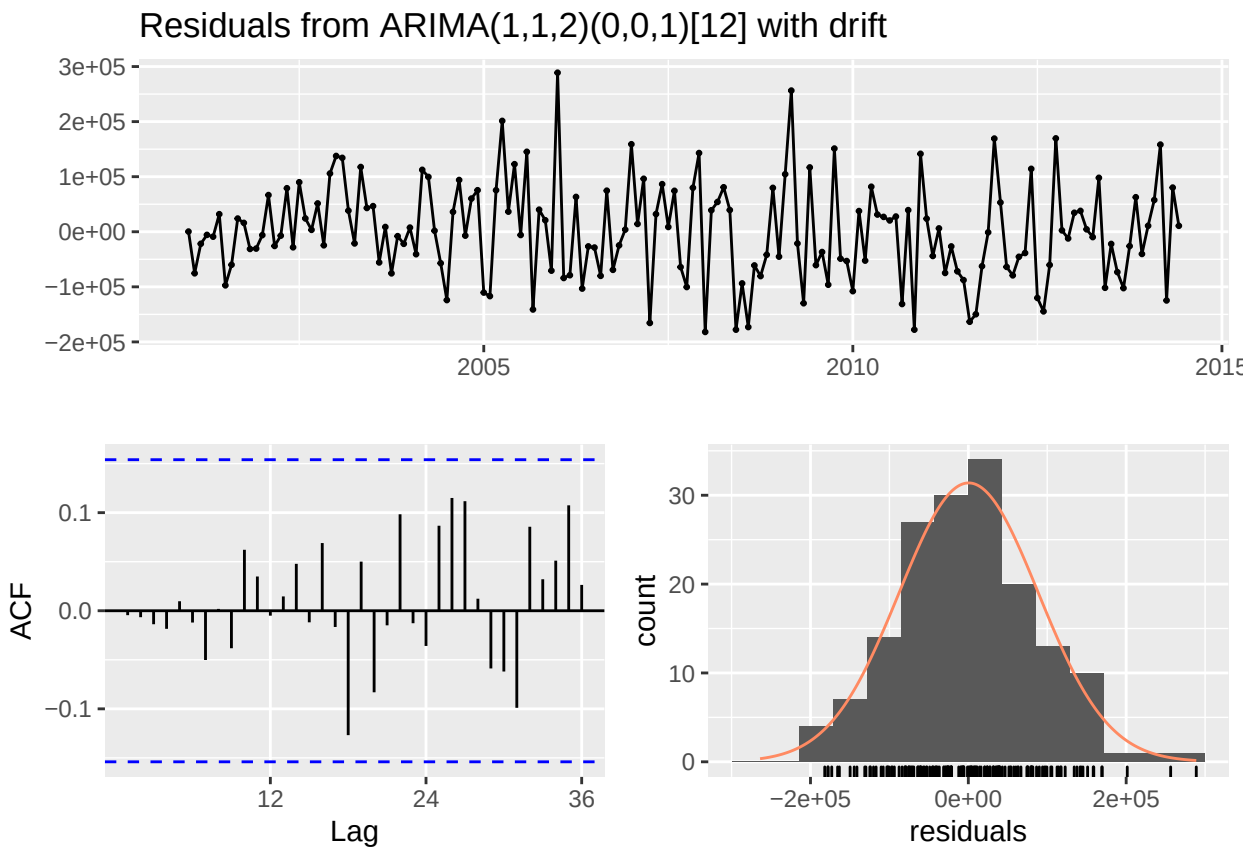
Los residuales parecen seguir una distribución normal. Los datos se distribuyen de forma dependiente puesto que el valor p es menor a 0.05. En efecto, se puede rechazar la hipótesis nula, es decir, el modelo **no** es aceptable para predecir.

Análisis de residuales del modelo automático

```
qqnorm(fitAutoArima$residuals)
qqline(fitAutoArima$residuals)
```




```
checkresiduals(fitAutoArima)
```

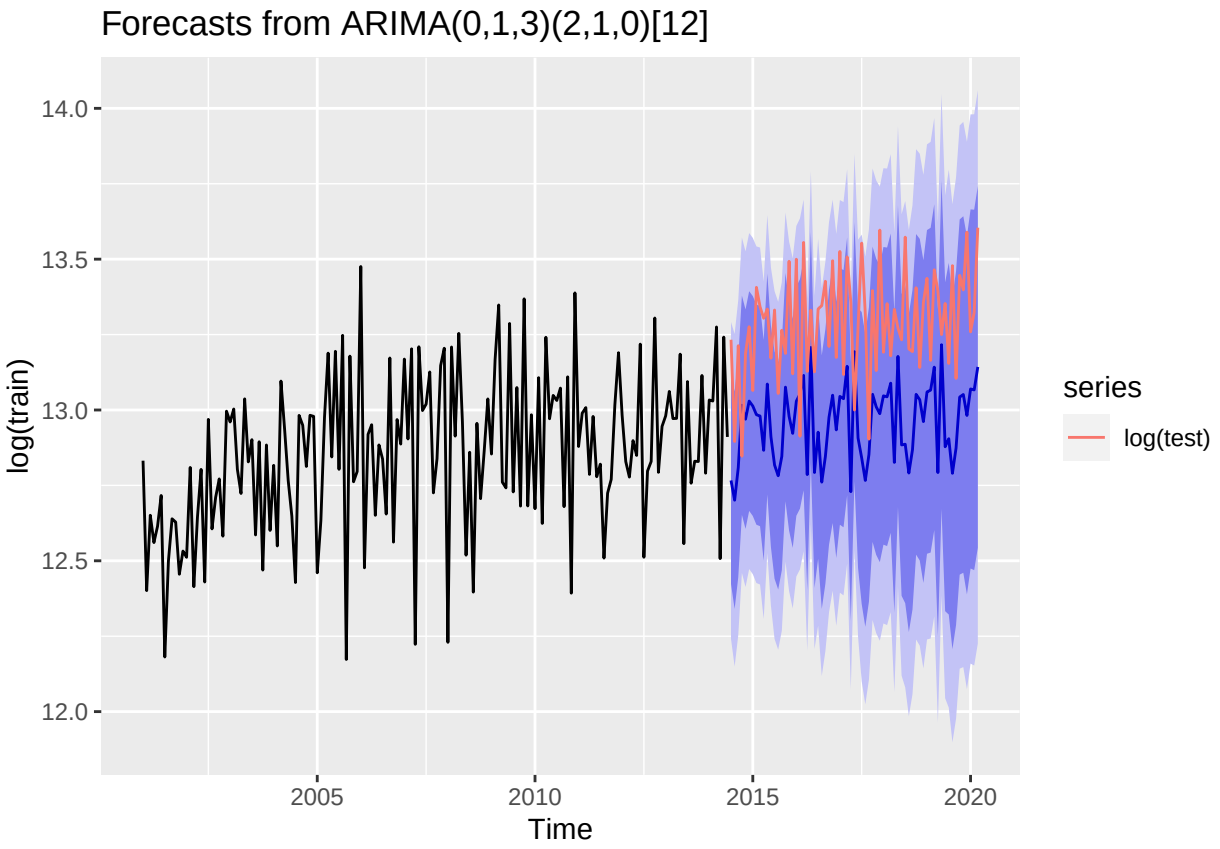



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,2)(0,0,1)[12] with drift
## Q* = 9.994286, df = 19, p-value = 0.953087
##
## Model df: 5.   Total lags used: 24
```

El test Ljung-Box dice que los datos se distribuyen de forma independiente puesto que el valor-p es mayor a 0.05. Puede rechazarse la hipótesis nula, el modelo es aceptable para predecir.

Predicción con modelo generado

```
fitArima %>%
  forecast(h) %>%
  autoplot() + autolayer(log(test))
```

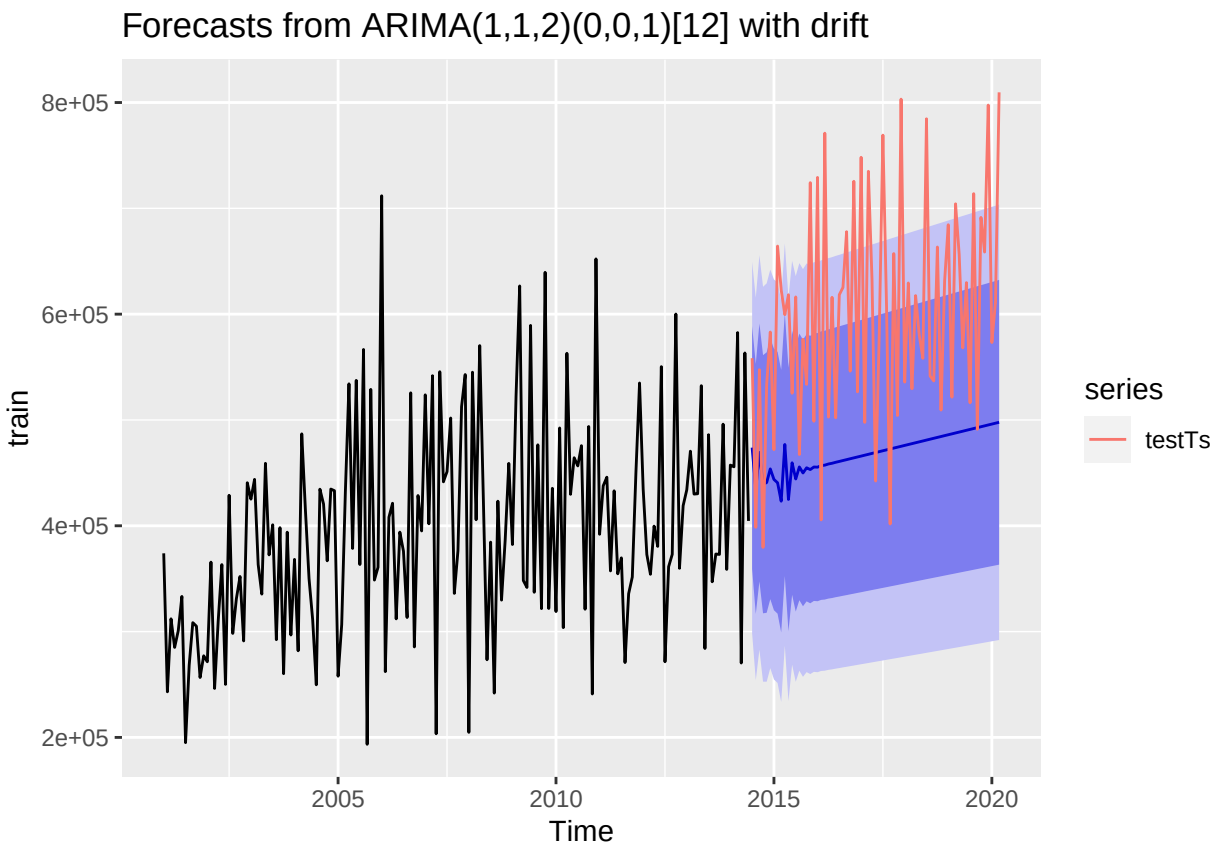


Interpretación: El modelo no llega a tener una predicción totalmente acertada, sin embargo, una gran parte del valor predecido entra en el área segura de predicción. Esto quiere decir que el modelo que hemos generado sí tiene la capacidad de predecir adecuadamente el comportamiento del gas superior, aunque no de una forma totalmente viable. Parte de esto puede deberse a que el modelo estuvo expuesto a un crecimiento constante por parte de los precios de la gasolina, sin embargo, a partir del 2015 los precios se mantuvieron casi al mismo nivel que los 8 años anteriores y eso causó una predicción no acertada.

Predicción con el modelo automatico

