Tarea 3

Modelación ARIMA

Modelos de series de Tiempo y Supervivencia

Profesor: Naranjo Albarrán Lizbeth

Adjuntos: Reyes González Belén

Rivas Godoy Yadira

Integrantes: Cuéllar Chávez Eduardo de Jesús

García Tapia Jesús Eduardo

Miranda Meraz Areli Gissell

Ramírez Maciel José Antonio

Saldaña Morales Ricardo

Grupo: 9249

Fecha: 10/NOV/2021

Analizar los datos Quarterly U.S. new plant/equip. expenditures 64 76 billions de la liberia tsdl de R.

1. Análisis descriptivo.

Grafique los datos, describa lo que observe (varianza constante o no constante, descomposición clásica, tendencia, ciclos estacionales, periodicidad de los ciclos).

Primero carguemos la librería tsdl, ya que los datos necesarios se encuentran en dicha librería; así como otras necesarias para esta tarea.

```
library(tsdl);library(ggplot2);library(itsmr);library(forecast);library(TSA);library(lmtest)
library(timeSeries);library(timeSeries);library(astsa);library(dygraphs);
library(tseries);library(forecast);library(nortest);library(dplyr);library(imputeTS)
```

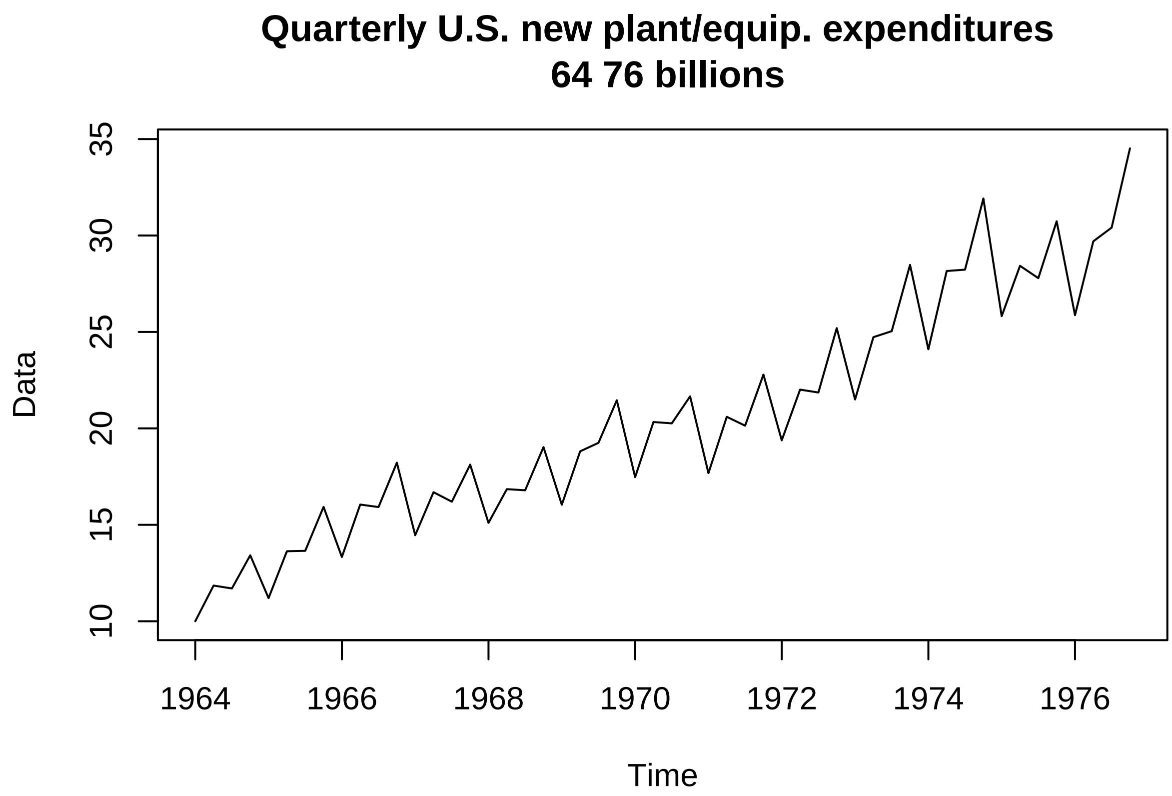
Cargamos la base de datos, nos aseguramos de que es la deseada.

```
Data <- tsdl[[12]]
attributes(Data)
```

```
## $tsp
## [1] 1964.00 1976.75    4.00
##
## $class
## [1] "ts"
##
## $source
## [1] "Abraham & Ledolter (1983)"
##
## $description
## [1] "Quarterly U.S. new plant/equip. expenditures -64 - -76 billions"
##
## $subject
## [1] "Microeconomic"
```

Ahora que noa aseguramos de que es la base que queriamos, procedemos a graficar:

```
plot(Data, main = "Quarterly U.S. new plant/equip. expenditures \n 64 76 billions")
```



Varianza Al menos de manera gráfica, la intuición nos dice que no hay varianza constante, pero probémoslo con un test de homocedasticidad:

```
#Los pasamos a series de tiempo
Serie<-ts(data=Data,start=c(1964,01),end=c(1976,4),frequency=4)
tiempo<-seq(1964+0/4, 1976+3/4, by = 1/4)
bptest(Serie~tiempo)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  Serie ~ tiempo
## BP = 5.3713, df = 1, p-value = 0.02047
```

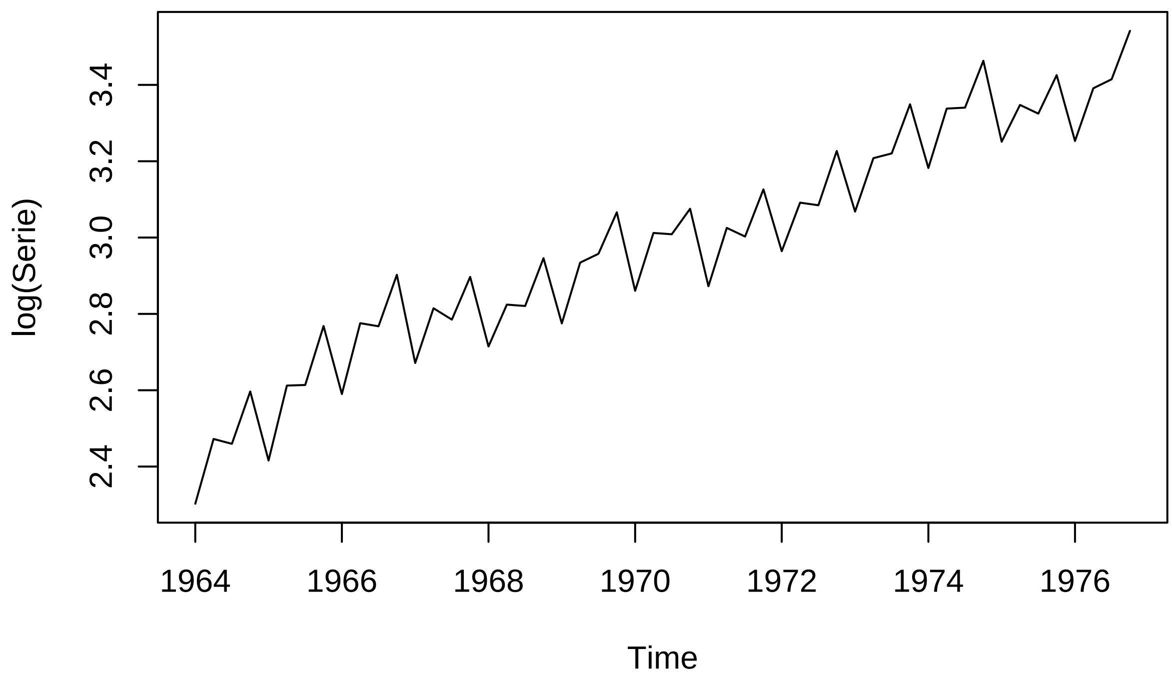
Efectivamente, no pasa el test de homocedasticidad.

Veamos qué pasa si aplicamos la transformación logaritmo:

```
bptest(log(Serie)~tiempo)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  log(Serie) ~ tiempo
## BP = 1.965, df = 1, p-value = 0.161
```

```
plot(log(Serie))
```

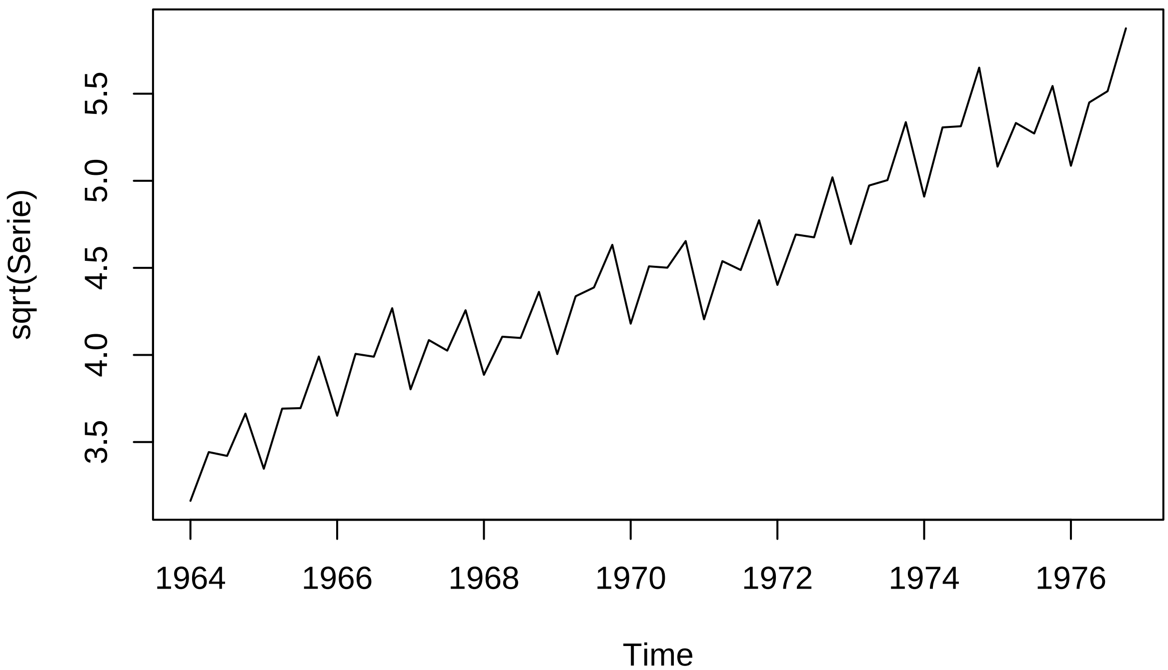


Tampoco ayudó, aunque mejoró un poco. Intentemos con la raíz cuadrada

```
bptest(sqrt(Serie)~tiempo)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  sqrt(Serie) ~ tiempo
## BP = 0.8651, df = 1, p-value = 0.3523
```

```
plot(sqrt(Serie))
```



¡Logramos estabilizarla!

Veamos si es estacionaria.

```
adf.test(sqrt(Serie))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  sqrt(Serie)
## Dickey-Fuller = -1.415, Lag order = 3, p-value = 0.8097
## alternative hypothesis: stationary
```

```
kpss.test(sqrt(Serie))
```

```
## Warning in kpss.test(sqrt(Serie)): p-value smaller than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  sqrt(Serie)
## KPSS Level = 1.389, Truncation lag parameter = 3, p-value = 0.01
```

Tendencia

Podemos observar una tendencia creciente que se presenta de manera lineal (al parecer), de manera general a lo largo de la serie

Ciclos estacionales

Los ciclos están bastante marcados , y tiene sentido puesto que son los datos de gastos de una empresa en maquinaria por trimestre, y en ese contexto es lógico que se presenten ciclos: Generalmente decae del último trimestre del año anterior al primero del año siguiente, para después crecer en el segundo semestre, en el tercero se mantiene casi al mismo nivel que el segundo, pero en el último aumenta; y es un comportamiento que se repite año con año

Periodicidad de los ciclos

Como comentamos en el apartado anterior, parece (al menos de manera gráfica) que se tienen ciclos anuales.

Descomposición clásica

Descomponeremos la serie por medio de filtros lineales:

Estabilización de la varianza

Aplicamos la transformación raíz cuadrada

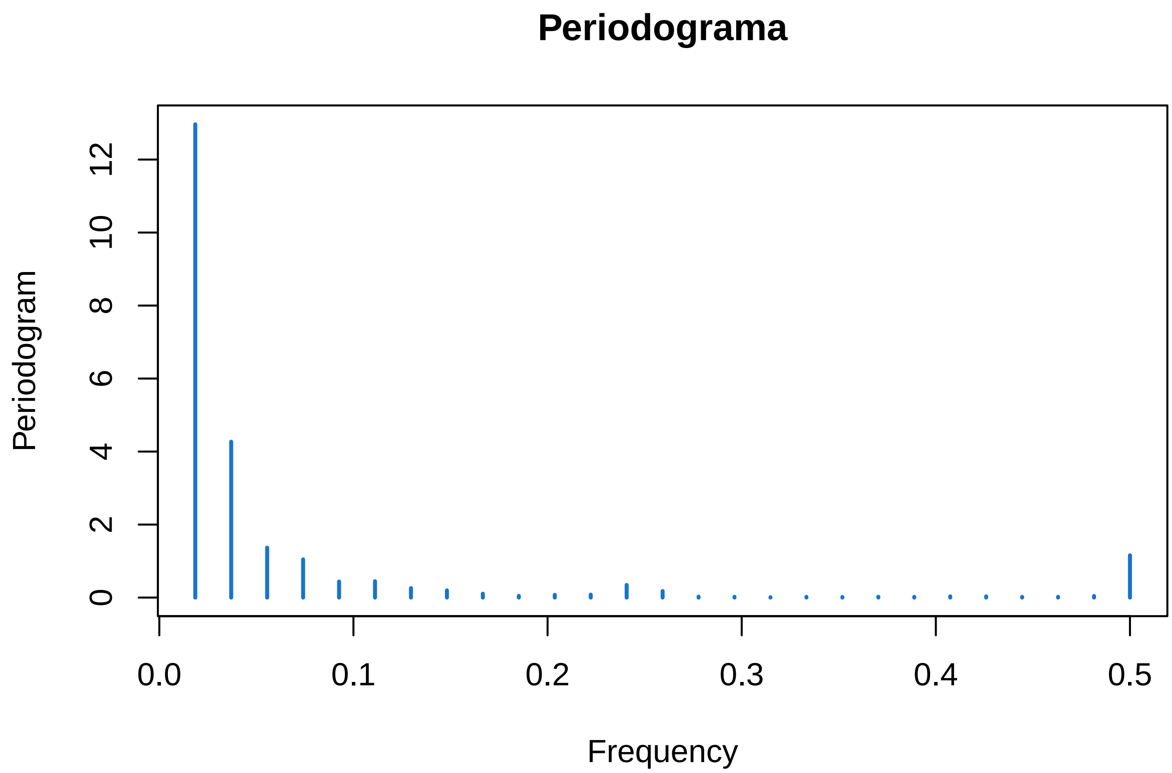
```
Serie_sq<-sqrt(Serie)
bptest(Serie_sq~tiempo)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  Serie_sq ~ tiempo
## BP = 0.8651, df = 1, p-value = 0.3523
```

Podemos asumir varianza constante

Periodicidad de ciclos

```
#Veamos la tendencia y los ciclos
Xt = Serie_sq
p = periodogram(Xt, main="Periodograma", col=4) # Obtenemos el periodograma
```




```r
names(p)
```

```
##  [1] "freq"      "spec"      "coh"       "phase"     "kernel"    "df"
##  [7] "bandwidth" "n.used"    "orig.n"    "series"    "snames"    "method"
## [13] "taper"     "pad"       "detrend"   "demean"
```

```r
# Ordenamos de mayor a menor las estimaciones del periodograma.
spec = sort(p$spec, decreasing = TRUE)
(spec = spec[1:10]) # Nos quedamos con los 8 coeficientes de mayor frecuencia.
```

```
##  [1] 12.9646281  4.2701894  1.3676068  1.1581310  1.0460373  0.4486585
##  [7]  0.4395386  0.3466661  0.2593484  0.1970454
```

```r
i = match(spec, p$spec) # Buscamos sus indices en el periodograma.
d = p$freq # Vemos las frecuencias del periodograma.
d = d[i] # Nos quedamos con las frecuencias que nos interesan.

cbind(spec,d,i)#
```

```
##            spec          d  i
##  [1,] 12.9646281 0.01851852  1
##  [2,]  4.2701894 0.03703704  2
##  [3,]  1.3676068 0.05555556  3
##  [4,]  1.1581310 0.50000000 27
##  [5,]  1.0460373 0.07407407  4
##  [6,]  0.4486585 0.11111111  6
##  [7,]  0.4395386 0.09259259  5
##  [8,]  0.3466661 0.24074074 13
```

```
##  [9,]  0.2593484 0.12962963  7
## [10,]  0.1970454 0.14814815  8
```

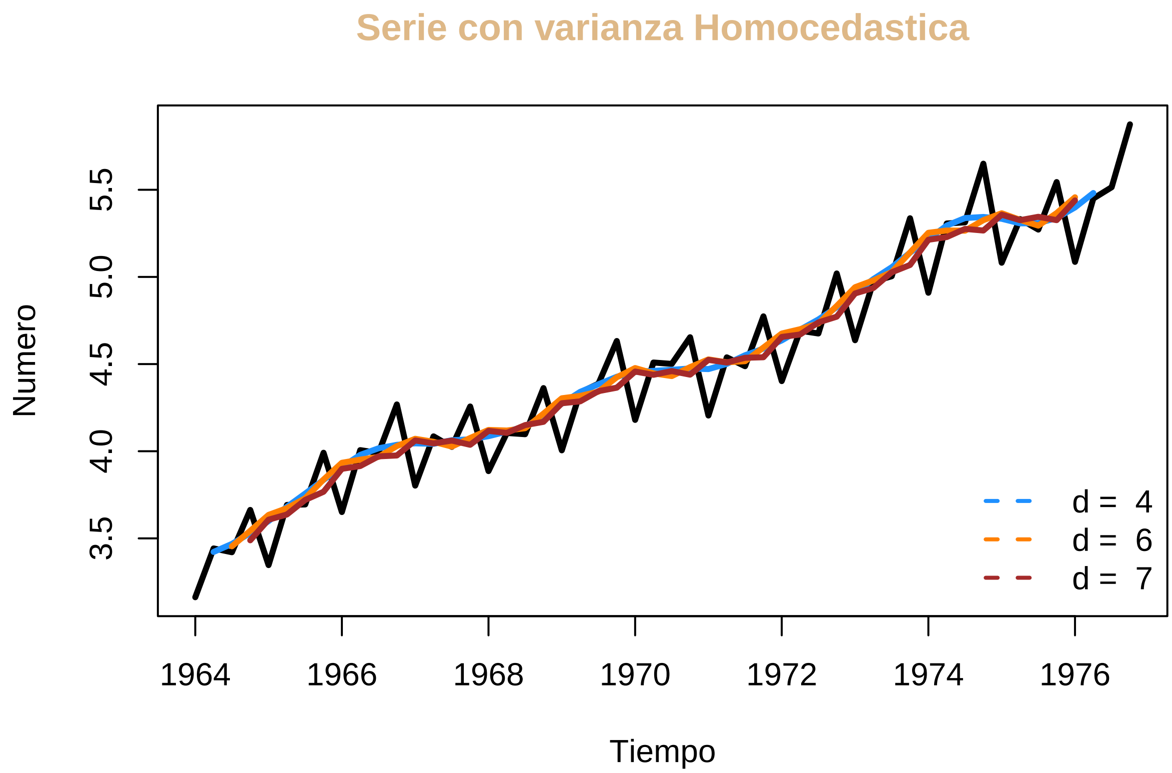
```
d = 1 / d # Obtenemos los parametros para utilizar en promedios moviles.
d = floor(d) #
(d = sort(d))
```

```
##  [1]  2  4  6  7  9 10 13 18 27 54
```

```
# Quitamos los periodos mas grandes
d = d[-length(d)]
d = d[-length(d)]
# Quitamos el más pequeño
d = d[-1]
d #Posibles periodos del ciclo
```