```
testTs<-ts(test,start = c(2014,7),end = c(2020,3),frequency = 12)

fitAutoArima %>%
  forecast(h) %>%
  autoplot() + autolayer(testTs)
```



El modelo generado por automáticamente no logró predecir adecuadamente. Muchas de sus predicciones no entran en el área segura aunque sí parece tener un comportamiento más constante.

Conclusión del modelo GAS SUPERIOR:

- Se logró obtener un modelo generado que predijera de forma considerablemente certera los datos reales.
- El modelo generado automáticamente por R no obtuvo resultados tan eficientes como en análisis de los comportamientos y la obtención de parámetros manuales.

Construcción del modelo Prophet

Preparamos los datos como los necesita este algoritmo:

```
df<-data.frame(ds=as.Date(as.yearmon(time(train))),y=as.matrix(train) )
testdf<-data.frame(ds=as.Date(as.yearmon(time(test))),y=as.matrix(test) )
```

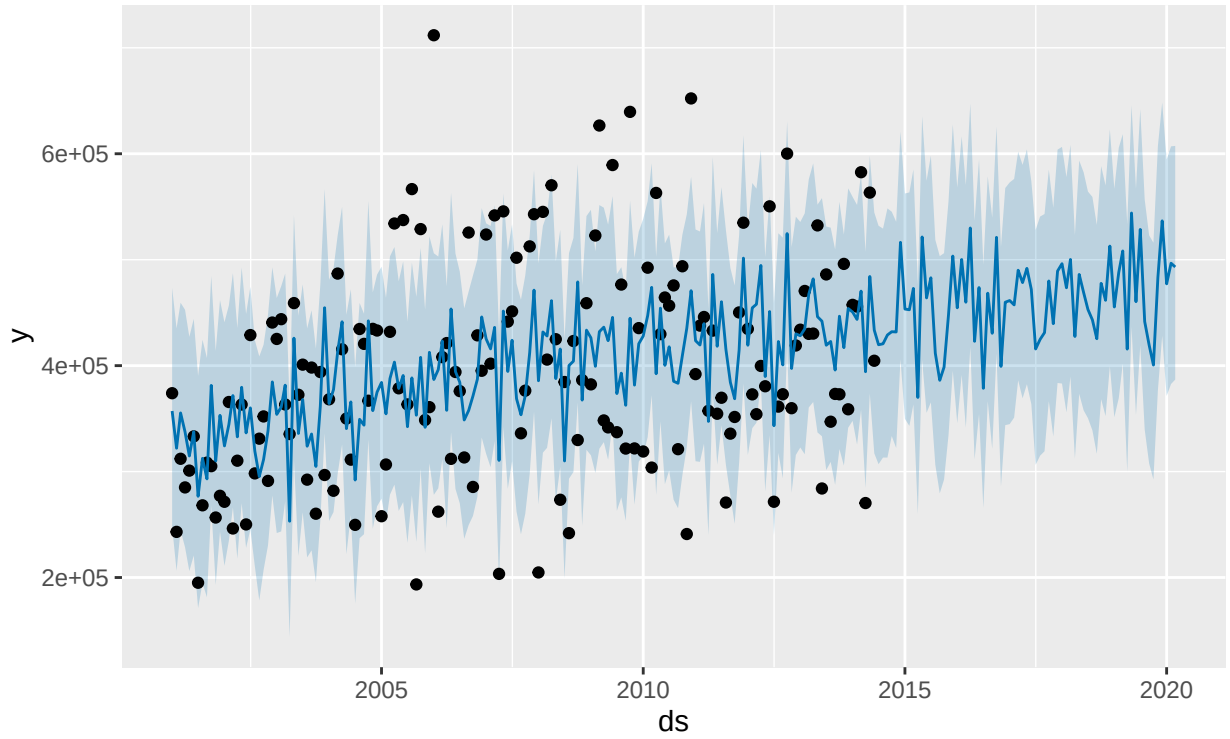
A continuación, elaboramos el modelo:

```
fitProphet<-prophet(df,yearly.seasonality = T,weekly.seasonality = T)
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

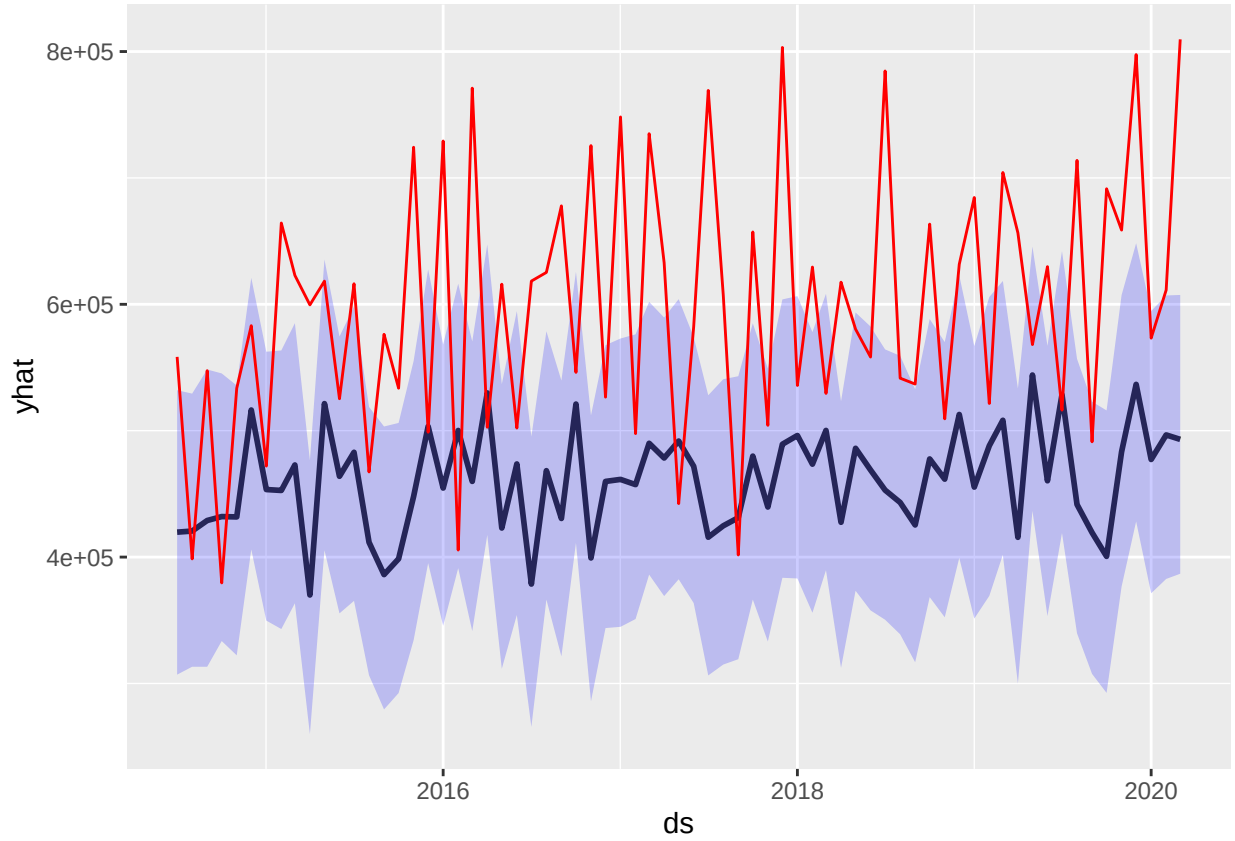
Usaremos el modelo para predecir la misma cantidad de meses que el modelo ARIMA:

```
future <- make_future_dataframe(fitProphet,periods = h,freq = "month", include_history = T)
p <- predict(fitProphet,future)
p<-p[,c("ds","yhat","yhat_lower","yhat_upper")]
plot(fitProphet,p)
```




```
pred<-tail(p,h)
pred$y<-testdf$y

ggplot(pred, aes(x=ds, y=yhat)) +
   geom_line(size=1, alpha=0.8) +
   geom_ribbon(aes(ymin=yhat_lower, ymax=yhat_upper), fill="blue", alpha=0.2) +
   geom_line(data=pred, aes(x=ds, y=y),color="red")
```



El modelo prophet no es tan eficiente como el modelo ARIMA construido por nosotros. Se sale de los intervalos de confianza en varias ocasiones y no alcanza a superar el modelo ARIMA previo.