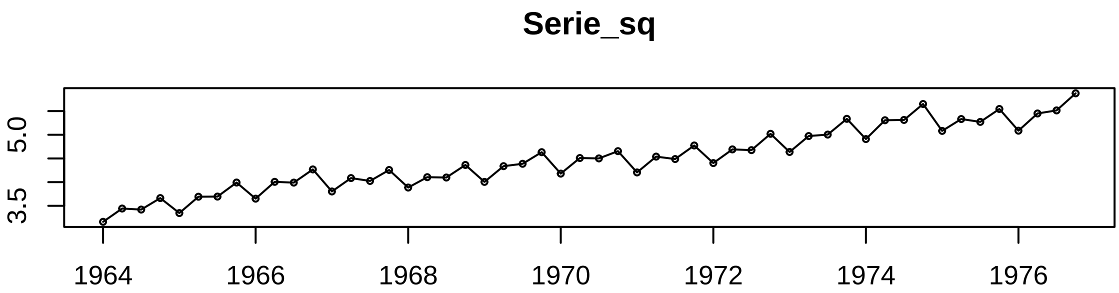
```
## [1]  4  6  7  9 10 13 18
```

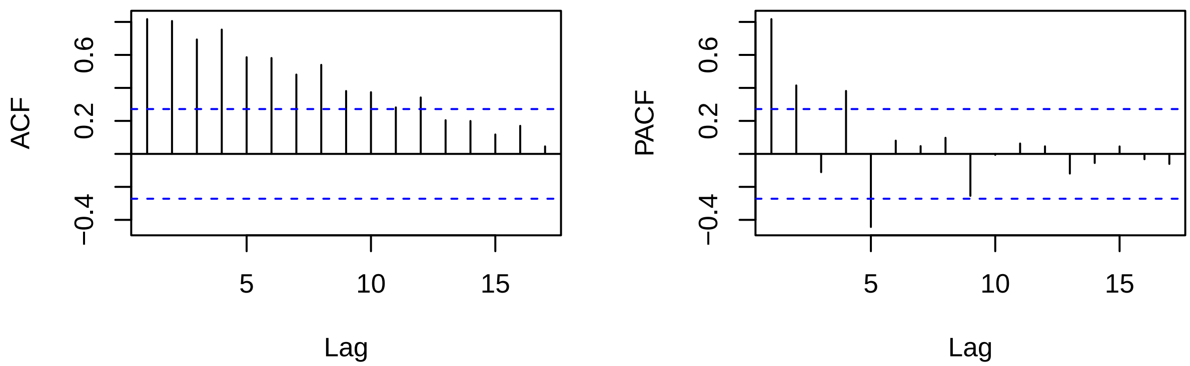
```
#Realizamos la grafica:
col = c("dodgerblue1", "darkorange1", "brown")
plot(Serie_sq, lwd = 3, xlab = "Tiempo", col = "gray0",
     main = "Serie con varianza Homocedastica",
     ylab = "Numero", col.main = "burlywood")
for (i in 1:3) {
  lines(tiempo, stats::filter(Serie_sq, rep(1 / d[i], d[i])), col = col[i],
        lwd = 3)
}
legend("bottomright", col = col, lty = 2, lwd = 2, bty = "n",
       legend = c(paste("d = ", d[1]), paste("d = ", d[2]),
                  paste("d = ", d[3])), cex = 1)
```



Notemos que $d = 2$ parece sobreajustar un poco nuestra gráfica, de hecho, bastante. Sin embargo, con $d = 4$ podemos obtener un buen suavizamiento sin pagar el costo de otros 2 datos al elegir $d = 6$. Veamos el ACF y PACF:

```
tsdisplay(Serie_sq)
```



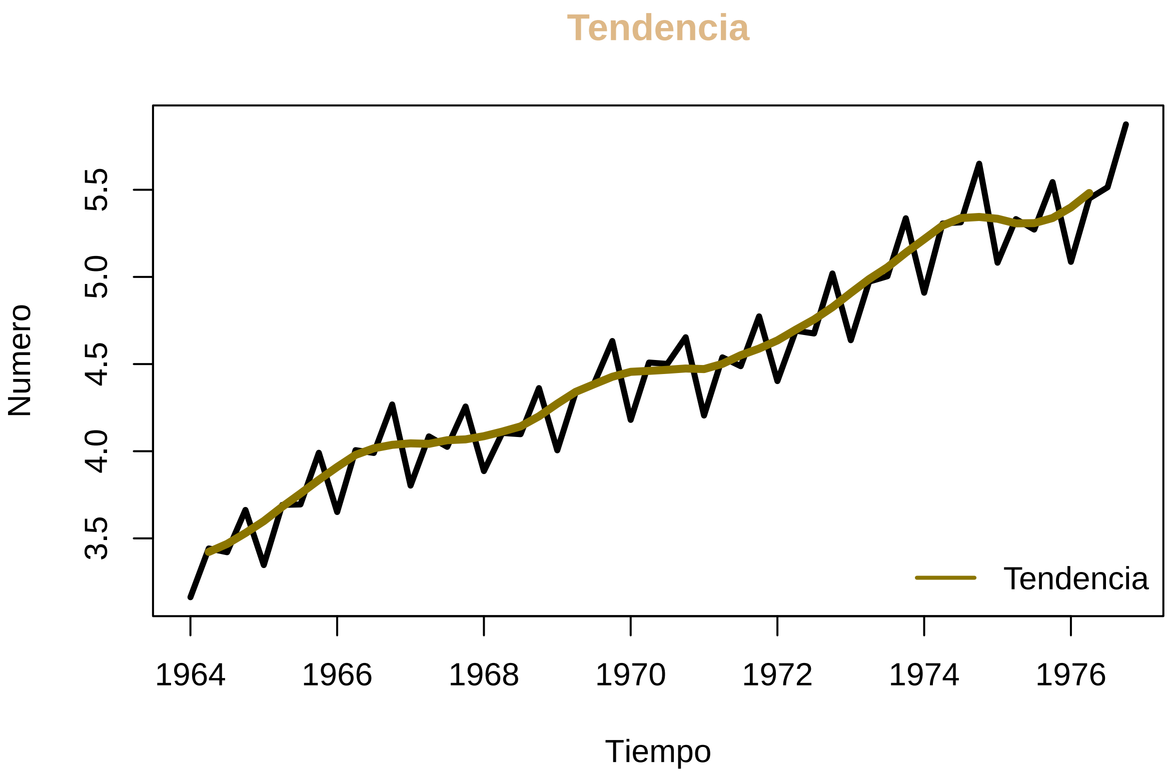


Y, junto con este último resultado, nos parece ideal concluir que el ciclo es $d = 4$

Tendencia

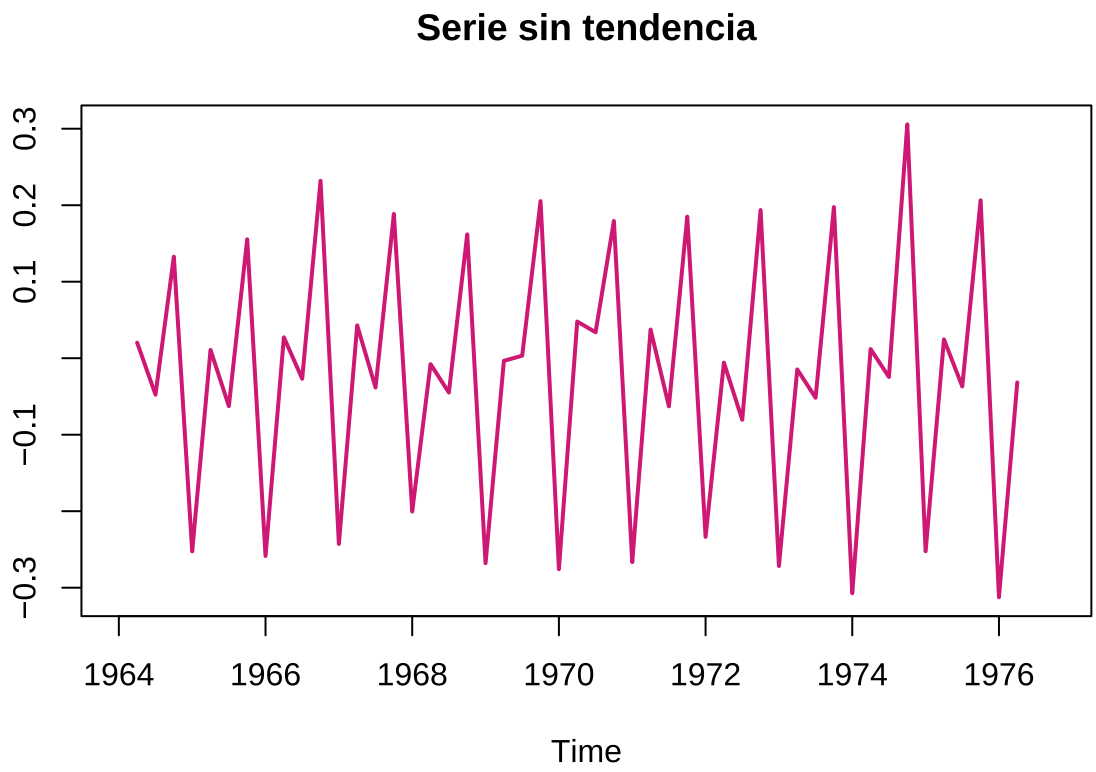
Ahora, aislemos la tendencia:

```
tendencia = stats::filter(Serie_sq, rep(1/4, 4))
plot(Serie_sq, lwd = 3, xlab = "Tiempo", col = "black",
     main = "Tendencia",
     ylab = "Numero", col.main = "burlywood")
lines(tendencia, col = "gold4", lwd = 4)
legend("bottomright", col = "gold4", lty = 1, lwd = 2, bty = "n",
       legend = "Tendencia", cex = 1)
```



Lo que refuerza lo que creíamos: Tiene tebdebcua creciente casi de manera general.

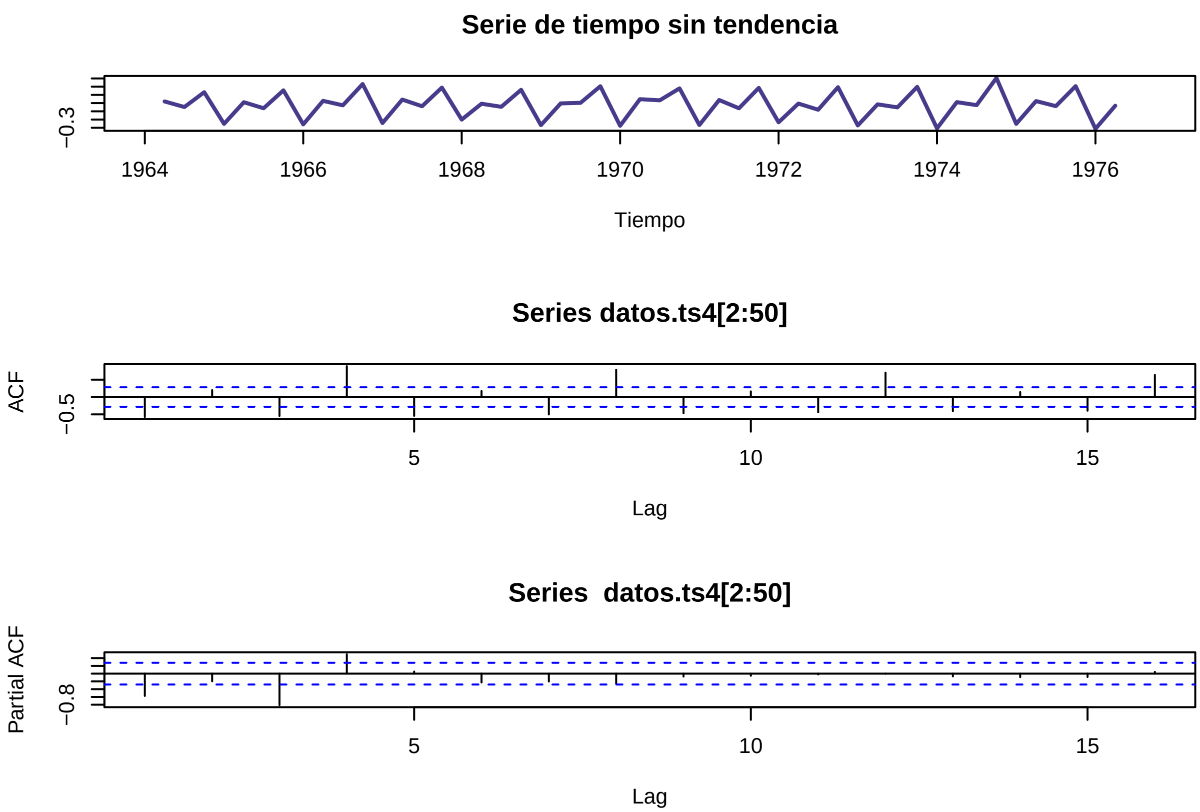
```
# Quitamos la tendencia
# Solo trabajamos con la serie cuya varianza es cte.

datosSinTendencia = Serie_sq - tendencia # Serie sin tendencia
plot(datosSinTendencia, main="Serie sin tendencia", lwd=2, ylab="", col=14)
```



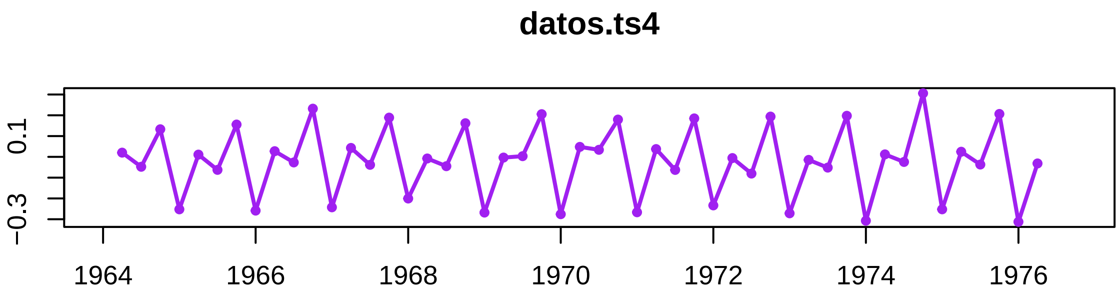

```r
# Convertimos datosSinTendencia en objeto TS, dado que hicimos promedios moviles

inicio=start(Serie_sq)
final=end(Serie_sq)

datos.ts4=ts(datosSinTendencia, frequency = 4, start=inicio,end=final)
which(is.na(datos.ts4)==T)
```

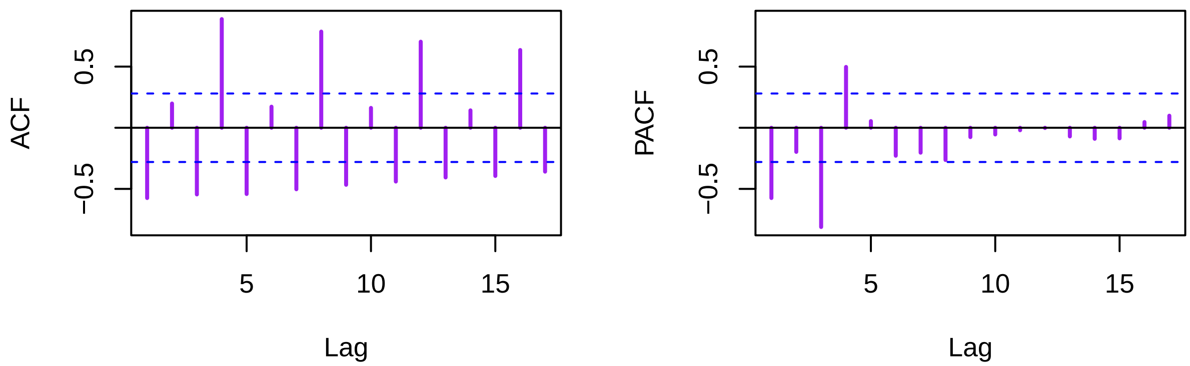
```
## [1]  1 51 52
```

```r
par(mfrow = c(3,1))
plot(datos.ts4, col = "slateblue4", lwd = 2, ylab = " ", type = "l",
     main = "Serie de tiempo sin tendencia", xlab = "Tiempo")

acf(datos.ts4[2:50])
pacf(datos.ts4[2:50])
```




```
par(mfrow = c(1,1))
tsdisplay(datos.ts4, col="purple", lwd=2)
```



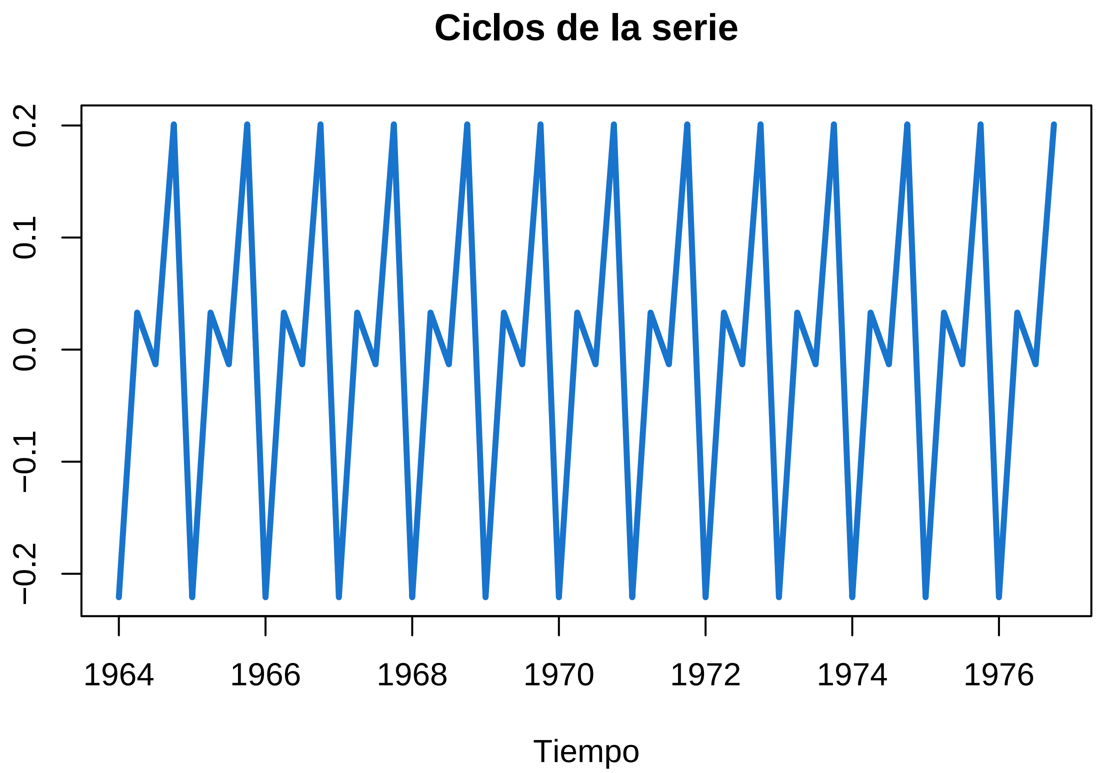


Parece que eliminamos la tendencia, ahora tratemos de verificar los ciclos

Ciclos o parte estacional

Ahora, estimaremos la parte estacional. Tenemos que d = 4. Originalmente contábamos con 52 datos, pero ahora tenemos 48 (por los NA), entonces $\frac{48}{4} = 12$ ciclos.

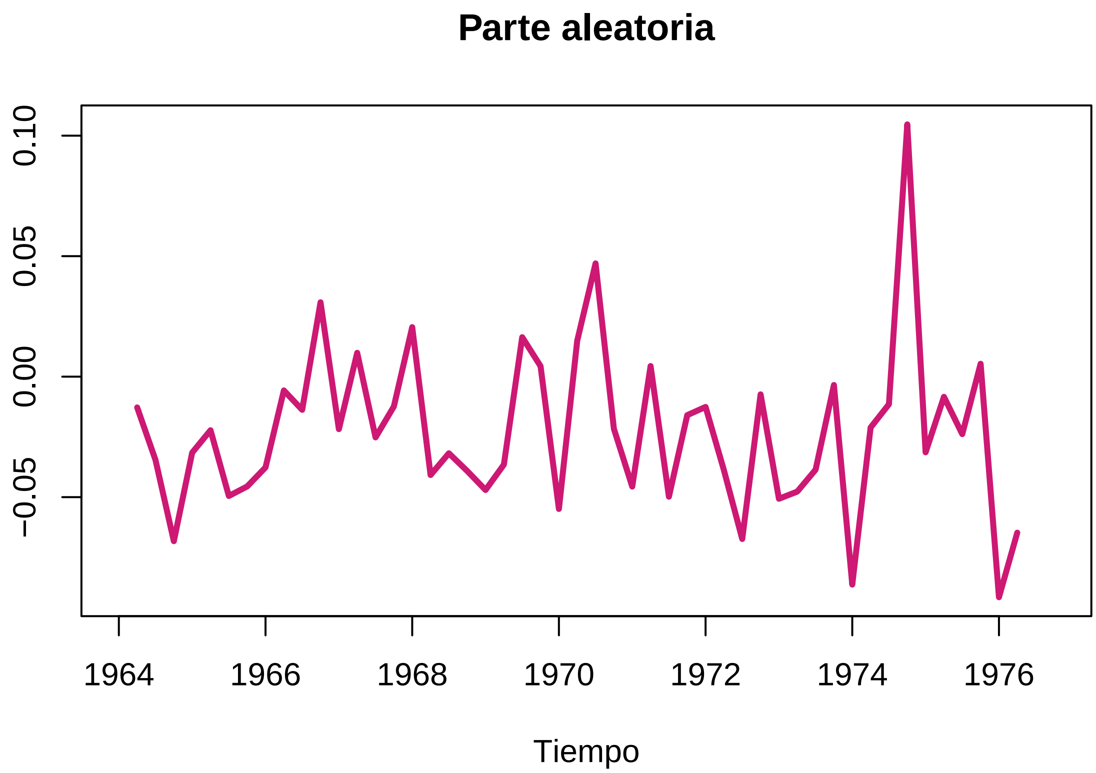
```
# Creamos un ciclo promedio que estime la parte estacional,
# usando la serie sin tendencia.
d = 4
k = length(datos.ts4) / d # Numero de ciclos de la serie sin tendencia
w = rep(0, 4)
# Para el resto de los trimestres
for (i in 1:4)
  w[i] = sum(datos.ts4[d * (0:(k-1)) + i], na.rm = TRUE) / k

# Ahora, ajustamos el ciclo obtenido
ciclo  = w - mean(w)
ciclo = ts(rep(ciclo, times = k), start = start(Serie_sq),
           frequency = frequency(Serie_sq))
par(mfrow = c(1, 1))
plot(ciclo, col =20, lwd = 3, ylab = " ", xlab = "Tiempo",
     main = "Ciclos de la serie")# Es el ciclo de la serie
```



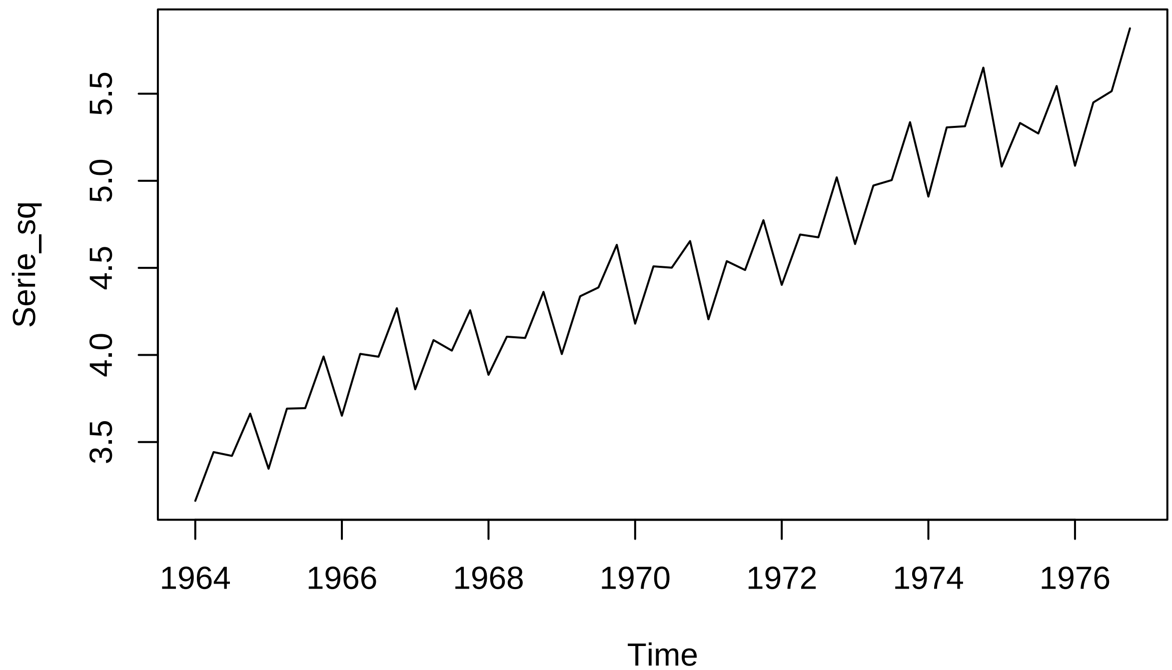

```
# Ciclos anuales
```

Ahora verifiquemos de manera gráfica

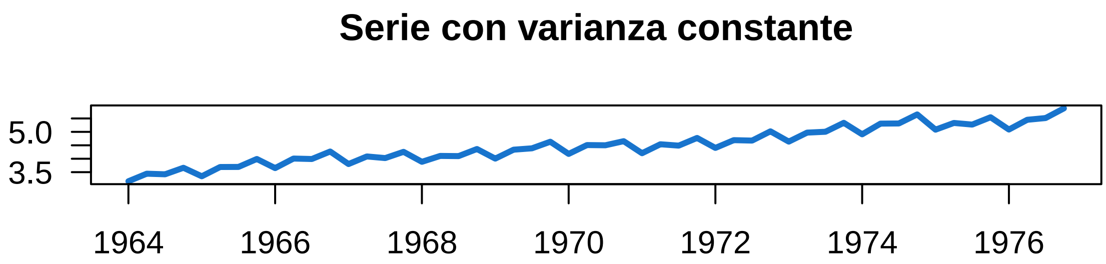
```
# Calculamos la parte aleatoria
parte_aleatoria = datos.ts4 - ciclo
plot(parte_aleatoria, main = "Parte aleatoria",
     col =30, lwd = 3, xlab = "Tiempo", ylab = "")
```

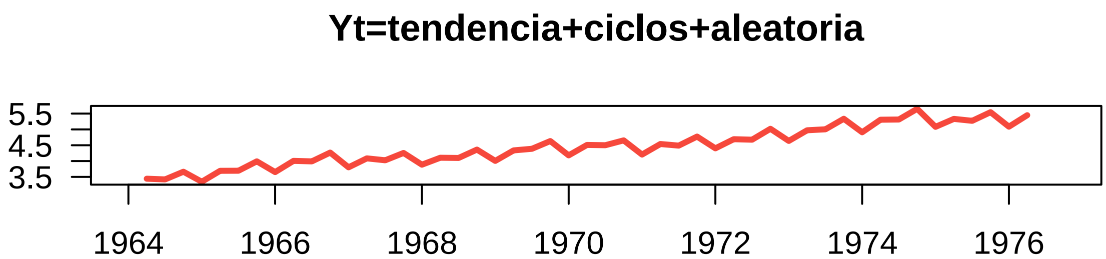



```
plot(Serie_sq)
```

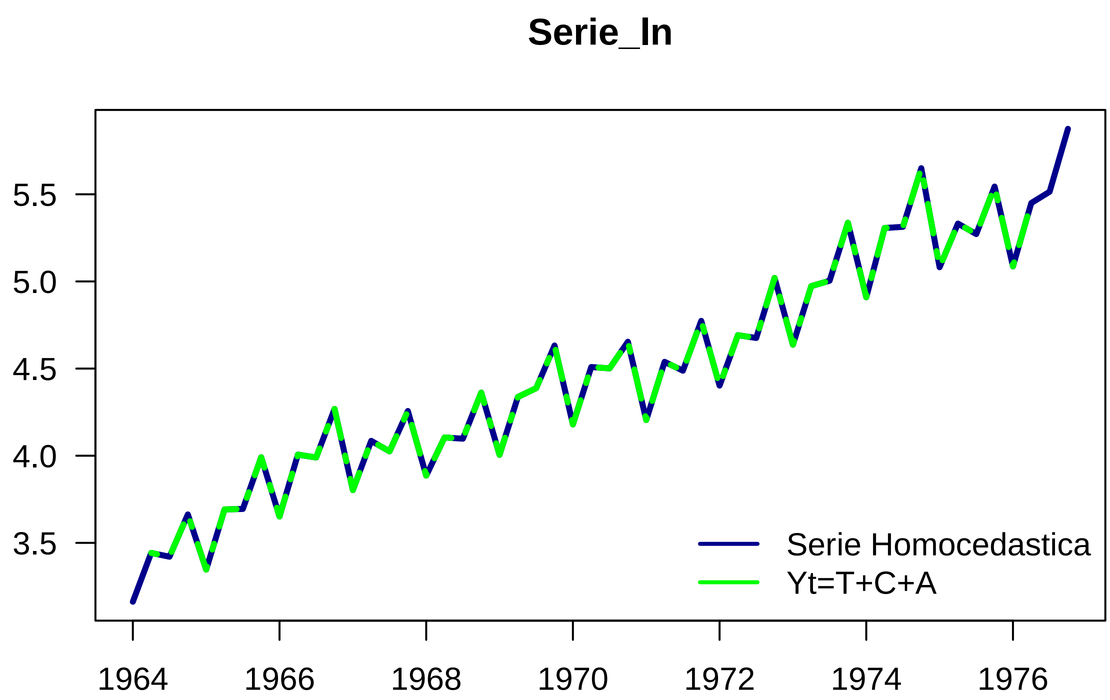



```r
# Con esto, ya tenemos nuestras series

componentes = tendencia + ciclo+parte_aleatoria
componentes = ts(componentes, start = start(Serie_sq), frequency = 4)
par(mfrow = c(2,1))
plot(Serie_sq, col=28, las=1, main="Serie con varianza constante", lwd=3, xlab="",ylab="")
plot(componentes, col = 18, lwd = 3, las=1, main="Yt=tendencia+ciclos+aleatoria", xlab="",ylab="")
```






```
par(mfrow = c(1,1))
plot(Serie_sq,col="darkblue", las=1, lwd=3,main="Serie_ln", ylab="",xlab="")
invisible(lines(componentes, type="l", lwd=3, col="green",lty=6))
legend("bottomright", col = c("darkblue","green"), lty = 1, lwd = 2, bty = "n",
       legend = c("Serie Homocedastica","Yt=T+C+A"), cex = 1)
```



¡Logramos identificar los componentes de la serie!

2. Missing data

Suponga que las observaciones de 1971 Qtr1, 1973 Qtr2 y 1973 Qtr3, son datos faltantes NA, es decir, sustituya estas observaciones por NA.

Use al menos dos métodos de imputación de la paqueteríaa imputeTS. ¿Cuál método es adecuado para estos datos? (Note que el valor imputado debe aproximarse al valor omitido).

Vamos a poner los datos que se piden, como NA:

```
Original=Serie[29]
Original[2]=Serie[38]
Original[3]=Serie[39]
Serie_2=Serie
Serie_2[29]=NA
Serie_2[38:39]=NA
Serie_2
```

```
##       Qtr1  Qtr2  Qtr3  Qtr4
## 1964 10.00 11.85 11.70 13.42
## 1965 11.20 13.63 13.65 15.93
## 1966 13.33 16.05 15.92 18.22
## 1967 14.46 16.69 16.20 18.12
## 1968 15.10 16.85 16.79 19.03
## 1969 16.04 18.81 19.25 21.46
```

```
## 1970 17.47 20.33 20.26 21.66
## 1971    NA 20.60 20.14 22.79
## 1972 19.38 22.01 21.86 25.20
## 1973 21.50    NA    NA 28.48
## 1974 24.10 28.16 28.23 31.92
## 1975 25.82 28.43 27.79 30.74
## 1976 25.87 29.70 30.41 34.52
```

Ahora, veamos método por método:

Un vistazo a la documentación de la paquetería mencionada nos menciona los siguientes métodos:

na_interpolation:

Missing Value Imputation by Interpolation. Acepta 3 tipos:

"linear" - for linear interpolation using approx (default choice)

```
Comparaciones=data.frame(Original)
na_interpolation_lineal=na_interpolation(Serie_2,option='linear')[29]-Original[1]
na_interpolation_lineal[2:3]=na_interpolation(Serie_2,option='linear')[38:39]-Original[2:3]
Comparaciones['na_interpolation_lineal']=na_interpolation_lineal
```

"spline" - for spline interpolation using spline

```
na_interpolation(Serie_2,option='spline')
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 21.91352 20.60000 20.14000 22.79000
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 22.89979 27.71251 28.48000
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```
na_interpolation_spline=na_interpolation(Serie_2,option='spline')[29]-Original[1]
na_interpolation_spline[2:3]=na_interpolation(Serie_2,option='spline')[38:39]-Original[2:3]
Comparaciones['na_interpolation_spline']=na_interpolation_spline
```

"stine" - for Stineman interpolation using stinterp

```
na_interpolation(Serie_2,option='stine')
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 21.13000 20.60000 20.14000 22.79000
```

```
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 23.10575 26.86069 28.48000
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```
na_interpolation_stine=na_interpolation(Serie_2,option='stine')[29]-Original[1]
na_interpolation_stine[2:3]=na_interpolation(Serie_2,option='stine')[38:39]-Original[2:3]
Comparaciones['na_interpolation_stine']=na_interpolation_stine
```

na_kalman:

Missing Value Imputation by Kalman Smoothing and State Space Models. Acepta los siguientes modelos: \begin{enumerate} "auto.arima" - For using the state space representation of arima model (using auto.arima) (default choice)

```
na_kalman(Serie_2,model='auto.arima')
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 17.88402 20.60000 20.14000 22.79000
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 24.99381 25.24268 28.48000
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```
na_kalman_auto.arima=na_kalman(Serie_2,model ='auto.arima')[29]-Original[1]
na_kalman_auto.arima[2:3]=na_kalman(Serie_2,model='auto.arima')[38:39]-Original[2:3]
Comparaciones['na_kalman_auto.arima']=na_kalman_auto.arima
```

"StructTS" - For using a structural model fitted by maximum likelihood (using StructTS)

```
na_kalman(Serie_2,model='StructTS')
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 17.98114 20.60000 20.14000 22.79000
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 24.82428 24.92793 28.48000
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```
na_kalman_StructTS=na_kalman(Serie_2,model='StructTS')[29]-Original[1]
na_kalman_StructTS[2:3]=na_kalman(Serie_2,model='StructTS')[38:39]-Original[2:3]
```

```
Comparaciones['na_StructTS']=na_kalman_StructTS
```

na_locf:

Missing Value Imputation by Last Observation Carried Forward. Acepta los siguientes métodos:

"locf" - for Last Observation Carried Forward (default choice)

```
na_locf(Serie_2,option = 'locf')
```

```
##       Qtr1  Qtr2  Qtr3  Qtr4
## 1964 10.00 11.85 11.70 13.42
## 1965 11.20 13.63 13.65 15.93
## 1966 13.33 16.05 15.92 18.22
## 1967 14.46 16.69 16.20 18.12
## 1968 15.10 16.85 16.79 19.03
## 1969 16.04 18.81 19.25 21.46
## 1970 17.47 20.33 20.26 21.66
## 1971 21.66 20.60 20.14 22.79
## 1972 19.38 22.01 21.86 25.20
## 1973 21.50 21.50 21.50 28.48
## 1974 24.10 28.16 28.23 31.92
## 1975 25.82 28.43 27.79 30.74
## 1976 25.87 29.70 30.41 34.52
```

```
na_locf=na_locf(Serie_2,option='locf')[29]-Original[1]
na_locf[2:3]=na_locf(Serie_2,option='locf')[38:39]-Original[2:3]
Comparaciones['na_locf']=na_locf
```

"nocb" - for Next Observation Carried Backward

```
na_locf(Serie_2,option = 'nocb')
```

```
##       Qtr1  Qtr2  Qtr3  Qtr4
## 1964 10.00 11.85 11.70 13.42
## 1965 11.20 13.63 13.65 15.93
## 1966 13.33 16.05 15.92 18.22
## 1967 14.46 16.69 16.20 18.12
## 1968 15.10 16.85 16.79 19.03
## 1969 16.04 18.81 19.25 21.46
## 1970 17.47 20.33 20.26 21.66
## 1971 20.60 20.60 20.14 22.79
## 1972 19.38 22.01 21.86 25.20
## 1973 21.50 28.48 28.48 28.48
## 1974 24.10 28.16 28.23 31.92
## 1975 25.82 28.43 27.79 30.74
## 1976 25.87 29.70 30.41 34.52
```

```
na_locf_nocb=na_locf(Serie_2,option='nocb')[29]-Original[1]
na_locf_nocb[2:3]=na_locf(Serie_2,option='nocb')[38:39]-Original[2:3]
Comparaciones['na_locf_nocb']=na_locf_nocb
```

na_ma Missing:

Value Imputation by Weighted Moving Average. Acepta los siguientes métodos:

"simple" - Simple Moving Average (SMA)

```r
na_ma(Serie_2,weighting = 'simple')
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 20.32875 20.60000 20.14000 22.79000
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 24.47286 25.36143 28.48000
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```r
na_ma_simple=na_ma(Serie_2,weighting='simple')[29]-Original[1]
na_ma_simple[2:3]=na_ma(Serie_2,weighting='simple')[38:39]-Original[2:3]
Comparaciones['na_ma_simple']=na_ma_simple
```

"linear" - Linear Weighted Moving Average (LWMA)

```r
na_ma(Serie_2,weighting = 'linear')
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 20.55065 20.60000 20.14000 22.79000
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 24.27452 25.54742 28.48000
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```r
na_ma_linear=na_interpolation(Serie_2,option='linear')[29]-Original[1]
na_ma_linear[2:3]=na_interpolation(Serie_2,option='linear')[38:39]-Original[2:3]
Comparaciones['na_ma_linear']=na_ma_linear
```

"exponential" - Exponential Weighted Moving Average (EWMA) (default choice)

```r
na_ma(Serie_2,weighting = 'exponential')
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 20.75900 20.60000 20.14000 22.79000
```

```
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 24.03682 25.77500 28.48000
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```
na_ma_exponential=na_ma(Serie_2,weighting='exponential')[29]-Original[1]
na_ma_exponential[2:3]=na_ma(Serie_2,weighting='exponential')[38:39]-Original[2:3]
Comparaciones['na_exponential']=na_ma_exponential
```

na_mean Missing:

Value Imputation by Mean Value. Acepta los siguientes métodos:

"mean" - take the mean for imputation (default choice)

```
na_mean(Serie_2,option = 'mean')
```

```
##       Qtr1  Qtr2  Qtr3  Qtr4
## 1964 10.00 11.85 11.70 13.42
## 1965 11.20 13.63 13.65 15.93
## 1966 13.33 16.05 15.92 18.22
## 1967 14.46 16.69 16.20 18.12
## 1968 15.10 16.85 16.79 19.03
## 1969 16.04 18.81 19.25 21.46
## 1970 17.47 20.33 20.26 21.66
## 1971 20.43 20.60 20.14 22.79
## 1972 19.38 22.01 21.86 25.20
## 1973 21.50 20.43 20.43 28.48
## 1974 24.10 28.16 28.23 31.92
## 1975 25.82 28.43 27.79 30.74
## 1976 25.87 29.70 30.41 34.52
```

```
na_mean=na_mean(Serie_2,option='mean')[29]-Original[1]
na_mean[2:3]=na_mean(Serie_2,option='mean')[38:39]-Original[2:3]
Comparaciones['na_mean']=na_mean
```

"median" - take the median for imputation

```
na_mean(Serie_2,option = 'median')
```

```
##       Qtr1  Qtr2  Qtr3  Qtr4
## 1964 10.00 11.85 11.70 13.42
## 1965 11.20 13.63 13.65 15.93
## 1966 13.33 16.05 15.92 18.22
## 1967 14.46 16.69 16.20 18.12
## 1968 15.10 16.85 16.79 19.03
## 1969 16.04 18.81 19.25 21.46
## 1970 17.47 20.33 20.26 21.66
## 1971 19.38 20.60 20.14 22.79
## 1972 19.38 22.01 21.86 25.20
## 1973 21.50 19.38 19.38 28.48
## 1974 24.10 28.16 28.23 31.92
## 1975 25.82 28.43 27.79 30.74
## 1976 25.87 29.70 30.41 34.52
```

```
na_mean_median=na_mean(Serie_2,option='median')[29]-Original[1]
na_mean_median[2:3]=na_mean(Serie_2,option='median')[38:39]-Original[2:3]
```

```r
Comparaciones['na_mean_median']=na_interpolation_lineal
```

"mode" - take the mode for imputation

```r
na_mean(Serie_2,option = 'mode')
```

```
##       Qtr1  Qtr2  Qtr3  Qtr4
## 1964 10.00 11.85 11.70 13.42
## 1965 11.20 13.63 13.65 15.93
## 1966 13.33 16.05 15.92 18.22
## 1967 14.46 16.69 16.20 18.12
## 1968 15.10 16.85 16.79 19.03
## 1969 16.04 18.81 19.25 21.46
## 1970 17.47 20.33 20.26 21.66
## 1971 10.00 20.60 20.14 22.79
## 1972 19.38 22.01 21.86 25.20
## 1973 21.50 10.00 10.00 28.48
## 1974 24.10 28.16 28.23 31.92
## 1975 25.82 28.43 27.79 30.74
## 1976 25.87 29.70 30.41 34.52
```

```r
na_mean_mode=na_mean(Serie_2,option='mode')[29]-Original[1]
na_mean_mode[2:3]=na_mean(Serie_2,option='mode')[38:39]-Original[2:3]
Comparaciones['na_mean_mode']=na_mean_mode
```

"harmonic" - take the harmonic mean

```r
na_mean(Serie_2,option = 'harmonic')
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 18.67181 20.60000 20.14000 22.79000
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 18.67181 18.67181 28.48000
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```r
na_mean_harmonic=na_mean(Serie_2,option='harmonic')[29]-Original[1]
na_mean_harmonic[2:3]=na_mean(Serie_2,option='harmonic')[38:39]-Original[2:3]
Comparaciones['na_mean_harmonic']=na_mean_harmonic
```

"geometric" - take the geometric mean

```r
na_mean(Serie_2,option = 'geometric')
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
```

```
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 19.54094 20.60000 20.14000 22.79000
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 19.54094 19.54094 28.48000
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```
na_mean_geometric=na_mean(Serie_2,option='geometric')[29]-Original[1]
na_mean_geometric[2:3]=na_mean(Serie_2,option='geometric')[38:39]-Original[2:3]
Comparaciones['na_mean_geometric']=na_mean_geometric
```

na_random:

Missing Value Imputation by Random Sample

```
na_random(Serie_2)
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 28.22092 20.60000 20.14000 22.79000
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 16.31476 26.27779 28.48000
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```
na_random=na_random(Serie_2)[29]-Original[1]
na_random[2:3]=na_random(Serie_2)[38:39]-Original[2:3]
Comparaciones['na_random']=na_random
```

na_seadec:

Seasonally Decomposed Missing Value Imputation. Admite los siguiente métodos:

"interpolation" - Imputation by Interpolation (default choice)

```
na_seadec(Serie_2,algorithm = 'interpolation')
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 18.02499 20.60000 20.14000 22.79000
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 24.90596 25.54880 28.48000
```

```
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```
na_seadec_interpolation=na_seadec(Serie_2,algorithm='interpolation')[29]-Original[1]
na_seadec_interpolation[2:3]=na_seadec(Serie_2,algorithm='interpolation')[38:39]-Original[2:3]
Comparaciones['na_seadec_interpolation']=na_seadec_interpolation
```

"locf" - Imputation by Last Observation Carried Forward

```
na_seadec(Serie_2,algorithm = 'locf')
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 17.78802 20.60000 20.14000 22.79000
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 23.93409 23.60507 28.48000
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```
na_seadec_locf=na_seadec(Serie_2,algorithm='locf')[29]-Original[1]
na_seadec_locf[2:3]=na_seadec(Serie_2,algorithm='locf')[38:39]-Original[2:3]
Comparaciones['na_seadec_locf']=na_seadec_locf
```

"mean" - Imputation by Mean Value

```
na_seadec(Serie_2,algorithm = 'mean')
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 18.33885 20.60000 20.14000 22.79000
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 20.67408 20.34506 28.48000
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```
na_seadec_mean=na_seadec(Serie_2,algorithm = 'mean')[29]-Original[1]
na_seadec_mean[2:3]=na_seadec(Serie_2,algorithm = 'mean')[38:39]-Original[2:3]
Comparaciones['na_seadec_mean']=na_seadec_mean
```

"random" - Imputation by Random Sample

```
na_seadec(Serie_2,algorithm = 'random')
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 15.94081 20.60000 20.14000 22.79000
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 24.06344 23.36207 28.48000
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```
na_seadec_random=na_seadec(Serie_2,algorithm = 'random')[29]-Original[1]
na_seadec_random[2:3]=na_seadec(Serie_2,algorithm = 'random')[38:39]-Original[2:3]
Comparaciones['na_seadec_random']=na_seadec_random
```

"kalman" - Imputation by Kalman Smoothing and State Space Models

```
na_seadec(Serie_2,algorithm = 'kalman')
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 17.86519 20.60000 20.14000 22.79000
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 24.88983 24.98804 28.48000
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```
na_seadec_kalman=na_seadec(Serie_2,algorithm = 'kalman')[29]-Original[1]
na_seadec_kalman[2:3]=na_seadec(Serie_2,algorithm = 'kalman')[38:39]-Original[2:3]
Comparaciones['na_seadec_kalman']=na_seadec_kalman
```

"ma" - Imputation by Weighted Moving Average

```
na_seadec(Serie_2,algorithm = 'ma')
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 18.16021 20.60000 20.14000 22.79000
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 24.58540 25.62559 28.48000
```

```
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```
na_seadec_ma=na_seadec(Serie_2,algorithm = 'ma')[29]-Original[1]
na_seadec_ma[2:3]=na_seadec(Serie_2,algorithm = 'ma')[38:39]-Original[2:3]
Comparaciones['na_seadec_ma']=na_seadec_ma
```

na_seasplit:

Seasonally Splitted Missing Value Imputation. Admite los siguiente métodos:

"interpolation" - Imputation by Interpolation (default choice)

```
na_seasplit(Serie_2,algorithm = 'interpolation')
```

```
##        Qtr1   Qtr2   Qtr3   Qtr4
## 1964 10.000 11.850 11.700 13.420
## 1965 11.200 13.630 13.650 15.930
## 1966 13.330 16.050 15.920 18.220
## 1967 14.460 16.690 16.200 18.120
## 1968 15.100 16.850 16.790 19.030
## 1969 16.040 18.810 19.250 21.460
## 1970 17.470 20.330 20.260 21.660
## 1971 18.425 20.600 20.140 22.790
## 1972 19.380 22.010 21.860 25.200
## 1973 21.500 25.085 25.045 28.480
## 1974 24.100 28.160 28.230 31.920
## 1975 25.820 28.430 27.790 30.740
## 1976 25.870 29.700 30.410 34.520
```

```
na_seasplit_interpolation=na_seasplit(Serie_2,algorithm = 'interpolation')[29]-Original[1]
na_seasplit_interpolation[2:3]=na_seasplit(Serie_2,algorithm = 'interpolation')[38:39]-Original[2:3]
Comparaciones['na_seadec_interpolation']=na_seasplit_interpolation
```

"locf" - Imputation by Last Observation Carried Forward

```
na_seasplit(Serie_2,algorithm = 'locf')
```

```
##       Qtr1  Qtr2  Qtr3  Qtr4
## 1964 10.00 11.85 11.70 13.42
## 1965 11.20 13.63 13.65 15.93
## 1966 13.33 16.05 15.92 18.22
## 1967 14.46 16.69 16.20 18.12
## 1968 15.10 16.85 16.79 19.03
## 1969 16.04 18.81 19.25 21.46
## 1970 17.47 20.33 20.26 21.66
## 1971 17.47 20.60 20.14 22.79
## 1972 19.38 22.01 21.86 25.20
## 1973 21.50 22.01 21.86 28.48
## 1974 24.10 28.16 28.23 31.92
## 1975 25.82 28.43 27.79 30.74
## 1976 25.87 29.70 30.41 34.52
```

```
na_seadec_locf=na_seasplit(Serie_2,algorithm = 'locf')[29]-Original[1]
na_seadec_locf[2:3]=na_seasplit(Serie_2,algorithm = 'locf')[38:39]-Original[2:3]
Comparaciones['na_seadec_locf']=na_seadec_locf
```

"mean" - Imputation by Mean Value

```r
na_seasplit(Serie_2,algorithm = 'mean')
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 17.85583 20.60000 20.14000 22.79000
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 20.25917 20.18333 28.48000
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```r
na_seadec_mean=na_seasplit(Serie_2,algorithm = 'mean')[29]-Original[1]
na_seadec_mean[2:3]=na_seasplit(Serie_2,algorithm = 'mean')[38:39]-Original[2:3]
Comparaciones['na_seadec_mean']=na_seadec_mean
```

"random" - Imputation by Random Sample

```r
na_seasplit(Serie_2,algorithm = 'random')
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 14.33650 20.60000 20.14000 22.79000
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 21.55354 26.53126 28.48000
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```r
na_seadec_random=na_seasplit(Serie_2,algorithm = 'random')[29]-Original[1]
na_seadec_random[2:3]=na_seasplit(Serie_2,algorithm = 'random')[38:39]-Original[2:3]
Comparaciones['na_seadec_random']=na_seadec_random
```

"kalman" - Imputation by Kalman Smoothing and State Space Models

```r
na_seasplit(Serie_2,algorithm = 'kalman')
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 18.37287 20.60000 20.14000 22.79000
```

```
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 25.07742 24.68497 28.48000
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```
na_seadec_kalman=na_seasplit(Serie_2,algorithm = 'kalman')[29]-Original[1]
na_seadec_kalman[2:3]=na_seasplit(Serie_2,algorithm = 'kalman')[38:39]-Original[2:3]
Comparaciones['na_seadec_kalman']=na_seadec_kalman
```

"ma" - Imputation by Weighted Moving Average

```
na_seasplit(Serie_2,algorithm = 'ma')
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1964 10.00000 11.85000 11.70000 13.42000
## 1965 11.20000 13.63000 13.65000 15.93000
## 1966 13.33000 16.05000 15.92000 18.22000
## 1967 14.46000 16.69000 16.20000 18.12000
## 1968 15.10000 16.85000 16.79000 19.03000
## 1969 16.04000 18.81000 19.25000 21.46000
## 1970 17.47000 20.33000 20.26000 21.66000
## 1971 18.78800 20.60000 20.14000 22.79000
## 1972 19.38000 22.01000 21.86000 25.20000
## 1973 21.50000 24.70172 24.58724 28.48000
## 1974 24.10000 28.16000 28.23000 31.92000
## 1975 25.82000 28.43000 27.79000 30.74000
## 1976 25.87000 29.70000 30.41000 34.52000
```

```
na_seadec_ma=na_seasplit(Serie_2,algorithm = 'ma')[29]-Original[1]
na_seadec_ma[2:3]=na_seasplit(Serie_2,algorithm = 'ma')[38:39]-Original[2:3]
Comparaciones['na_seadec_ma']=na_seadec_ma
```

En el dataframe Comparaciones, lo que hacemos es guardar las diferencias respecto a la observación original, acorde al método respectivo de ajuste.

Veamos cuál es el que minimiza dicho error:

```
Qr1_1971<-Comparaciones[1,]
Qr2_1973<-Comparaciones[2,]
Qr3_1973<-Comparaciones[3,]

#Para el primer trimiestre de 1971
dif_minima_Qr1_1971<-min(abs(Qr1_1971))
dif_minima_Qr1_1971
```

```
## [1] 0.1758333
```

```
metodo_minimiza_Qr1_1971<-which.min(abs(Qr1_1971))
metodo_minimiza_Qr1_1971
```

```
## na_seadec_mean
##             20
```

```
#Los 5 mejores
aux_indices1<-(sort.list(abs(Qr1_1971)))[1:5]
```

```
## Warning in xtfrm.data.frame(x): cannot xtfrm data frames
```

```r
i=1
for (indice in aux_indices1){
  print(Qr1_1971[indice])
  i=i+1
}
```

```
##   na_seadec_mean
## 1      0.1758333
##   na_kalman_auto.arima
## 1            0.2040217
##   na_seadec_locf
## 1          -0.21
##   na_StructTS
## 1    0.301137
##   na_seadec_kalman
## 1        0.6928704
```

```r
#Para el segundo trimiestre de 1973
dif_minima_Qr2_1973<-min(abs(Qr2_1973))
dif_minima_Qr2_1973
```

```
## [1] 0.02827586
```

```r
metodo_minimiza_Qr2_1973<-which.min(abs(Qr2_1973))
metodo_minimiza_Qr2_1973
```

```
## na_seadec_ma
##           23
```

```r
#Los 5 mejores
aux_indices2<-(sort.list(abs(Qr2_1973)))[1:5]
```

```
## Warning in xtfrm.data.frame(x): cannot xtfrm data frames
```

```r
i=1
for (indice in aux_indices2){
  print(Qr2_1973[indice])
  i=i+1
}
```

```
##   na_seadec_ma
## 2  -0.02827586
##   na_StructTS
## 2  0.09427935
##   na_ma_simple
## 2   -0.2571429
##   na_kalman_auto.arima
## 2            0.2638126
##   na_seadec_kalman
## 2        0.3474205
```

```r
#Para el tercer trimiestre de 1973
dif_minima_Qr3_1973<-min(abs(Qr3_1973))
dif_minima_Qr3_1973
```

```
## [1] 0.005
```

```r
metodo_minimiza_Qr3_1973<-which.min(abs(Qr3_1973))
metodo_minimiza_Qr3_1973
```

```
## na_seadec_interpolation
##                      18
```

```
#Los 5 mejores
aux_indices3<-(sort.list(abs(Qr3_1973)))[1:5]
```

```
## Warning in xtfrm.data.frame(x): cannot xtfrm data frames
```

```
i=1
for (indice in aux_indices3){
  print(Qr2_1973[indice])
  i=i+1
}
```

```
##   na_seadec_interpolation
## 2                   0.355
##   na_StructTS
## 2  0.09427935
##   na_kalman_auto.arima
## 2            0.2638126
##   na_ma_simple
## 2   -0.2571429
##   na_seadec_kalman
## 2        0.3474205
```

3. Ajuste

Con los datos observados completos, ajuste un modelo ARIMA o SARIMA adecuado.

Obtenga correlogramas, revise si los parámetros son significativos, compruebe los supuestos (que los residuales sean ruido blanco con distribución normal). Obtenga dos o más posibles modelos, realice análisis de residuales y calcule medidas de bondad de ajuste. Haga la comparación para decidir cuál modelo sería el más adecuado.