Modelo de Diesel

```
#Leer
dataSet = read.csv("datos.csv",stringsAsFactors = FALSE, na.strings = TRUE, strip.white = TRUE,sep = ",
```

```
#Seleccionar
dataset = dataSet[c(1,2,5,6, 9,10)]
dataset = dataset[-c(46, 96, 146, 196),]
```

```
#Limpiar
dataset$Diesel[dataset$Diesel =="-"] <- 0
dataset$DieselLS[dataset$DieselLS =="-"] <- 0
dataset2 = dataset
```

```
#COnvertir
options(digits=9)
dataset2$Anio = as.numeric(dataset2$Anio)
dataset2$Mes = as.numeric(dataset2$Mes)
dataset2$GasSuperior = as.numeric(gsub(",", "", dataset2$GasSuperior))
dataset2$GasRegular = as.numeric(gsub(",", "", dataset2$GasRegular))
dataset2$Diesel = as.numeric(gsub(",", "", dataset2$Diesel))
```

```r
dataset2$DieselLS = as.numeric(gsub(",", "", dataset2$DieselLS))
#Unir Diesel
dataset2$Diesel = dataset2$Diesel + dataset2$DieselLS
dataset2 = dataset2[-6]
dataSet = dataset2
View(dataSet)
```

Serie de tiempo Diesel

Primero creamos la serie de tiempo para los datos del Diesel

```r
View(dataSet)
diesel <- ts(dataSet[dataSet$Diesel!="Diesel" & dataSet$Diesel!="-","Diesel"], start=c(2001, 1), end=c(
```

```r
class(diesel)
```

```
## [1] "ts"
```

```r
View(diesel)
```

Construcción del modelo ARIMA

Identificación

A continuación se exploran las características de la serie:

Frecuencia de la serie:

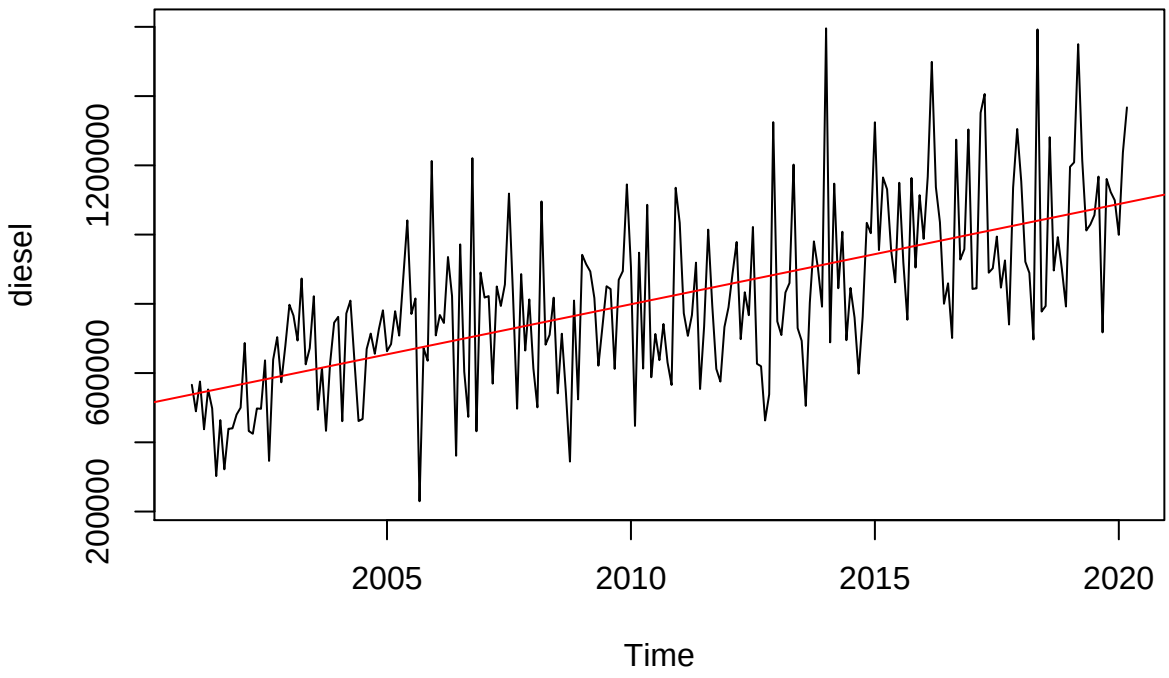
```r
frequency(diesel)
```

```
## [1] 12
```

Frecuencia de la serie:

Gráfica de la serie de tiempo

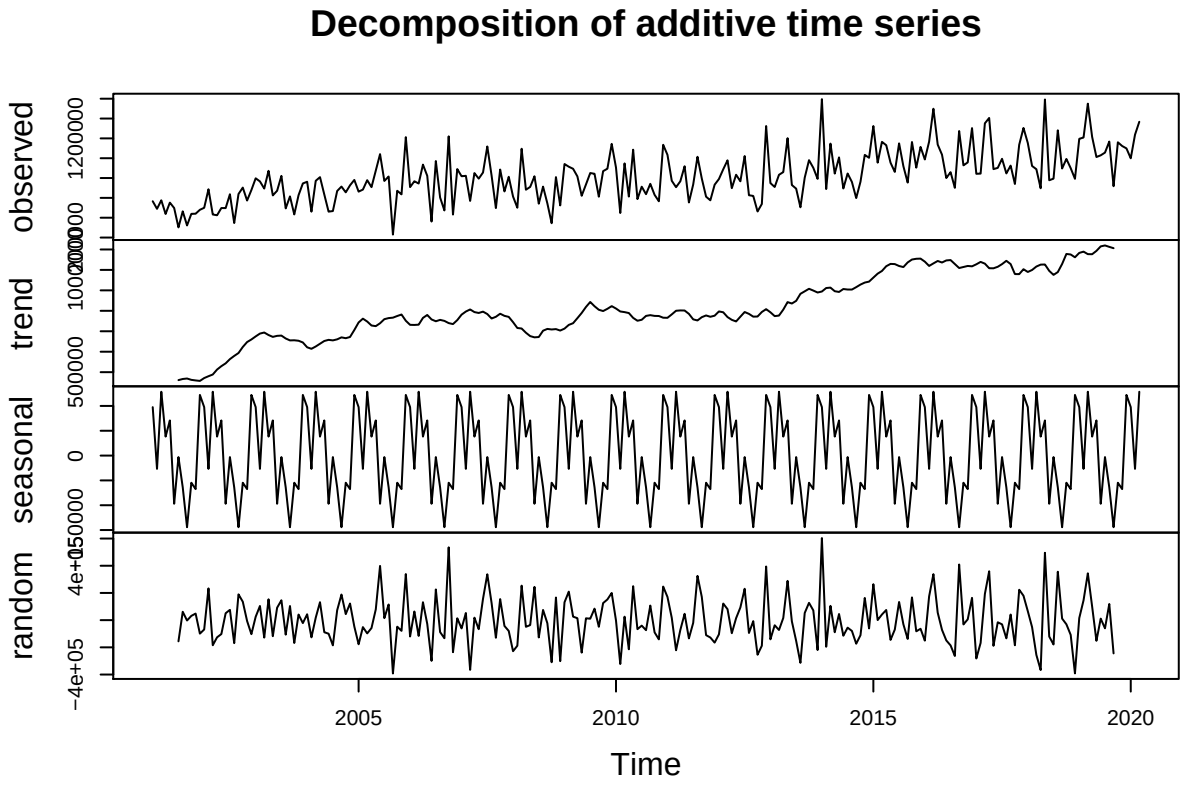
```r
plot(diesel)
abline(reg=lm(diesel~time(diesel)), col=c("red"))
```



Podemos observar que es una serie de tiempo continua con ligera tendencia a crecer por lo que es una serie de tiempo no estacionaria.

Descomposición de la serie

```
plot(decompose(diesel))
```



Podemos observar una serie con tendencia a aumentar, que no es estacionaria en varianza y también tiene estacionalidad.

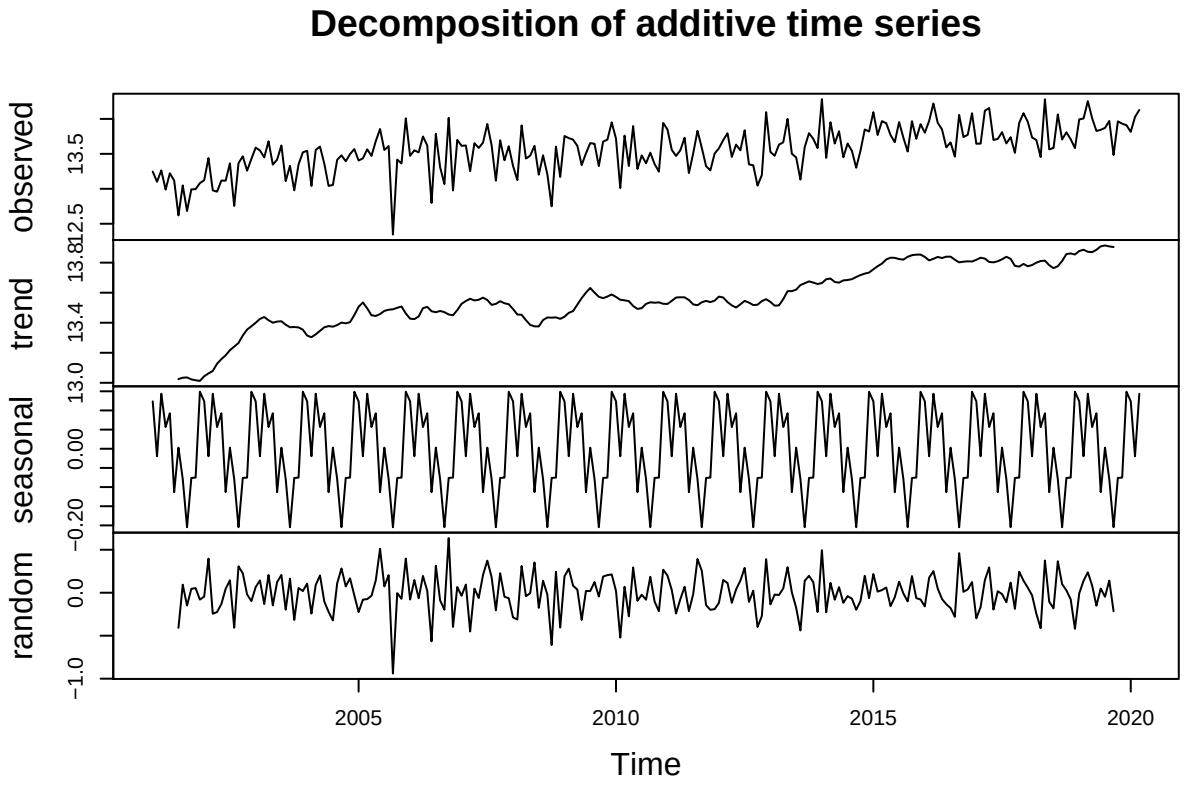
A continuación se separa la serie de tiempo en entrenamiento y prueba

```
train <- head(diesel, round(length(diesel) * 0.7))
h <- length(diesel) - length(train)
test <- tail(diesel, h)
```