Estacionariedad

En la primera parte vimos que si le aplicamos la transformación raíz cuadrada a la serie original, pasa la prueba para varianza constnte:

```
bptest(Serie_sq~tiempo)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  Serie_sq ~ tiempo
## BP = 0.8651, df = 1, p-value = 0.3523
```

Ahora, las pruebas para estacionariedad:

```
adf.test(Serie_sq)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  Serie_sq
## Dickey-Fuller = -1.415, Lag order = 3, p-value = 0.8097
## alternative hypothesis: stationary
```

```
kpss.test(Serie_sq)
```

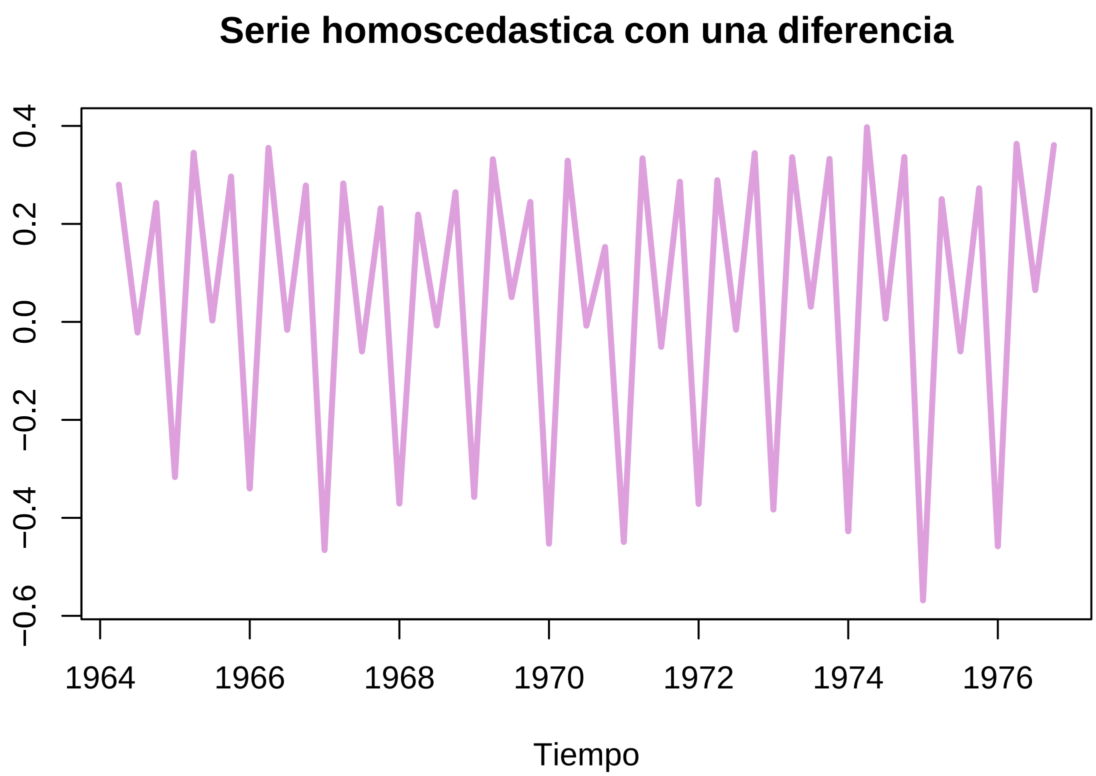
```
## Warning in kpss.test(Serie_sq): p-value smaller than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  Serie_sq
## KPSS Level = 1.389, Truncation lag parameter = 3, p-value = 0.01
```

No las pasa.

Aplicamos una diferencia

```
plot(diff(Serie_sq), col = "plum", lwd = 3, xlab = "Tiempo", ylab = " ",
     main = "Serie homoscedastica con una diferencia" )
```




```
adf.test(diff(Serie_sq))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(Serie_sq)
## Dickey-Fuller = -1.6911, Lag order = 3, p-value = 0.6986
## alternative hypothesis: stationary
```

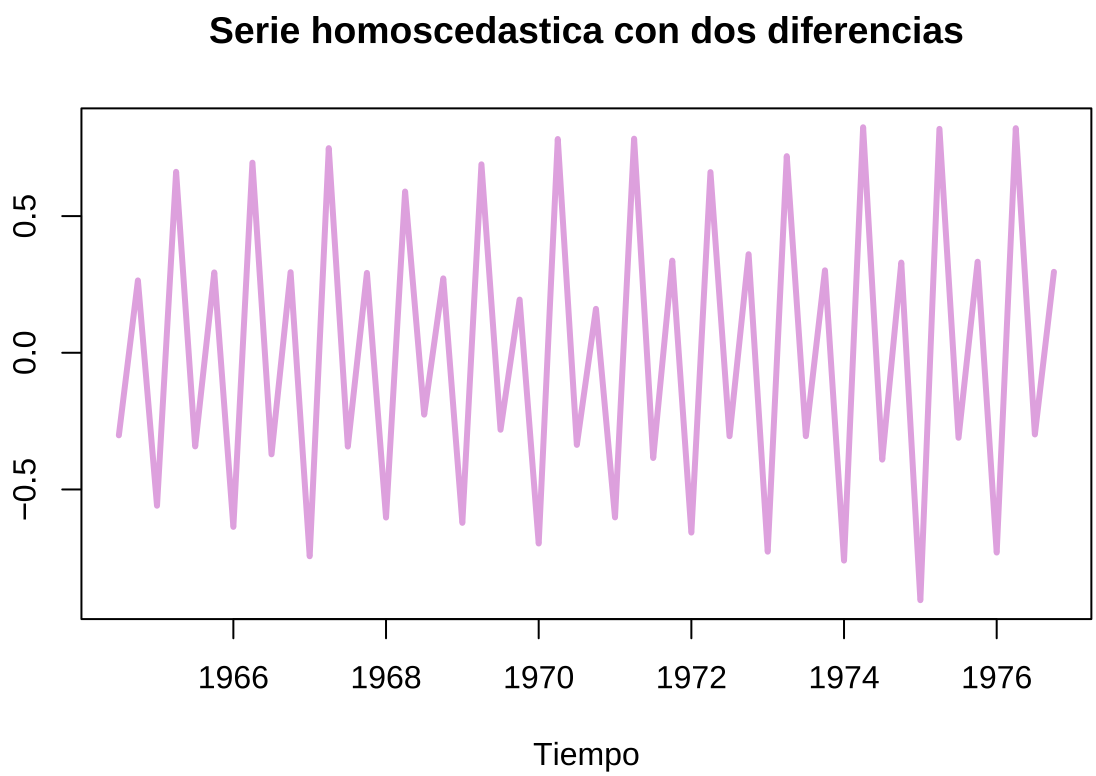
```
kpss.test(diff(Serie_sq))
```

```
## Warning in kpss.test(diff(Serie_sq)): p-value greater than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  diff(Serie_sq)
## KPSS Level = 0.1743, Truncation lag parameter = 3, p-value = 0.1
```

Se contradicen. Apliquemos otra diferencia

```
plot(diff(diff(Serie_sq)), col = "plum", lwd = 3, xlab = "Tiempo", ylab = " ",
     main = "Serie homoscedastica con dos diferencias" )
```




```
adf.test(diff(diff(Serie_sq)))
```

```
## Warning in adf.test(diff(diff(Serie_sq))): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(diff(Serie_sq))
## Dickey-Fuller = -4.2222, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

```
kpss.test(diff(diff(Serie_sq)))
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  diff(diff(Serie_sq))
## KPSS Level = 0.44788, Truncation lag parameter = 3, p-value = 0.05652
```

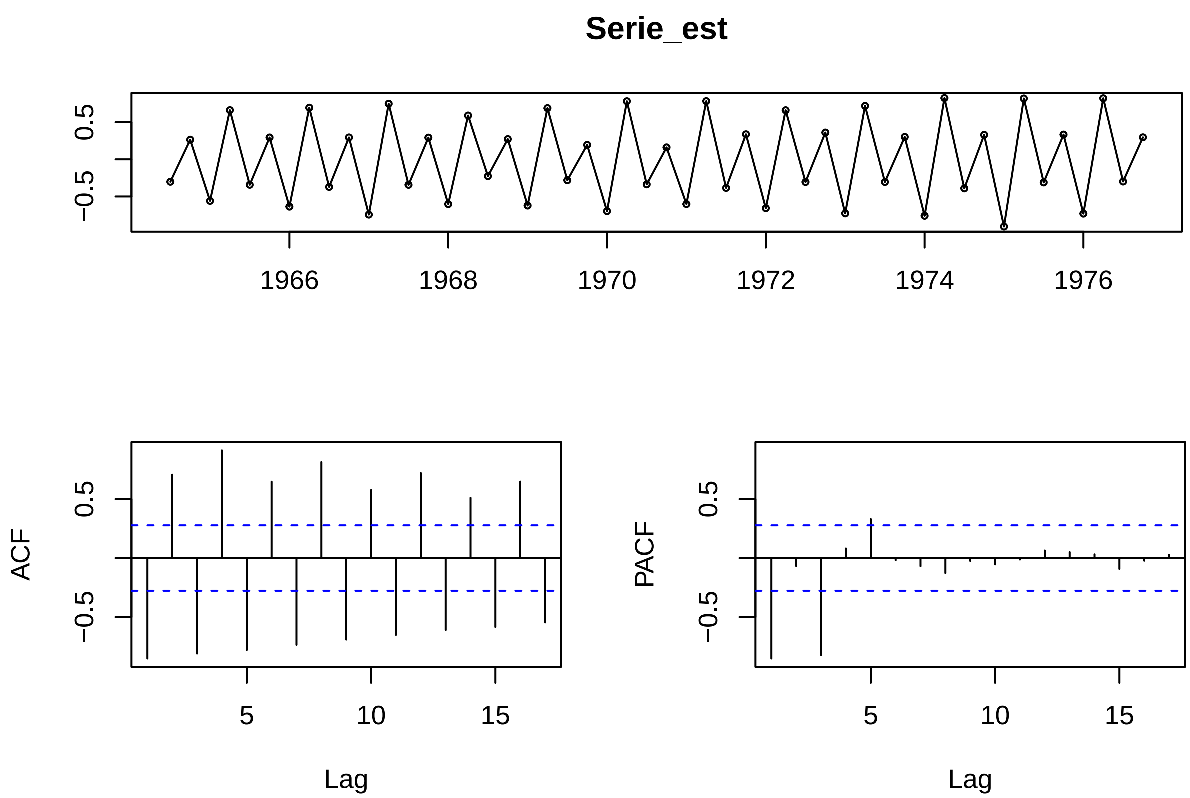
Pasa las dos pruebas si aplicamos dos diferencias; trabajaremos con esa.

Correlogramas

```
Serie_est<-diff(diff(Serie_sq))
```

Visualizamos ACF y PACF

```
tsdisplay(Serie_est)
```




```
k=length(Serie_est)
banda<-qnorm(0.95)/(sqrt(k))
auxacf=acf(Serie_est,plot = F)#MA(6)
ACF_superior<-sum(auxacf$acf>banda)
ACF_inferior<-sum(auxacf$acf< -banda)
superan_banda_acf<-ACF_superior+ACF_inferior
superan_banda_acf
```

```
## [1] 16
```

```
pauxacf=pacf(Serie_est,plot = F)#AR(6)
PACF_superior<-sum(pauxacf$acf>banda)
PACF_inferior<-sum(pauxacf$acf< -banda)
superan_banda_pacf<-PACF_superior+PACF_inferior
superan_banda_pacf
```

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

Parece, a simple vista, que puede que tengamos un ARIMA(3,2,16). Sin embargo, no hemos considerado la parte de los ciclos aún.

Veamos el ARIMA(3,2,16)...

```
ARIMA<-arima(Serie_sq,order=c(3,2,16))
```

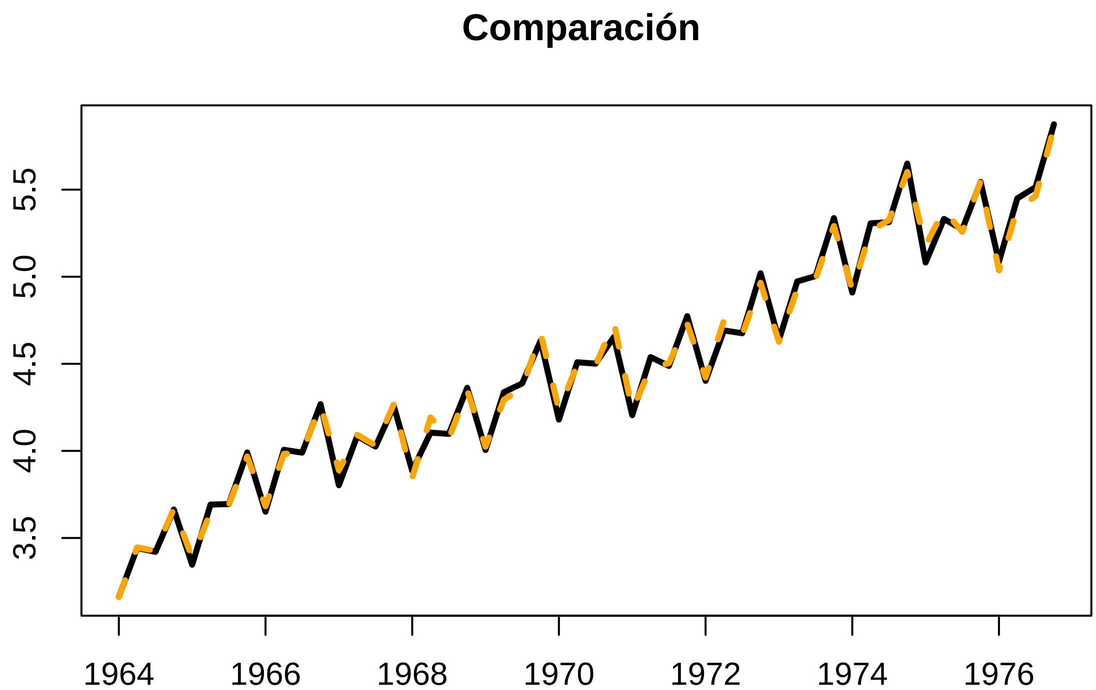
```
## Warning in stats::arima(x = x, order = order, seasonal = seasonal, xreg =
## xreg, : possible convergence problem: optim gave code = 1
```

```
ARIMA
```

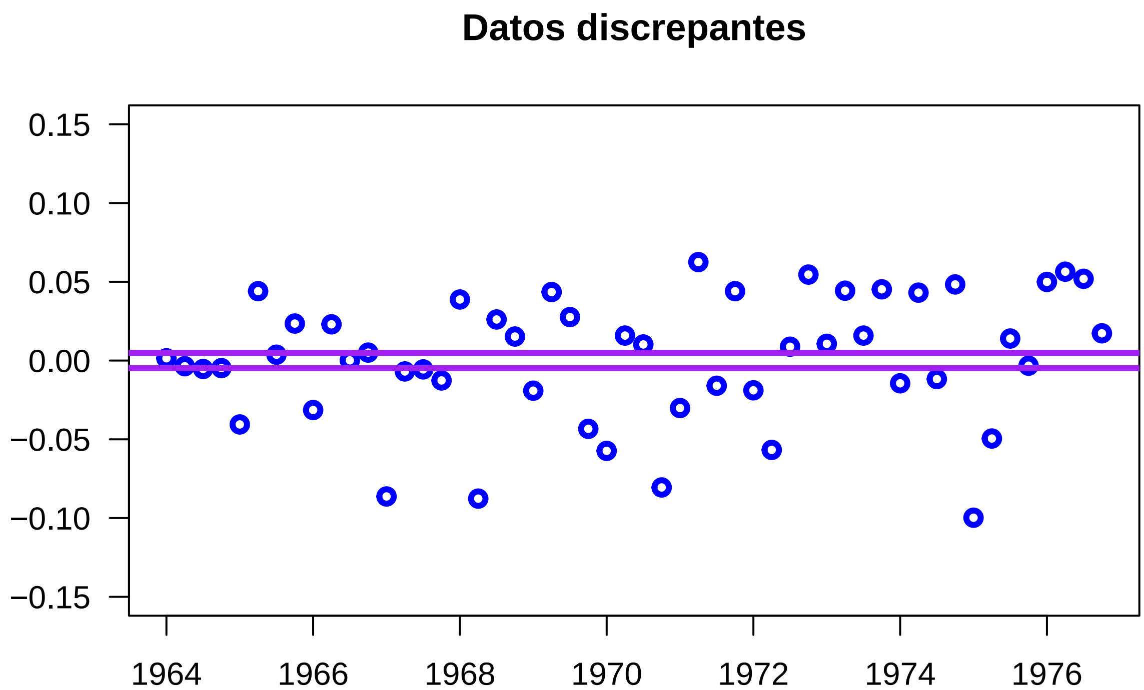
```
##
## Call:
## arima(x = Serie_sq, order = c(3, 2, 16))
##
## Coefficients:
##          ar1      ar2      ar3     ma1     ma2     ma3      ma4      ma5
##      -0.9944  -0.9932  -0.9987  0.0916  0.3508  0.1531  -0.5243  -0.3060
## s.e.  0.0072   0.0074   0.0017  0.1824  0.2205  0.2262   0.2668   0.2554
##          ma6      ma7      ma8      ma9     ma10    ma11    ma12    ma13
##      -0.2524  -0.2833  -0.6654  -0.1581  -0.1254  0.0519  0.5361  0.3603
## s.e.  0.1789   0.2009   0.2805   0.3226   0.2876  0.2392  0.2432  0.2316
##        ma14     ma15    ma16
##      0.1052  -0.0599  0.0633
## s.e. 0.2092   0.2192  0.2458
##
## sigma^2 estimated as 0.001648:  log likelihood = 75.1,  aic = -112.19
```

Ahora veamos la gráfica

```
ARIMA_ajuste <- (Serie_sq - residuals(ARIMA))
ts.plot(Serie_sq, lwd=3, main="Comparación ", ylab="", xlab="")
points(ARIMA_ajuste, type = "l", col ="orange", lty = 2, lwd=3)
```




```
###Y ahora los residuales
plot(ARIMA$residuals, type="p", col="blue", ylim=c(-0.15,0.15), ylab="", xlab="", main="Datos discrepan
abline(h=3*(var(ARIMA$residuals)), col="purple", lwd=3)
abline(h=-3*(var(ARIMA$residuals)), col="purple",lwd=3)
```



¡Tenemos muchísimos datos discrepantes! Estamos sobreajustando los datos

Ahora atacaremos el problema con un SARIMA(p,d,q)x(P,D,Q), usando la estrategia definida en la página 180 de Introduction to Time Series and Forecasting de Peter Brockwell y Richard Davis.

Por el análisis hecho en la primera parte, sabemos que:

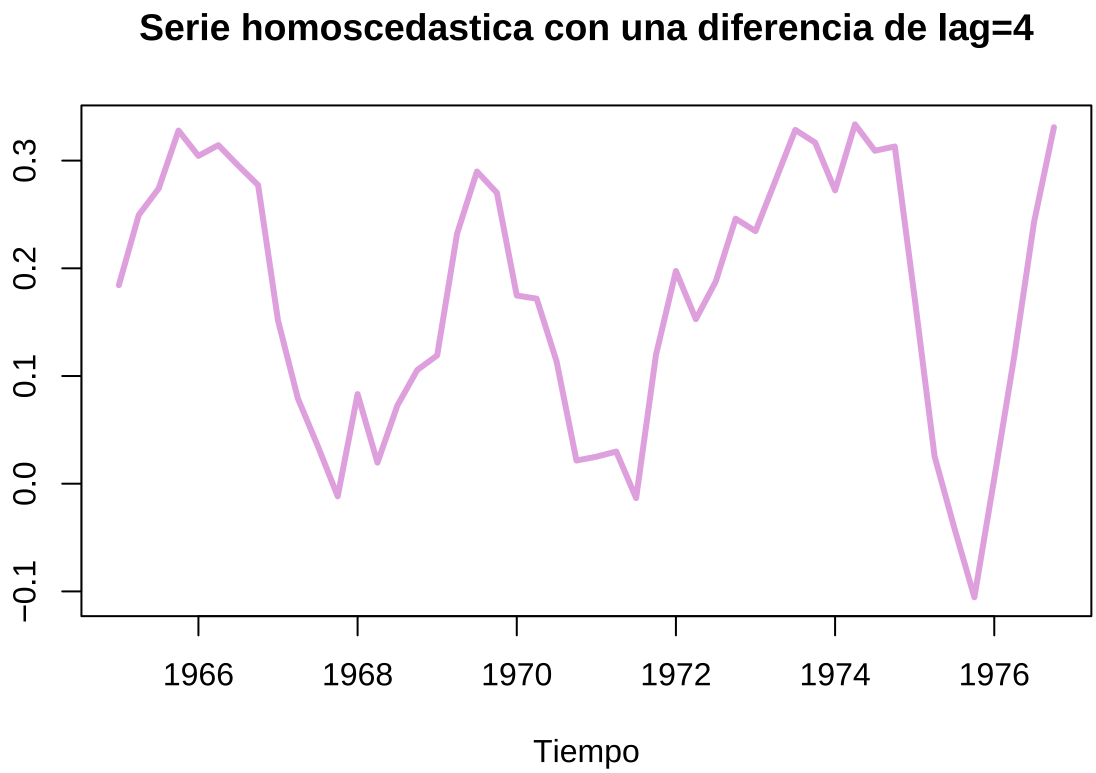
$$s = 4$$

Podemos estimar d y D como aquellos que hacen que

$$Y_t = (1 - B)^d (1 - B^s)^D$$

Sea estacionaria. Probemos con d=0 y D=1

```
plot(diff(Serie_sq,lag=4), col = "plum", lwd = 3, xlab = "Tiempo", ylab = " ",
     main = "Serie homoscedastica con una diferencia de lag=4" )
```




```
adf.test(diff(Serie_sq,lag=4))
```

```
## Warning in adf.test(diff(Serie_sq, lag = 4)): p-value smaller than printed p-
## value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(Serie_sq, lag = 4)
## Dickey-Fuller = -4.3985, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

```
kpss.test(diff(Serie_sq,lag=4))
```

```
## Warning in kpss.test(diff(Serie_sq, lag = 4)): p-value greater than printed p-
## value
```

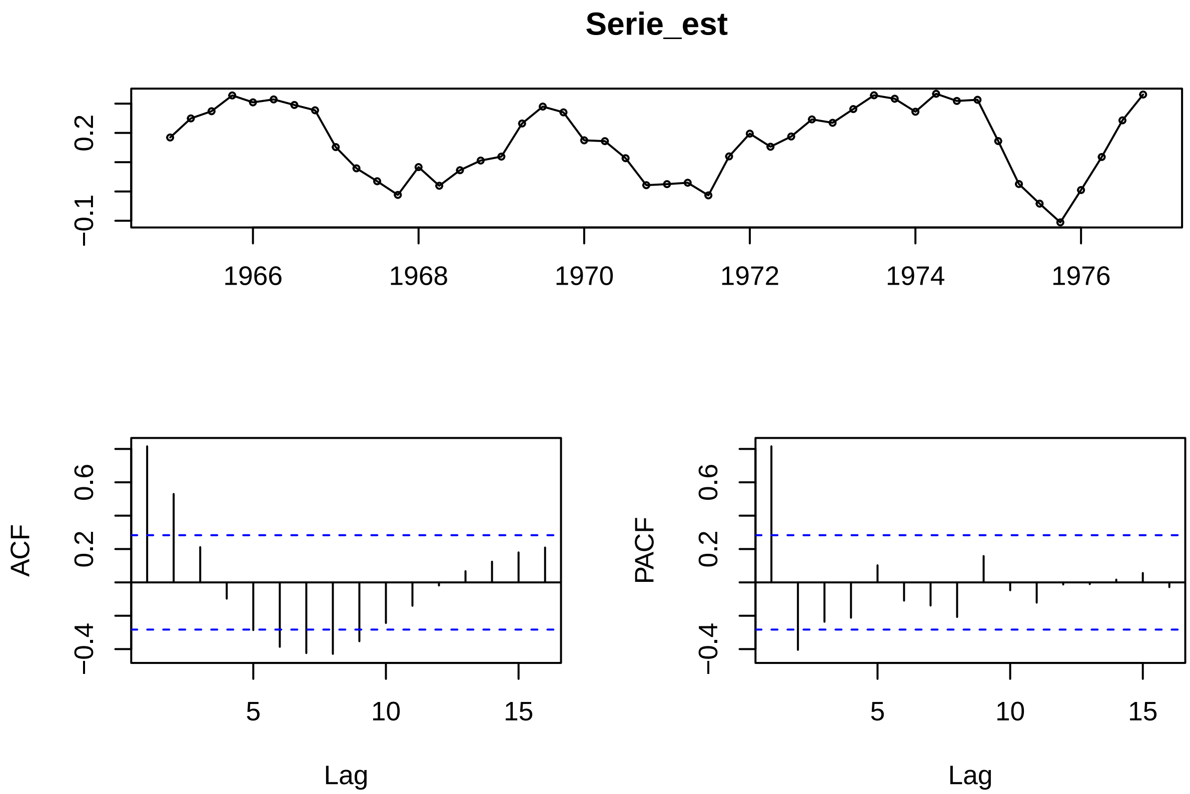
```
##
##  KPSS Test for Level Stationarity
##
## data:  diff(Serie_sq, lag = 4)
## KPSS Level = 0.076816, Truncation lag parameter = 3, p-value = 0.1
```

¡Las pasa! Entonces:

$$d = 0, D = 1$$

Trabajaremos con esta serie. Veamos su ACF y PACF

```
Serie_est<-diff(Serie_sq,lag=4)
tsdisplay(Serie_est)
```



¡Se ve mucho mejor que el anterior!

¿Cómo elegimos P y Q? La metodología seguida nos dice que veamos los lags que son múltiplos del ciclo (de s=4), y ver aquellos que se ajustan a un ARMA(P,Q). Es decir, en los lags ks $s = 4, k \in \mathbb{N} \setminus \{0\}$

¿Cómo elegimos p y q? Debemos fijarnos en aquellos lags entre 1 y s-1, es decir: Nos fijaremos en los lags $1, 2, 3$ y los ajustaremos a un ARMA(p,q)

Veamos el ACF

```
k=length(diff(Serie_sq,lag=4))
banda<-qnorm(0.95)/(sqrt(k))
auxacf=acf(diff(Serie_sq,lag=4),plot = F)#MA(6)
ACF_superior<-sum(auxacf$acf>banda)
ACF_inferior<-sum(auxacf$acf< -banda)
superan_banda_acf<-ACF_superior+ACF_inferior
superan_banda_acf
```

```
## [1] 8
```

```
#Los que superan las bandas del acf son:
which(abs(auxacf$acf) > banda)
```

```
## [1]  1  2  5  6  7  8  9 10
```

Por lo que:

$$q = 2, Q = 1$$

Para el PACF:

```
pauxacf=pacf(diff(Serie_sq,lag=4),plot = F)
PACF_superior<-sum(pauxacf$acf>banda)
PACF_inferior<-sum(pauxacf$acf< -banda)
superan_banda_pacf<-PACF_superior+PACF_inferior
superan_banda_pacf
```

```
## [1] 2
```

```
#Los que superan las bandas del pacf son:
which(abs(pauxacf$acf) > banda)
```

```
## [1] 1 2
```

Por lo que:

$$p = 2, P = 0$$

Entonces tenemos un modelo

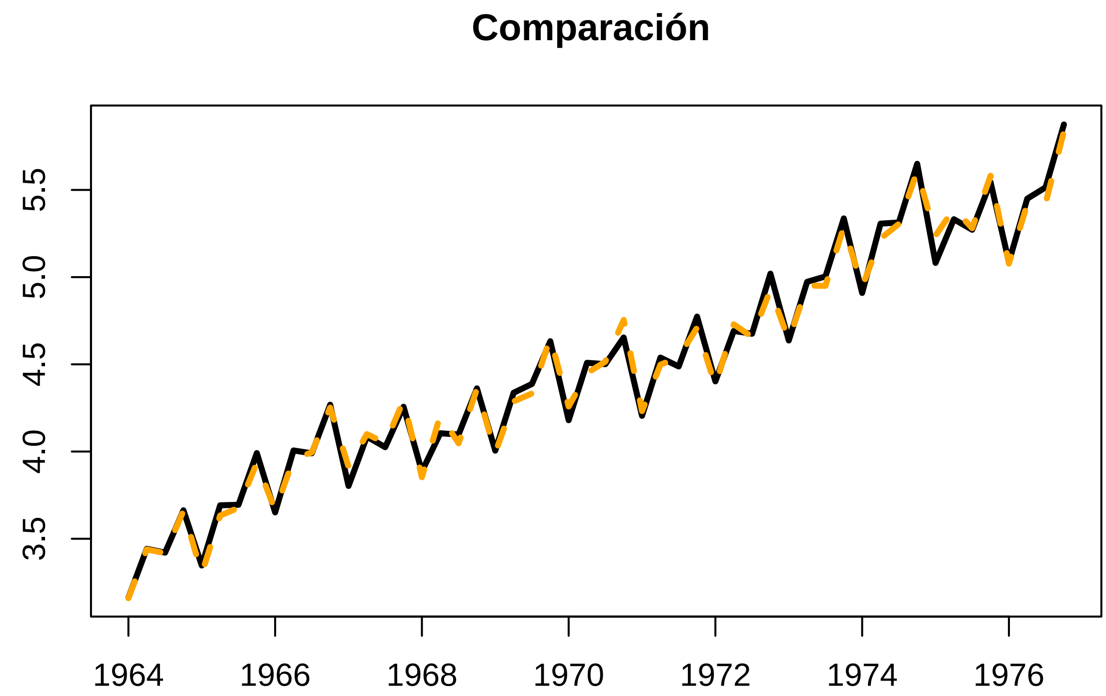
$$SARIMA(2, 0, 2) \times (0, 1, 1)_{[4]}$$

Ajustémoslo:

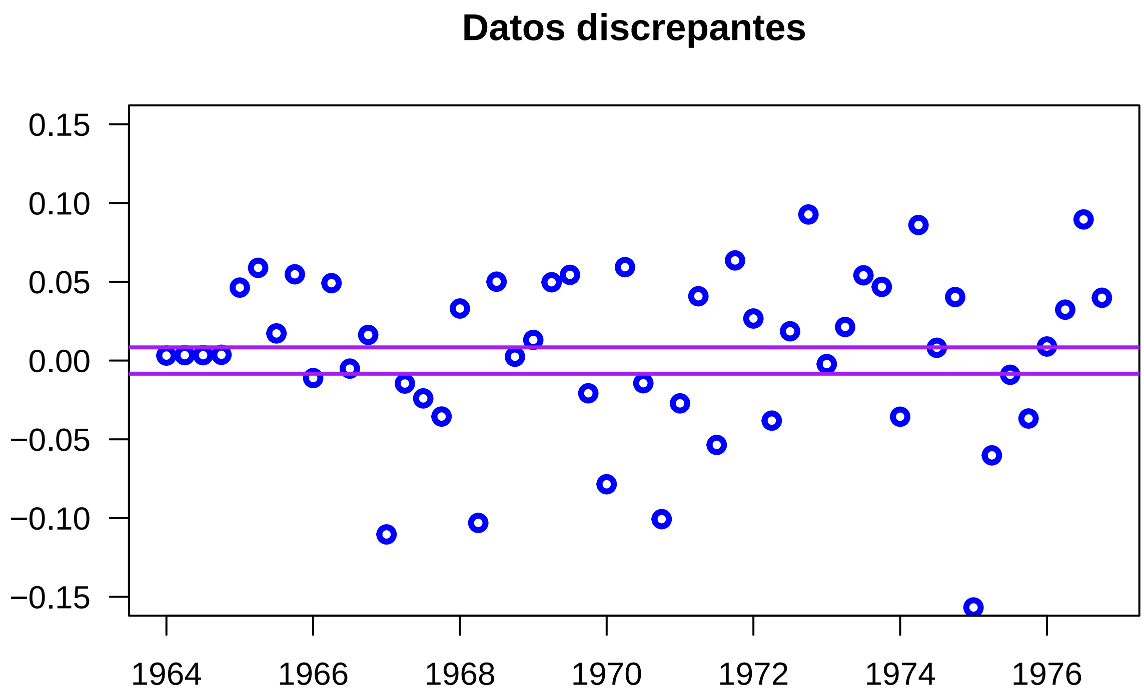
```
SARIMA<-arima(Serie_sq,order=c(2,0,2),seasonal=list(order=c(0,1,1),period=4), include.mean=F)
SARIMA
```

```
##
## Call:
## arima(x = Serie_sq, order = c(2, 0, 2), seasonal = list(order = c(0, 1, 1),
##     period = 4), include.mean = F)
##
## Coefficients:
##          ar1     ar2     ma1     ma2     sma1
##       0.0779  0.9168  1.1411  0.2977  -0.8051
## s.e.  0.5505  0.5574  0.4604  0.2173   0.1948
##
## sigma^2 estimated as 0.002979:  log likelihood = 68.54,  aic = -127.08
```

```
SARIMA_ajuste <- Serie_sq - residuals(SARIMA)
ts.plot(Serie_sq, lwd=3, main="Comparación ", ylab="", xlab="")
points(SARIMA_ajuste, type = "l", col ="orange", lty = 2, lwd=3)
```




```
###Y ahora los residuales
plot(SARIMA$residuals, type="p", col="blue", ylim=c(-.15,.15), ylab="", xlab="", main="Datos discrepant
abline(h=3*(var(SARIMA$residuals)), col="purple", lwd=2)
abline(h=-3*(var(SARIMA$residuals)), col="purple",lwd=2)
```



Comparemos con el que ajusta R:

```
modelo_automatico<-auto.arima((diff(Serie_sq,lag=4)))
modelo_automatico
```

```
## Series: (diff(Serie_sq, lag = 4))
## ARIMA(2,0,1)(0,0,2)[4] with non-zero mean
##
## Coefficients:
##          ar1      ar2      ma1     sma1     sma2    mean
##       1.6142  -0.7911  -0.5142  -0.3092  -0.3808  0.1685
## s.e.  0.1300   0.1123   0.1760   0.2012   0.2298  0.0089
##
## sigma^2 estimated as 0.002567:  log likelihood=75.7
## AIC=-137.41   AICc=-134.61   BIC=-124.31
```

Probemos con otra diferencia:

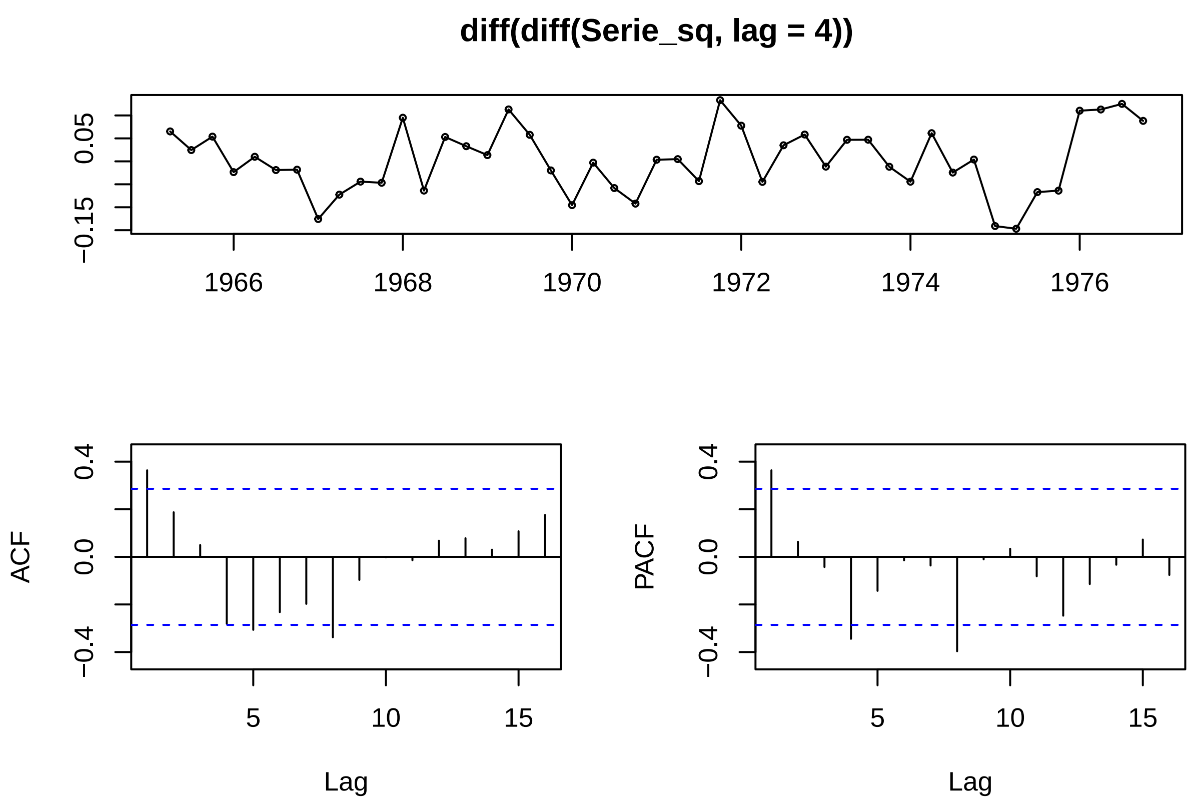
```
adf.test(diff(diff(Serie_sq,lag=4)))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(diff(Serie_sq, lag = 4))
## Dickey-Fuller = -3.5973, Lag order = 3, p-value = 0.0434
## alternative hypothesis: stationary
```

```r
kpss.test((diff(diff(Serie_sq,lag=4))))
```

```
## Warning in kpss.test((diff(diff(Serie_sq, lag = 4)))): p-value greater than
## printed p-value

##
##  KPSS Test for Level Stationarity
##
## data:  (diff(diff(Serie_sq, lag = 4)))
## KPSS Level = 0.06211, Truncation lag parameter = 3, p-value = 0.1
```

```r
tsdisplay(diff(diff(Serie_sq,lag=4)))
```



Sí pasa la prueba; ahora intentemos ajustar con dichas diferencias

```r
modelo_automatico<-auto.arima((diff(diff(Serie_sq,lag=4))))
modelo_automatico
```