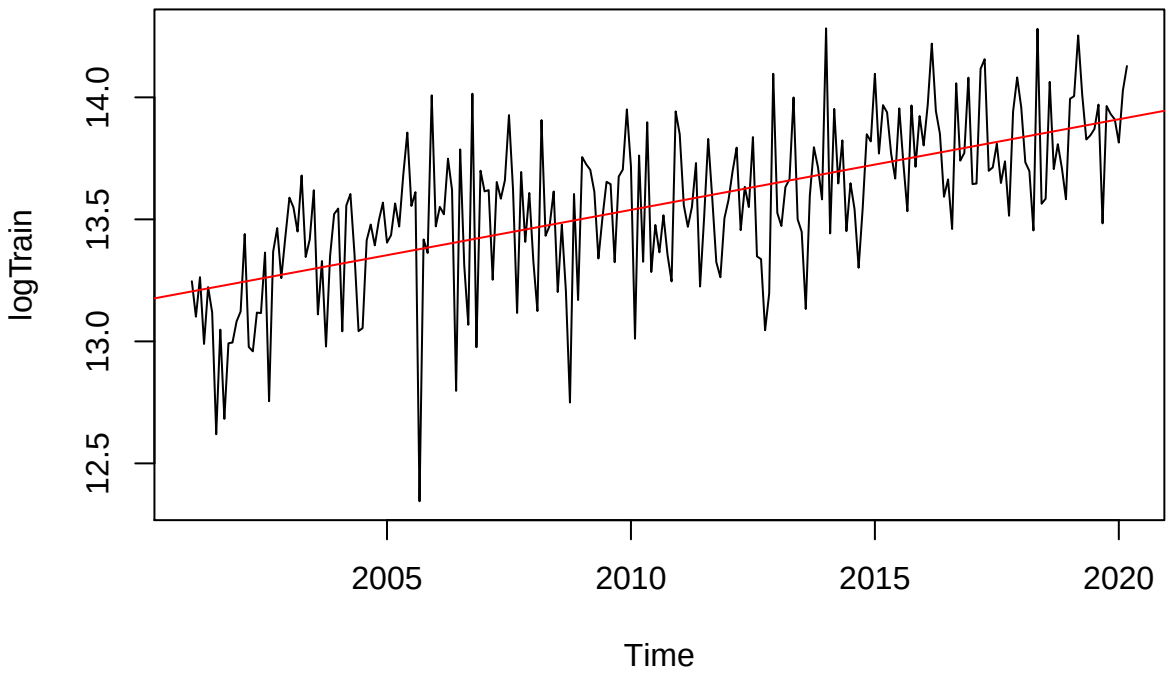
Estimar los parámetros del modelo

A continuación aplicaremos una transformación logaritmica para hacer que la serie sea constante en varianza.

```
logTrain <- log(diesel)
plot(decompose(logTrain))
```




```
plot(logTrain)
abline(reg=lm(logTrain~time(logTrain)), col=c("red"))
```



Podemos notar que se logro hacer un poco más constante la varianza de la serie. A continuación verificaremos que es estacionaria en media. Si tiene raices unitarias podemos decir que no es estacionaria en media.

```
adfTest(train)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -1.1157
##   P VALUE:
##     0.2588
##
## Description:
##  Mon Aug 03 21:43:30 2020 by user: Oscar
```

```
unitrootTest(train)
```

```
##
## Title:
```

```
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     DF: -1.1157
##   P VALUE:
##     t: 0.2396
##     n: 0.4514
##
## Description:
##  Mon Aug 03 21:43:30 2020 by user: Oscar
```

Como podemos notar, en ambas pruebas el valor de p es mayor a 0.05 por lo que no se puede rechazar la hipótesis nula de ausencia de raíces unitarias. Entonces por ende no es estacionaria en media.

A coninuación hacemos lo mismo pero con una diferenciación:

```
adfTest(diff(train))
```

```
## Warning in adfTest(diff(train)): p-value smaller than printed p-value
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -15.0038
##   P VALUE:
##     0.01
##
## Description:
##  Mon Aug 03 21:43:30 2020 by user: Oscar
```

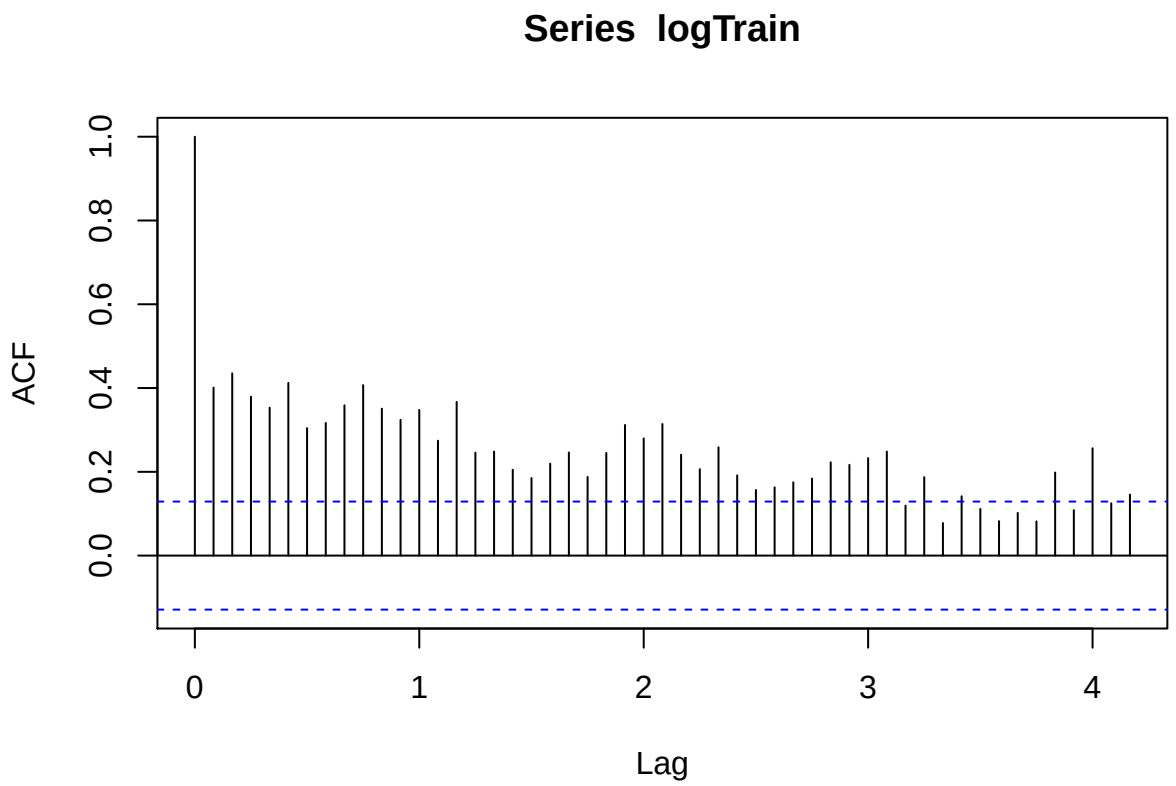
```
unitrootTest(diff(train))
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     DF: -15.0038
##   P VALUE:
##     t: < 2.2e-16
##     n: 0.006087
##
## Description:
##  Mon Aug 03 21:43:30 2020 by user: Oscar
```

Como podemos observar en esta ocasión el valor de p está por debajo de 0.05 por lo que ahora si se puede descartar la hipótesis nula de que existen raíces unitarias. Podemos notar entonces que solo es necesaria una diferenciación (d=1).

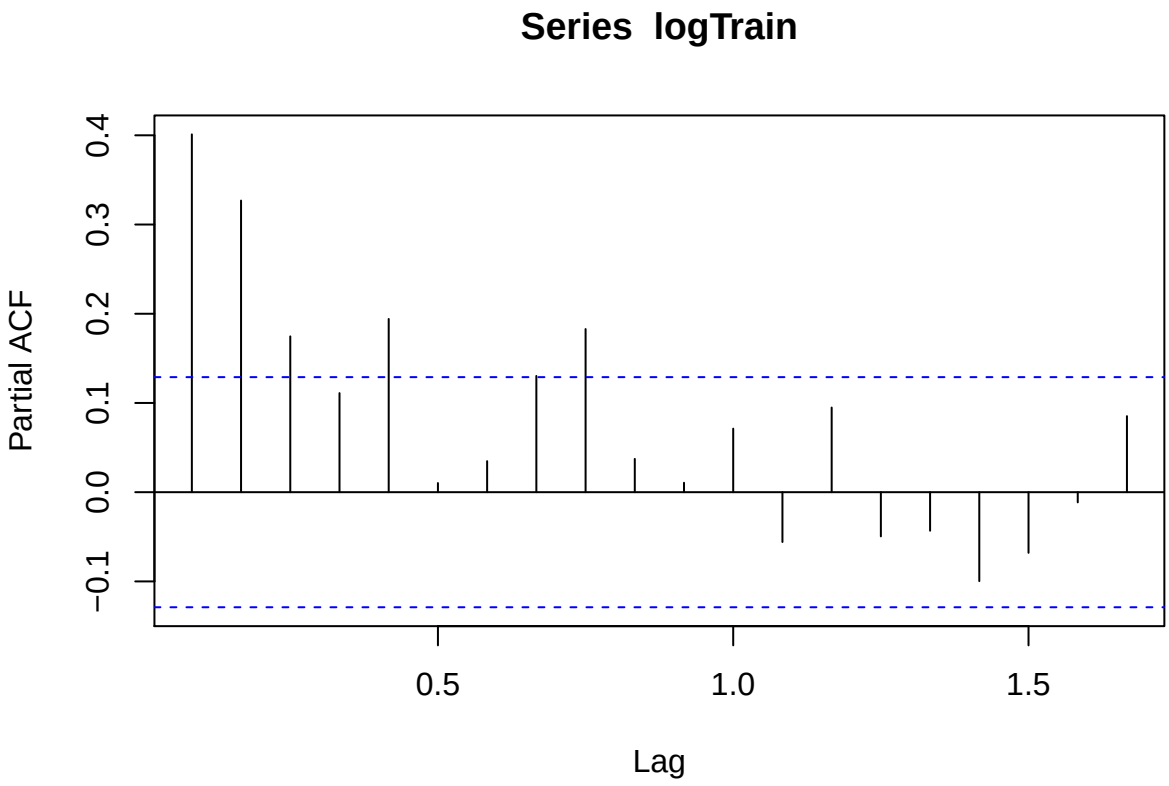
El siguiente paso es intentar identificar los parámetros p y q usando los gráficos de autocorrelación y autocorrelación parcial.

```
acf(logTrain,50)
```



En el gráfico de la función podemos notar que se anula después del tercer retardo por lo que se puede sugerir q=3.

```
pacf(logTrain,20)
```



En el gráfico de la función podemos notar que se anula después del retardo 0 por lo que podríamos sugerir un coeficiente p=0.

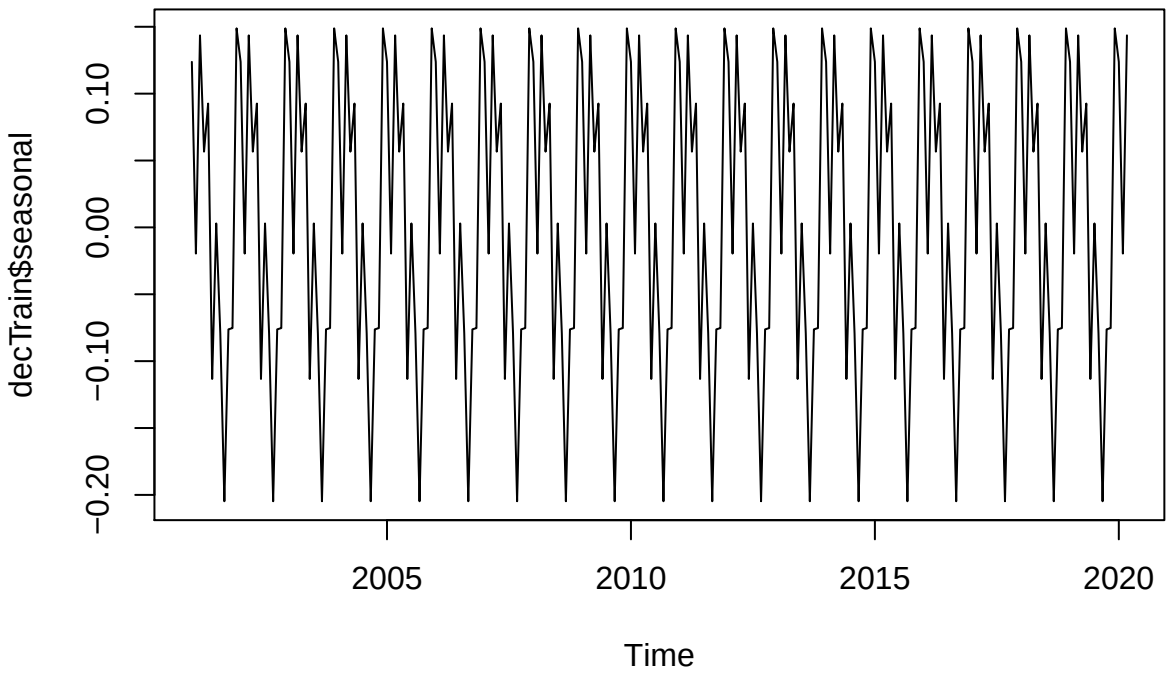
Podemos sugerir un modelo con los siguientes parámetros:

- p=0.
- q=3.
- d=1.

O en otras palabras ARIMA(p,d,q).

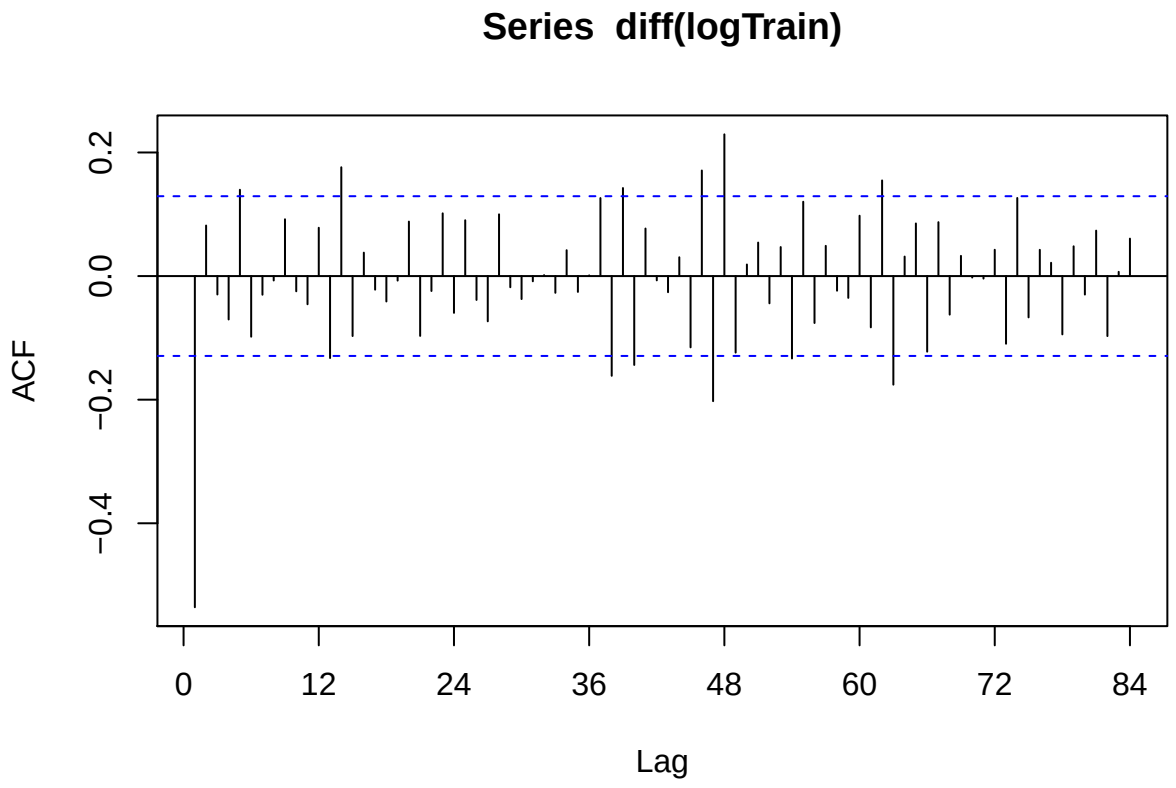
Volvamos a ver la descomposición de la serie paran determinar si hay estacionalidad

```
decTrain <- decompose(logTrain)
plot(decTrain$seasonal)
```



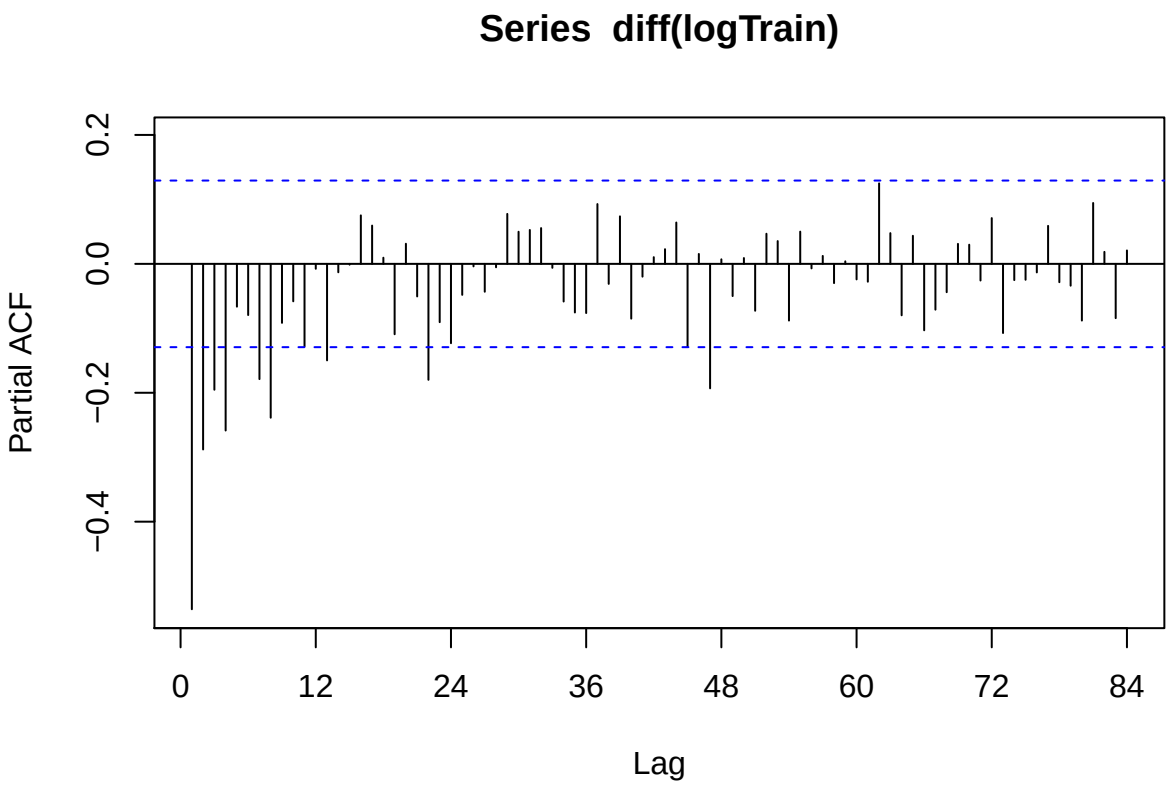
Podemos notar que si existe estacionalidad en la serie. A continuación, se buscarán los parámetros estacionales usando las funciones de autocorrelación y autocorrelación parcial.

```
Acf(diff(logTrain),84)
```



Podemos notar que en casi todos los períodos hay dos decaimientos estacionales por lo que podemos sugerir P=2.

```
Pacf(diff(logTrain),84)
```



Podemos darnos cuenta que los pico significativos están al inicio de la gráfica por lo que podemos sugerir D=1. Finalmente Q=0.