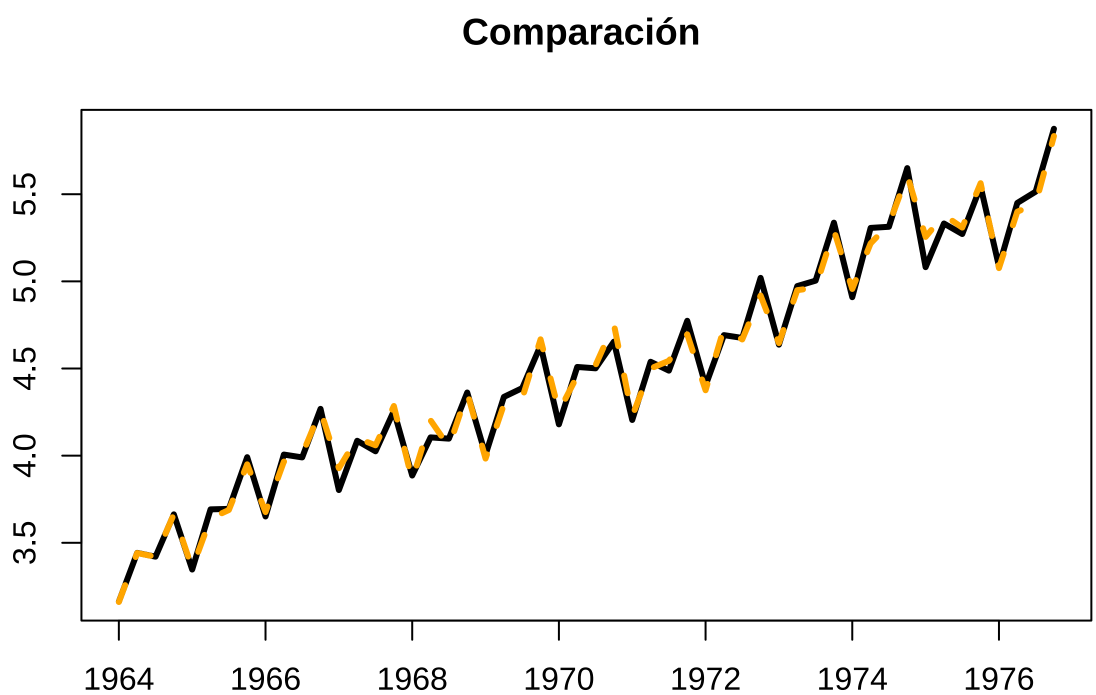
```
## Series: (diff(diff(Serie_sq, lag = 4)))
## ARIMA(1,0,0)(0,0,1)[4] with zero mean
##
## Coefficients:
##          ar1     sma1
##       0.2682  -0.6739
## s.e.  0.1468   0.1292
##
## sigma^2 estimated as 0.003262:  log likelihood=67.64
## AIC=-129.27   AICc=-128.72   BIC=-123.72
```

Entonces es un SARIMA(1,1,0)x(0,1,1)[4]

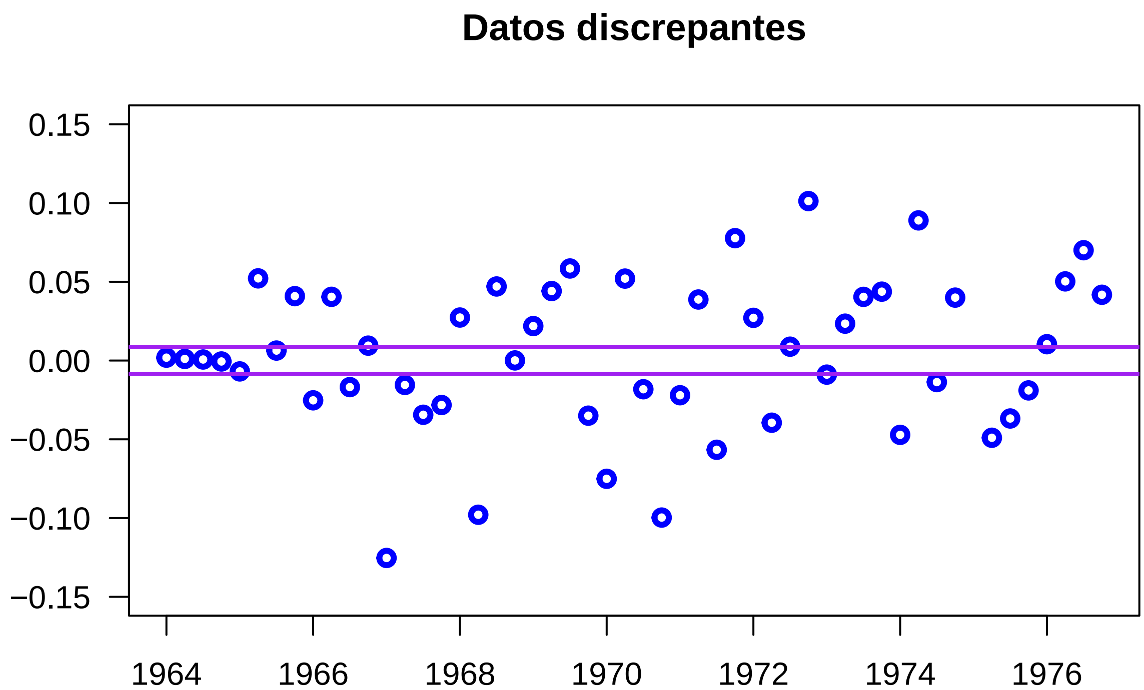
```
modelo_automatico<-arima(Serie_sq,order=c(1,1,0),seasonal=list(order=c(0,1,1),period=4))
modelo_automatico
```

```
##
## Call:
## arima(x = Serie_sq, order = c(1, 1, 0), seasonal = list(order = c(0, 1, 1),
##     period = 4))
##
## Coefficients:
##          ar1     sma1
##       0.2682  -0.6739
## s.e.  0.1468   0.1292
##
## sigma^2 estimated as 0.003123:  log likelihood = 67.64,  aic = -131.27
```

```
AUTOMATICO_ajuste <- Serie_sq - residuals(modelo_automatico)
ts.plot(Serie_sq, lwd=3, main="Comparación ", ylab="", xlab="")
points(AUTOMATICO_ajuste, type = "l", col ="orange", lty = 2, lwd=3)
```




```
###Y ahora los residuales
plot(modelo_automatico$residuals, type="p", col="blue", ylim=c(-.15,.15), ylab="", xlab="", main="Datos
abline(h=3*(var(modelo_automatico$residuals)), col="purple", lwd=2)
abline(h=-3*(var(modelo_automatico$residuals)), col="purple",lwd=2)
```



Tratemos de ajustar otro SARIMA manualmente; con el enfoque antes mencionado:

Veamos el ACF

```
k=length(diff(diff(Serie_sq,lag=4)))
banda<-qnorm(0.95)/(sqrt(k))
auxacf=acf(diff(diff(Serie_sq,lag=4)),plot = F)#MA(6)
ACF_superior<-sum(auxacf$acf>banda)
ACF_inferior<-sum(auxacf$acf< -banda)
superan_banda_acf<-ACF_superior+ACF_inferior
superan_banda_acf
```

```
## [1] 4
```

```
#Los que superan las bandas del acf son:
which(abs(auxacf$acf) > banda)
```

```
## [1] 1 4 5 8
```

Por lo que:

$$q = 1, Q = 2$$

Para el PACF:

```
pauxacf=pacf(diff(diff(Serie_sq,lag=4)),plot = F)
PACF_superior<-sum(pauxacf$acf>banda)
PACF_inferior<-sum(pauxacf$acf< -banda)
superan_banda_pacf<-PACF_superior+PACF_inferior
superan_banda_pacf
```

```
## [1] 4
```

```
#Los que superan las bandas del pacf son:
which(abs(pauxacf$acf) > banda)
```

```
## [1]  1  4  8 12
```

Por lo que:

$$p = 1, P = 3$$

Entonces tenemos un modelo

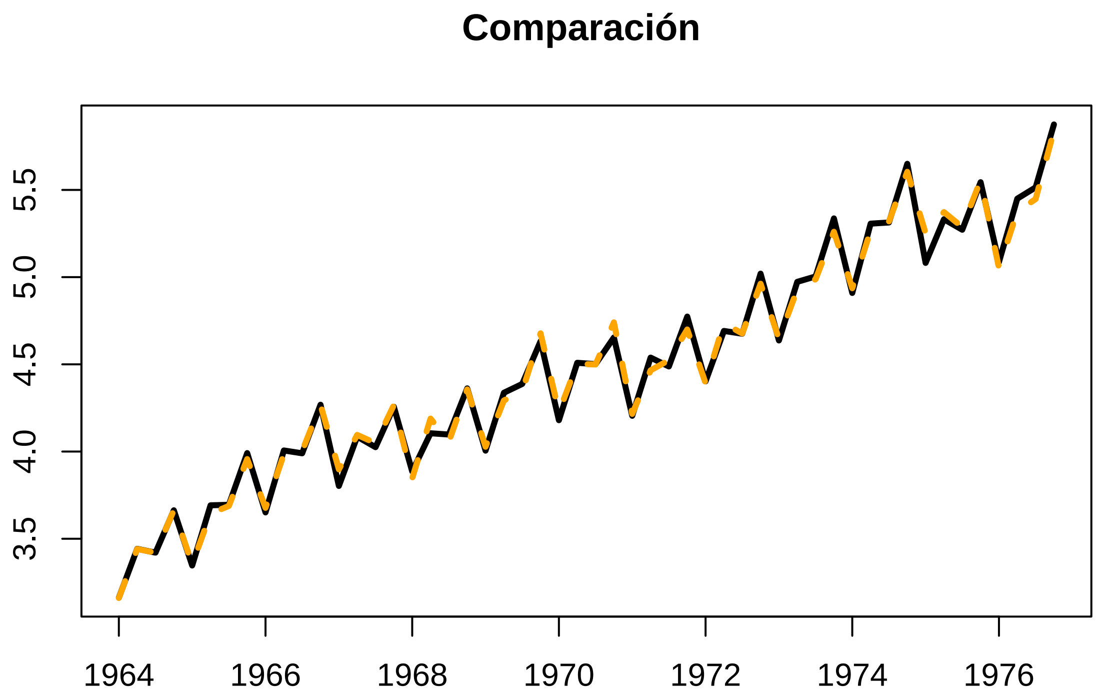
$$SARIMA(1,1,1) \times (3,1,2)_{[4]}$$

Ajustémoslo:

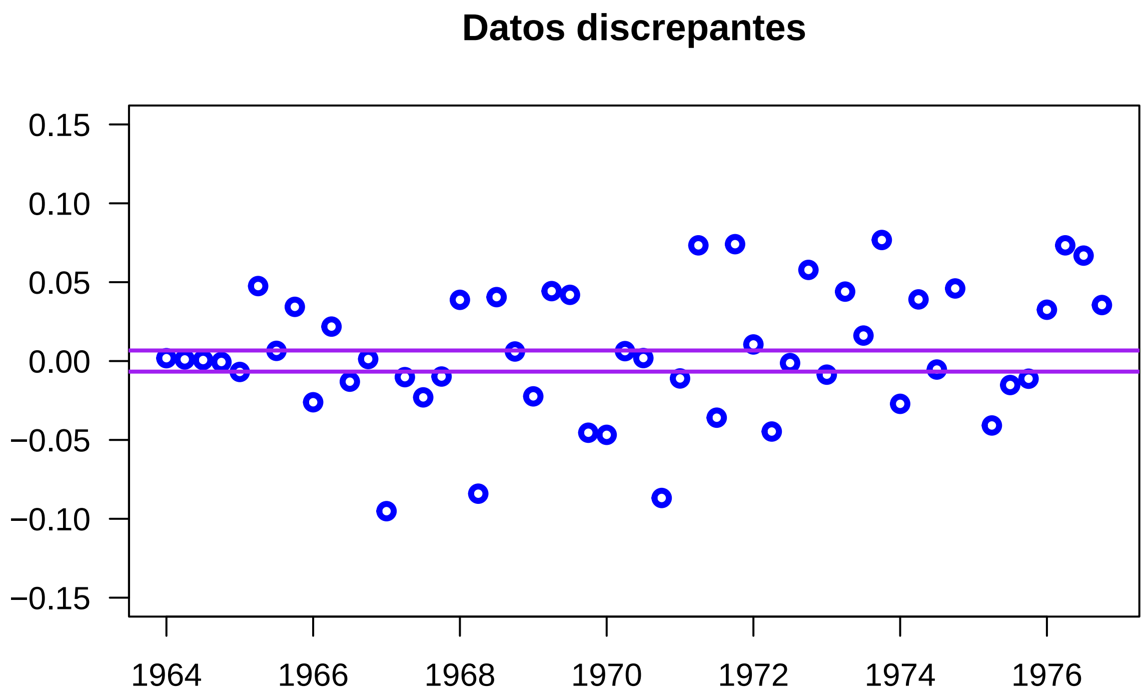
```
SARIMA2<-arima(Serie_sq,order=c(1,1,1),seasonal=list(order=c(3,1,2),period=4), include.mean=F)
SARIMA2
```

```
##
## Call:
## arima(x = Serie_sq, order = c(1, 1, 1), seasonal = list(order = c(3, 1, 2),
##     period = 4), include.mean = F)
##
## Coefficients:
##          ar1      ma1     sar1     sar2     sar3    sma1     sma2
##       0.5047  -0.2597  -0.9560  -0.5185  -0.3367  0.5136  -0.4861
## s.e.  0.3125   0.3233   0.2893   0.3190   0.2335  0.4041   0.2919
##
## sigma^2 estimated as 0.002429:  log likelihood = 71.29,  aic = -128.59
```

```
SARIMA_ajuste2 <- Serie_sq - residuals(SARIMA2)
ts.plot(Serie_sq, lwd=3, main="Comparación ", ylab="", xlab="")
points(SARIMA_ajuste2, type = "l", col ="orange", lty = 2, lwd=3)
```




```
###Y ahora los residuales
plot(SARIMA2$residuals, type="p", col="blue", ylim=c(-.15,.15), ylab="", xlab="", main="Datos discrepant
abline(h=3*(var(SARIMA2$residuals)), col="purple", lwd=2)
abline(h=-3*(var(SARIMA2$residuals)), col="purple",lwd=2)
```



Por el AIC y BIC

```
AIC<-c(SARIMA$aic,ARIMA$aic,modelo_automatico$aic, SARIMA2$aic)
BIC<-c(BIC(SARIMA),BIC(ARIMA),BIC(modelo_automatico), BIC(SARIMA2))
loglik<-c(SARIMA$loglik,ARIMA$loglik,modelo_automatico$loglik, SARIMA2$loglik)
Comparar<-data.frame('AIC'=AIC,'BIC'=BIC,'Loglik'=loglik,row.names = c('SARIMA','ARIMA','Automatico', 'S
Comparar
```

```
##                   AIC        BIC   Loglik
## SARIMA     -127.0795 -113.85233 68.53977
## ARIMA      -112.1905  -71.95005 75.09525
## Automatico -131.2738 -123.72339 67.63692
## SARIMA2    -128.5861 -111.78489 71.29303
```

Comparemos los errores:

```
comparar_=cbind("ARIMA",ARIMA$aic,BIC(ARIMA), mean(ARIMA$residuals),
                mean(abs(ARIMA$residuals)),sqrt(mean((ARIMA$residuals)^2)),
                length(ARIMA$coef))

comparar_2=cbind("SARIMA",SARIMA$aic,BIC(SARIMA), mean(SARIMA$residuals),
                 mean(abs(SARIMA$residuals)),sqrt(mean((SARIMA$residuals)^2)),
                 length(SARIMA$coef))

comparar_3=cbind("SARIMA AUTOMÁTICO",modelo_automatico$aic,BIC(modelo_automatico), mean(modelo_automatic
                 mean(abs(modelo_automatico$residuals)),sqrt(mean((modelo_automatico$residuals)^2)),
                 length(modelo_automatico$coef))
comparar_4=cbind("SARIMA2",SARIMA2$aic,BIC(SARIMA2), mean(SARIMA2$residuals),
```

```r
                 mean(abs(SARIMA2$residuals)),sqrt(mean((SARIMA2$residuals)^2)),
                 length(SARIMA2$coef))
nombres=cbind("AJUSTE", "AIC", "BIC","ME","MAE", "RMSE","#Paramatros")

resultados<-rbind(comparar_,comparar_2,comparar_3, comparar_4)
resultados<-as.table(resultados)
colnames(resultados)=c("AJUSTE", "AIC", "BIC","ME","MAE", "RMSE","#Parametros")
rownames(resultados)=c("","", "","")

(resultados)
```

```
##  AJUSTE           AIC               BIC               ME
##  ARIMA            -112.190505919999 -71.950045811436  0.00113404901103327
##  SARIMA           -127.079539350415 -113.852333284968 0.00480342160710416
##  SARIMA AUTOMÁTICO -131.273837494611 -123.723394689481 0.000394350469091238
##  SARIMA2          -128.586067490355 -111.784886676675 0.00191986834666404
##  MAE                RMSE               #Parametros
##  0.0313590810867887 0.0398103705554394 19
##  0.0408775073139969 0.0524452560585087 5
##  0.0406078930009833 0.0531376953726595 2
##  0.0342715205550597 0.0468668451395007 7
```

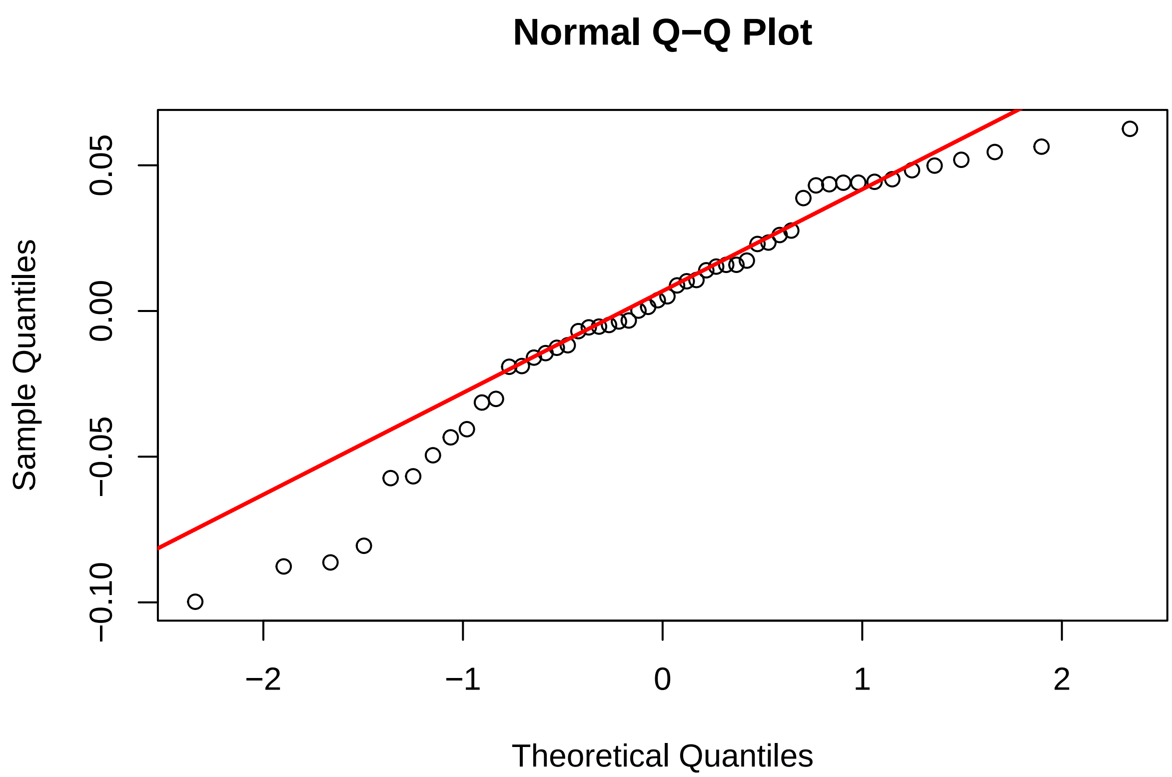
Parece que el SARIMA ajustado de manera automática es quien mejores indicadores tiene, después sería el segundo SARIMA que ajustamos nosotros, después el primer SARIMA y finalmente, el ARIMA