Con los parámetros determinados generamos un modelo:

```
fitArima <- arima(logTrain,order=c(0,1,3),seasonal = c(2,1,0))
```

R tiene la capacidad de generar un modelo automáticamente, entonces también lo tomaremos en cuenta:

```
fitAutoArima <- auto.arima(train)
```

Significación de los coeficientes:

A continuación, analizaremos que tan significativos son los coeficientes de cada modelo

```
coeftest(fitArima)
```

```
##
## z test of coefficients:
##
##         Estimate  Std. Error   z value   Pr(>|z|)
## ma1  -1.09772587  0.07069463 -15.52771 < 2.22e-16 ***
## ma2   0.18113128  0.09822645   1.84402   0.065181 .
## ma3  -0.01392184  0.07177497  -0.19397   0.846203
## sar1 -0.72662466  0.06862060 -10.58902 < 2.22e-16 ***
```

```
## sar2 -0.32607818  0.06657797  -4.89769  9.697e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Podemos notar que todos los coeficientes son significativo ya que todos son menores a 0.05.

```
coeftest(fitAutoArima)
```

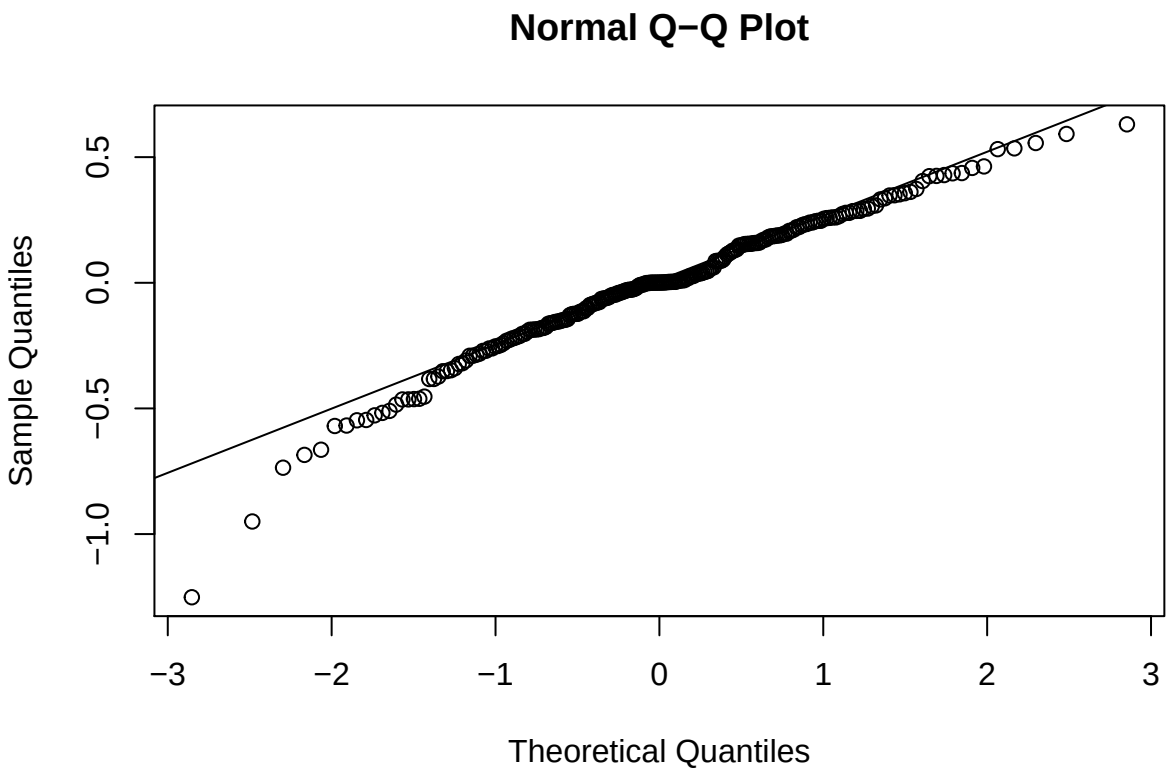
```
##
## z test of coefficients:
##
##            Estimate    Std. Error  z value   Pr(>|z|)
## ar1   -1.551293e+00  3.216116e-01 -4.82350 1.4106e-06 ***
## ar2   -8.306880e-01  2.731574e-01 -3.04106  0.0023575 **
## ma1    5.230484e-01  3.671542e-01  1.42460  0.1542725
## ma2   -6.423266e-01  1.513212e-01 -4.24479 2.1880e-05 ***
## ma3   -7.246198e-01  2.918135e-01 -2.48316  0.0130222 *
## sar1   2.321362e-03  9.416122e-02  0.02465  0.9803317
## drift  2.315523e+03  8.495563e+02  2.72557  0.0064191 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

En este caso podemos notar no todos los coeficientes son significativos.

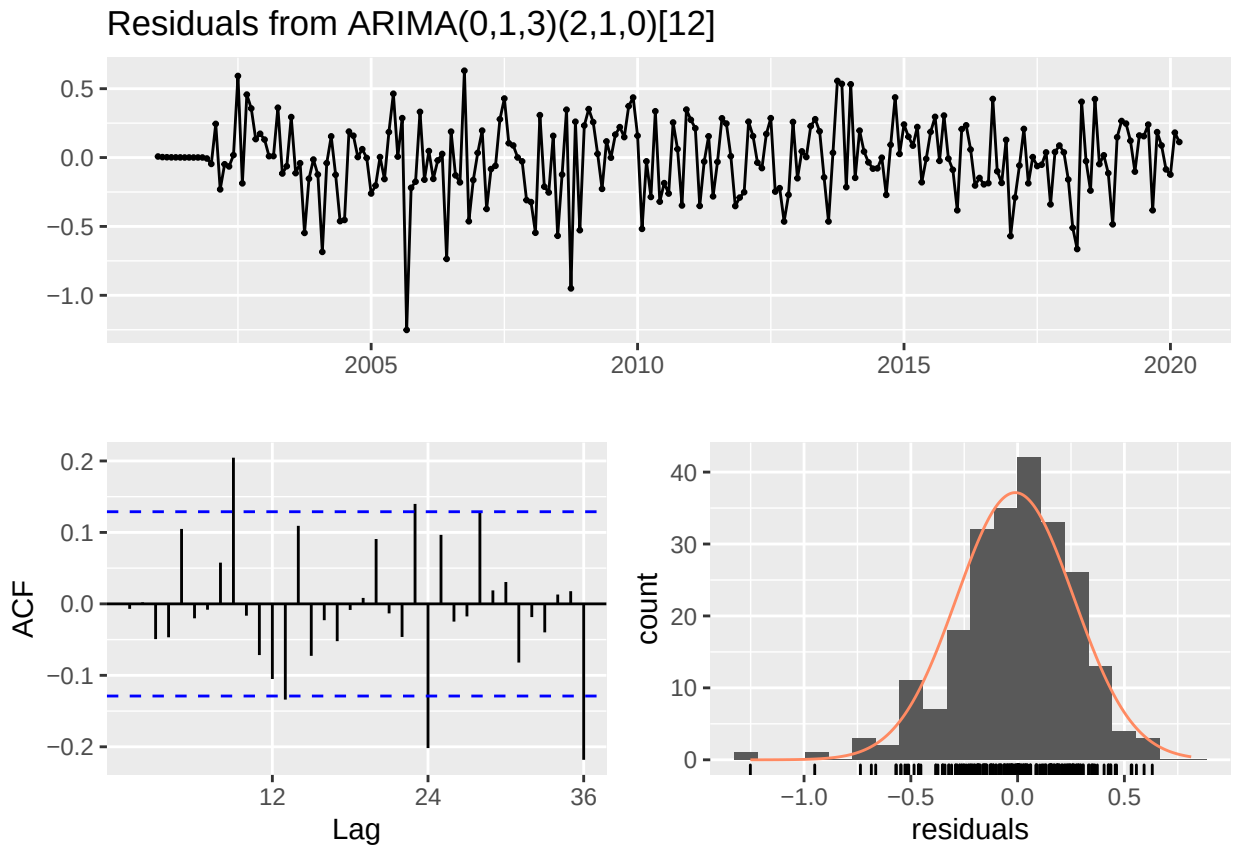
Análisis de Residuales

A continuación, se analizarán los residuales de ambos modelos. Estos deben estar distribuidos normalmente y se deben de parecer al ruido blanco.

```
qqnorm(fitArima$residuals)
qqline(fitArima$residuals)
```




```
checkresiduals(fitArima)
```

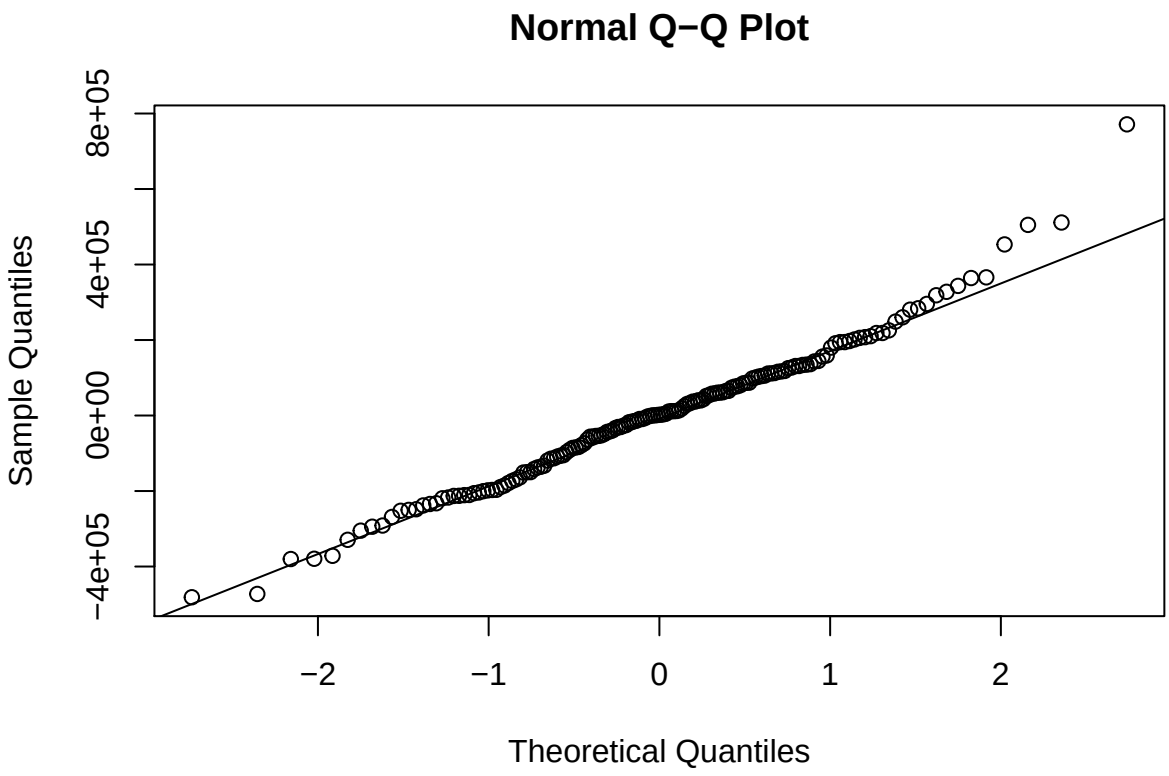



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,3)(2,1,0)[12]
## Q* = 46.73241, df = 19, p-value = 0.000390028
##
## Model df: 5.   Total lags used: 24
```

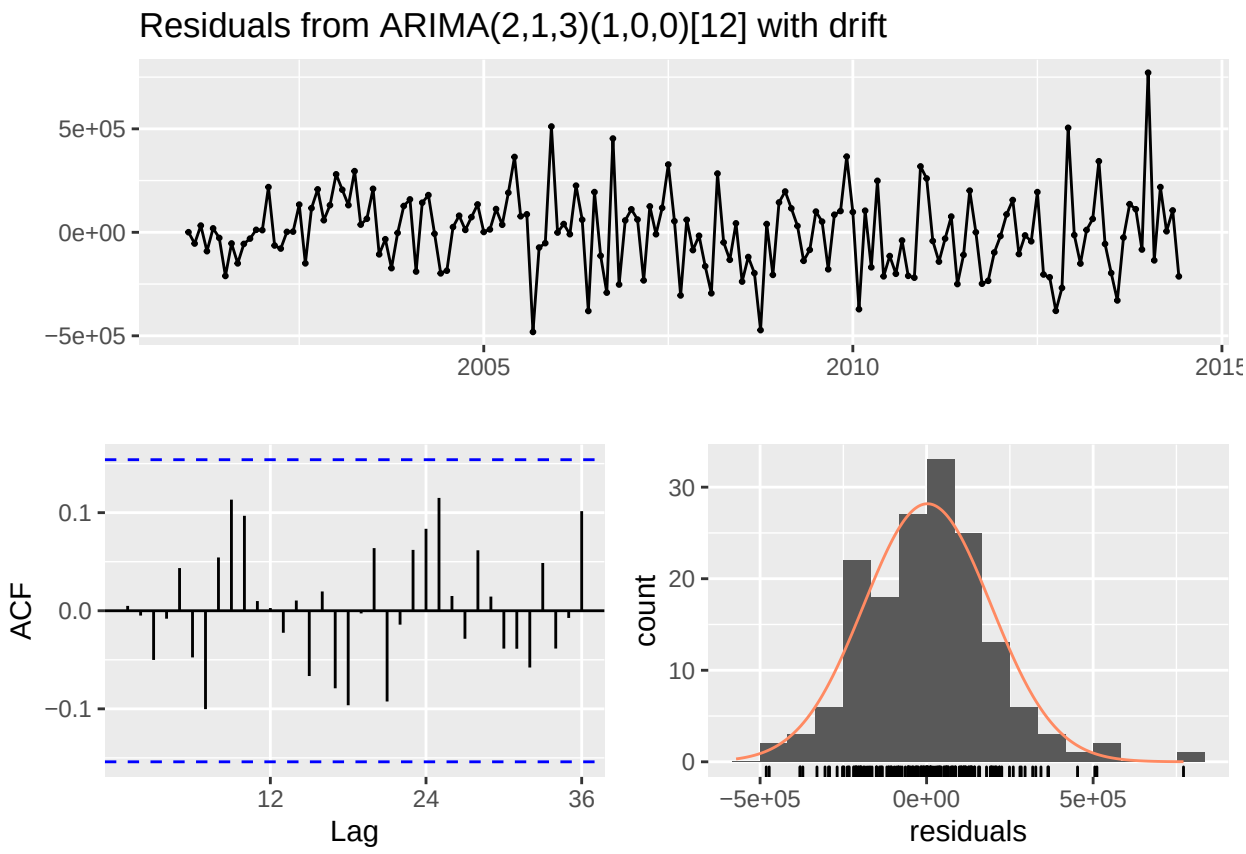
Podemos notar que los datos tienen una distribución normal. Sin embargo, según el test de Ljung-Box los datos se distribuyen de forma dependiente puesto que el p-value es menor a 0.05 y se puede rechazar la hipótesis nula. Esto quiere decir que el modelo no es aceptable para predecir.

Analicemos también el modelo generado automáticamente por R:

```
qqnorm(fitAutoArima$residuals)
qqline(fitAutoArima$residuals)
```




```
checkresiduals(fitAutoArima)
```



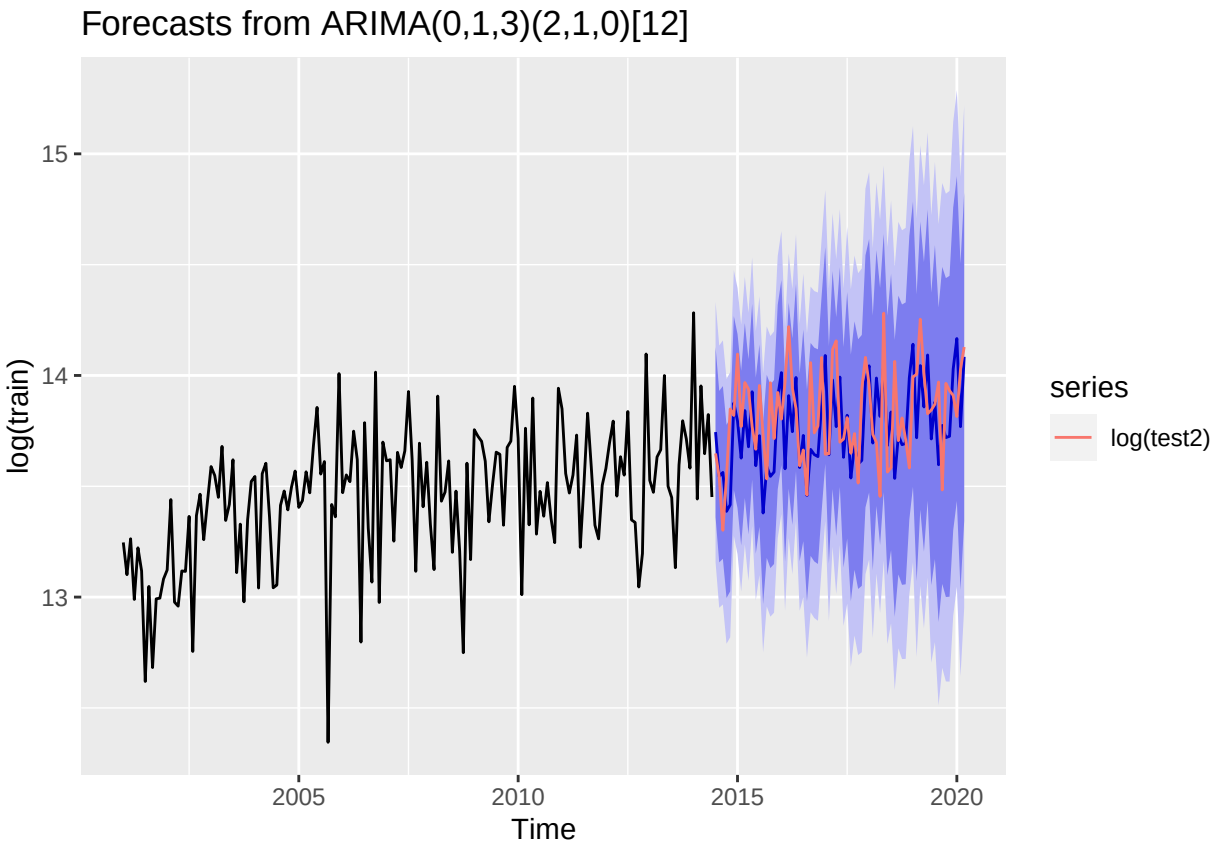

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,3)(1,0,0)[12] with drift
## Q* = 15.59534, df = 17, p-value = 0.552681
##
## Model df: 7.   Total lags used: 24
```

En este caso podemos notar que el test Ljung-Box dice que los datos se distribuyen de forma independiente puesto que el p-value es mayor a 0.05, por lo tanto, puede rechazarse la hipótesis nula y el modelo es aceptable para predecir.

Predicción con el modelo generado

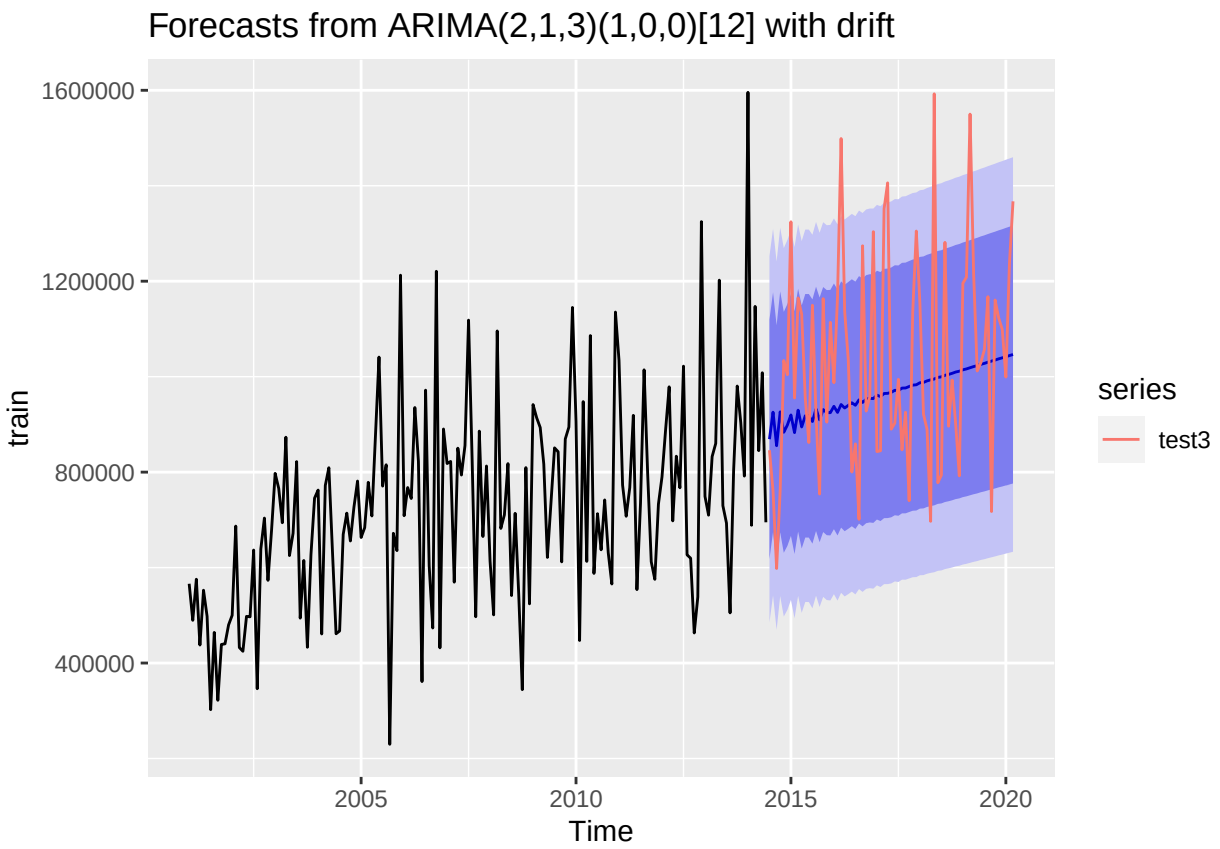
Pasaremos a predecir con la función forecast la misma cantidad de meses que hay en test y compararemos con los valores reales.

```
fitArima <- arima(log(train),c(0,1,3),seasonal =list( order=c(2,1,0),period=12))
test2<-ts(test,start = c(2014,7),end = c(2020,3),frequency = 12)
fitArima %>%
  forecast(h) %>%
  autoplot() + autolayer(log(test2))
```



Podemos notar que nuestro modelo es bueno prediciendo, ya que este se parece bastante a los datos reales, así como también está dentro del intervalo de confianza.

```
fitAutoArima <- auto.arima(train)
test3<-ts(test,start = c(2014,7),end = c(2020,3),frequency = 12)
fitAutoArima %>%
  forecast(h) %>%
  autoplot() + autolayer(test3)
```



Podemos notar que el modelo realizado automáticamente por R no predice muy bien, ya que no se parece en nada a los datos reales. Entonces podemos decir que nosotros hemos construido un mejor modelo al que genera automáticamente R.

Construcción del modelo Prophet

Preparamos los datos como los necesita este algoritmo:

```
df<-data.frame(ds=as.Date(as.yearmon(time(train))),y=as.matrix(train) )
testdf<-data.frame(ds=as.Date(as.yearmon(time(test))),y=as.matrix(test) )
```

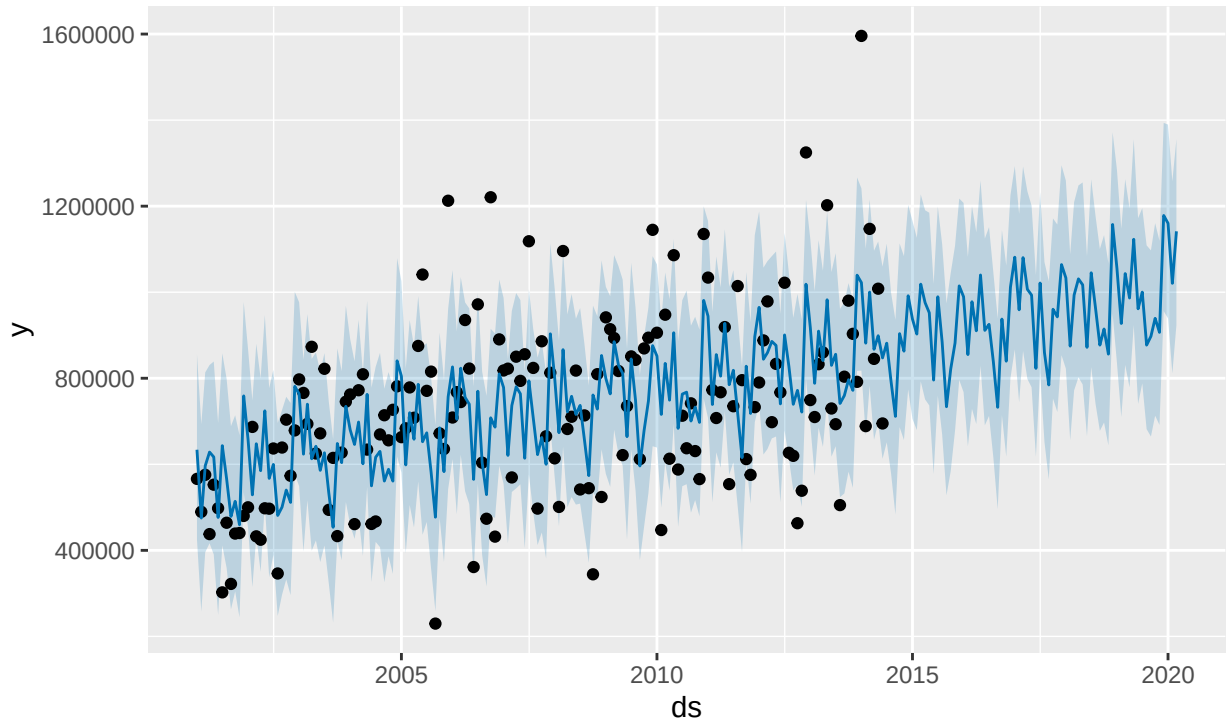
A continuación, elaboramos el modelo:

```
fitProphet<-prophet(df,yearly.seasonality = T,weekly.seasonality = T)
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

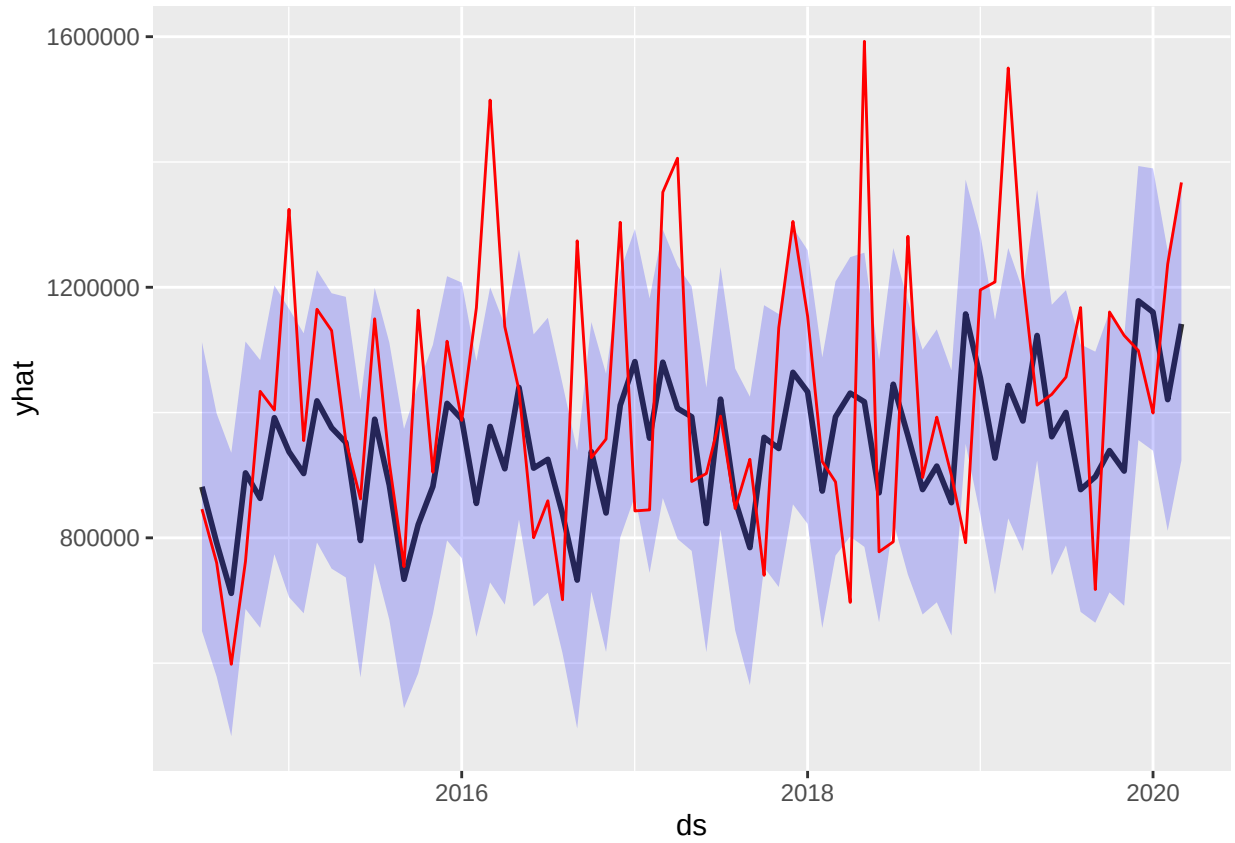
Usaremos el modelo para predecir la misma cantidad de meses que el modelo ARIMA:

```
future <- make_future_dataframe(fitProphet,periods = h,freq = "month", include_history = T)
p <- predict(fitProphet,future)
p<-p[,c("ds","yhat","yhat_lower","yhat_upper")]
plot(fitProphet,p)
```




```
pred<-tail(p,h)
pred$y<-testdf$y

ggplot(pred, aes(x=ds, y=yhat)) +
   geom_line(size=1, alpha=0.8) +
   geom_ribbon(aes(ymin=yhat_lower, ymax=yhat_upper), fill="blue", alpha=0.2) +
   geom_line(data=pred, aes(x=ds, y=y),color="red")
```



Podemos notar que el modelo Prophet es considerablemente bueno aunque se sale un poco de los intervalos de confianza, sin embargo si hacemos una comparación con el modelo que realizamos ARIMA , nuestro predicción se parece más a los datos reales por lo que podemos decir que el modelo ARIMA es mejor.