Hagamos la comprobación de supuestos:

Normalidad

ARIMA

```r
#ARIMA
qqnorm(ARIMA$residuals)
qqline(ARIMA$residuals, col="red", lwd=2)
```




```r
#Prueba Anderson-Darling
ad.test(ARIMA$residuals)
```

```
##
##  Anderson-Darling normality test
##
## data:  ARIMA$residuals
## A = 0.74525, p-value = 0.04904
```

```r
#Prueba de Shapiro
shapiro.test(ARIMA$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  ARIMA$residuals
## W = 0.94576, p-value = 0.01933
```

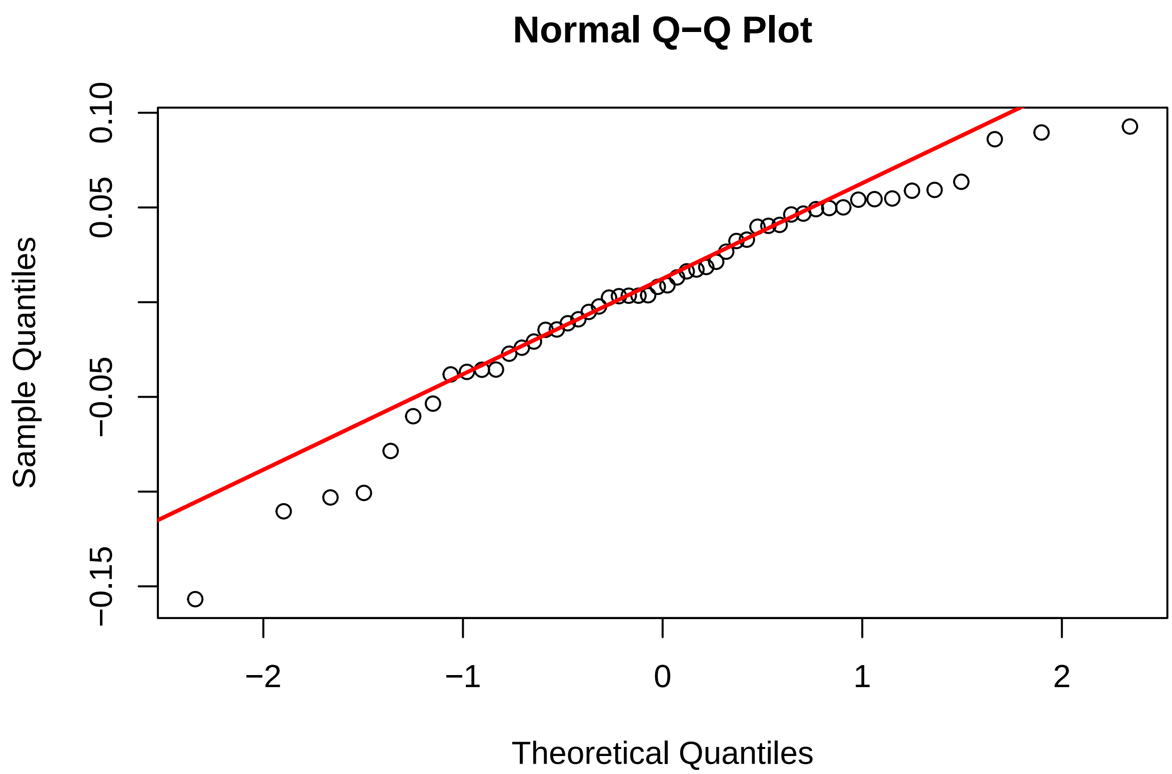
```r
#Jarque-Bera Test
jarque.bera.test(ARIMA$residuals)
```

```
##
##  Jarque Bera Test
##
## data:  ARIMA$residuals
## X-squared = 3.9831, df = 2, p-value = 0.1365
```

Solo pasa Jarque-Bera

SARIMA

```r
#SARIMA
qqnorm(SARIMA$residuals)
qqline(SARIMA$residuals, col="red", lwd=2)
```




```r
#Prueba Anderson-Darling
ad.test(SARIMA$residuals)
```

```
##
##  Anderson-Darling normality test
##
## data:  SARIMA$residuals
## A = 0.74048, p-value = 0.0504
```

```r
#Prueba de Shapiro
shapiro.test(SARIMA$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  SARIMA$residuals
## W = 0.94837, p-value = 0.02487
```

```r
#Jarque-Bera Test. tseries
jarque.bera.test(SARIMA$residuals)
```

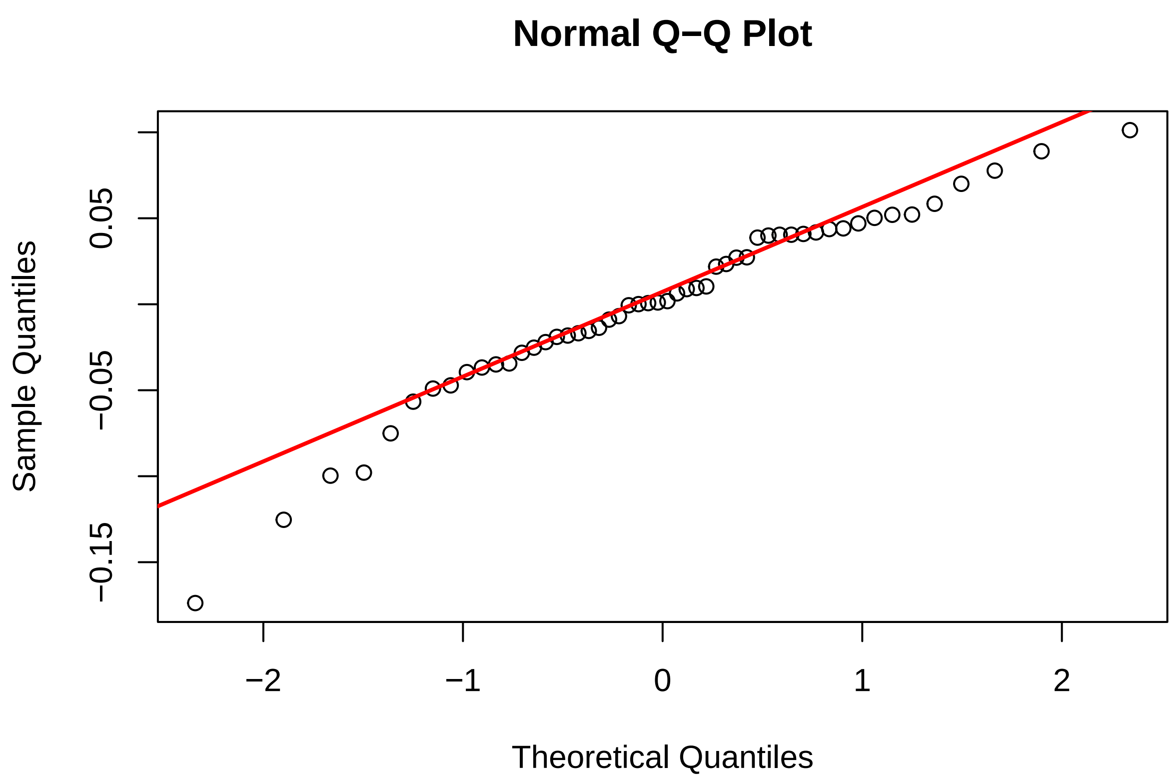
```
##
##  Jarque Bera Test
```

```
##
## data:  SARIMA$residuals
## X-squared = 7.3243, df = 2, p-value = 0.02568
```

Solo pasa Anderson-Darling

Apenas pasamos Anderson-Darling ### SARIMA AUTOMÁTICO

```
#SARIMA AUTOMÁTICO
qqnorm(modelo_automatico$residuals)
qqline(modelo_automatico$residuals, col="red", lwd=2)
```




```
#Prueba Anderson-Darling
ad.test(modelo_automatico$residuals)
```

```
##
##  Anderson-Darling normality test
##
## data:  modelo_automatico$residuals
## A = 0.61243, p-value = 0.1055
```

```
#Prueba de Shapiro
shapiro.test(modelo_automatico$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  modelo_automatico$residuals
## W = 0.95524, p-value = 0.04873
```

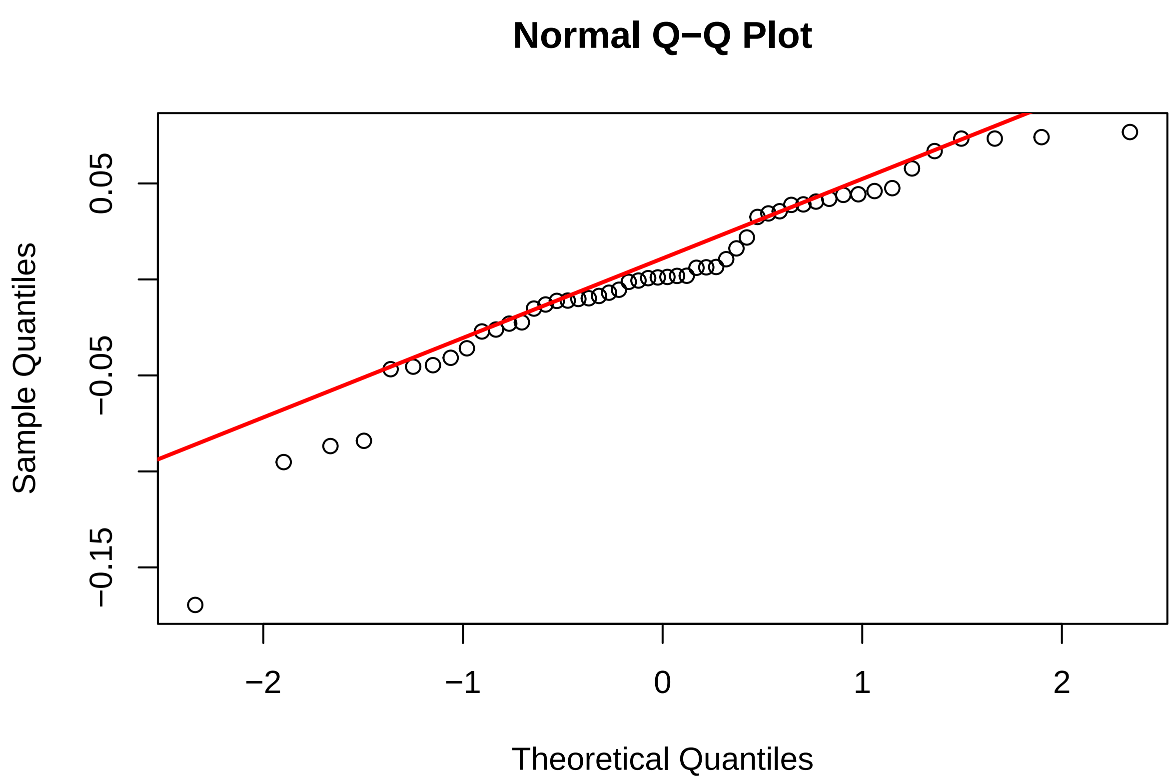
```r
#Jarque-Bera Test. tseries
jarque.bera.test(modelo_automatico$residuals)
```

```
##
##  Jarque Bera Test
##
## data:  modelo_automatico$residuals
## X-squared = 8.8165, df = 2, p-value = 0.01218
```

Pasa Anderson-Darling con un p-value no tan cerca de 00.05, pero las demás pruebas las rechaza

SARIMA2

```r
#SARIMA2
qqnorm(SARIMA2$residuals)
qqline(SARIMA2$residuals, col="red", lwd=2)
```




```r
#Prueba Anderson-Darling
ad.test(SARIMA2$residuals)
```

```
##
##  Anderson-Darling normality test
##
## data:  SARIMA2$residuals
## A = 0.76981, p-value = 0.04256
```

```r
#Prueba de Shapiro
shapiro.test(SARIMA2$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  SARIMA2$residuals
## W = 0.93269, p-value = 0.005712
```

```r
#Jarque-Bera Test. tseries
jarque.bera.test(SARIMA2$residuals)
```

```
##
##  Jarque Bera Test
##
## data:  SARIMA2$residuals
## X-squared = 17.165, df = 2, p-value = 0.0001874
```

No pasa ningún test.

El modelo ajustado por auto.arima tiene el mejor ajuste acorde a las pruebas

Varianza constante

ARIMA

```r
#ARIMA
Y <- as.numeric(ARIMA$residuals)
X <- 1:length(ARIMA$residuals)
bptest(Y ~ X)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  Y ~ X
## BP = 1.2041, df = 1, p-value = 0.2725
```

Lo pasa

SARIMA

```r
#SARIMA
Y <- as.numeric(SARIMA$residuals)
X <- 1:length(SARIMA$residuals)
bptest(Y ~ X)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  Y ~ X
## BP = 1.6613, df = 1, p-value = 0.1974
```

Lo pasa

SARIMA AUTOMÁTICO

```r
#SARIMA AUTOMÁTICO
Y <- as.numeric(modelo_automatico$residuals)
X <- 1:length(modelo_automatico$residuals)
bptest(Y ~ X)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  Y ~ X
## BP = 1.8803, df = 1, p-value = 0.1703
```

Lo pasa

SARIMA2

```r
#SARIMA2
Y <- as.numeric(SARIMA2$residuals)
X <- 1:length(SARIMA2$residuals)
bptest(Y ~ X)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  Y ~ X
## BP = 2.3289, df = 1, p-value = 0.127
```

Lo pasa

Todos pasan las pruebas; el mayor p-value lo obtuvo el ARIMA

Media cero

ARIMA

```r
t.test(ARIMA$residuals,mu=0)
```

```
##
##  One Sample t-test
##
## data:  ARIMA$residuals
## t = 0.20352, df = 51, p-value = 0.8395
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.01005282  0.01232092
## sample estimates:
##   mean of x
## 0.001134049
```

Lo pasa

SARIMA

```r
t.test(SARIMA$residuals,mu=0)
```

```
##
##  One Sample t-test
```

```
##
## data:  SARIMA$residuals
## t = 0.65684, df = 51, p-value = 0.5142
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.009877914  0.019484757
## sample estimates:
##   mean of x
## 0.004803422
```

Lo pasa

SARIMA AUTOMÁTICO

```
t.test(modelo_automatico$residuals,mu=0)
```

```
##
##  One Sample t-test
##
## data:  modelo_automatico$residuals
## t = 0.053, df = 51, p-value = 0.9579
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.0145432  0.0153319
## sample estimates:
##    mean of x
## 0.0003943505
```

Lo pasa

SARIMA2

```
t.test(SARIMA2$residuals,mu=0)
```

```
##
##  One Sample t-test
##
## data:  SARIMA2$residuals
## t = 0.29279, df = 51, p-value = 0.7709
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.01124418  0.01508392
## sample estimates:
##   mean of x
## 0.001919868
```

Lo pasa

Todos pasan esta prueba

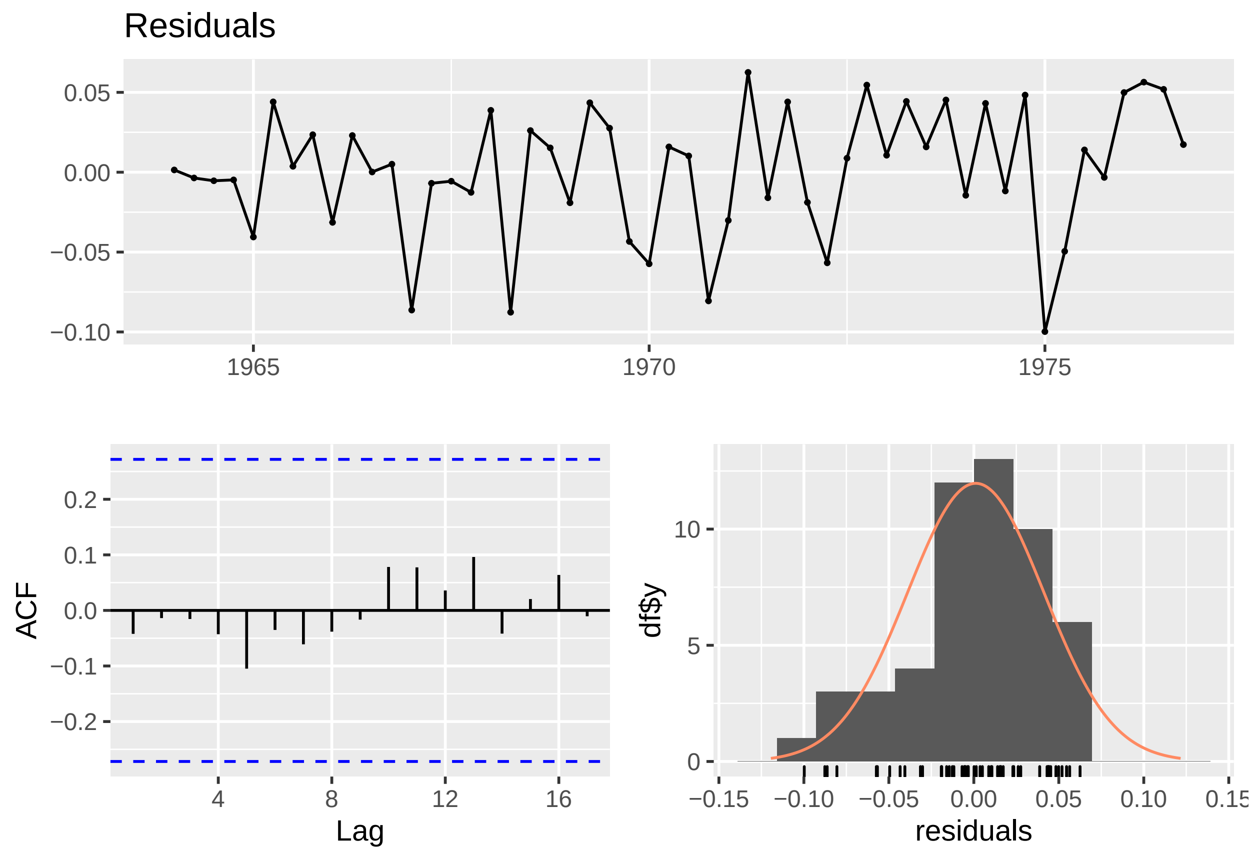
Residuales no correlacionados

ARIMA

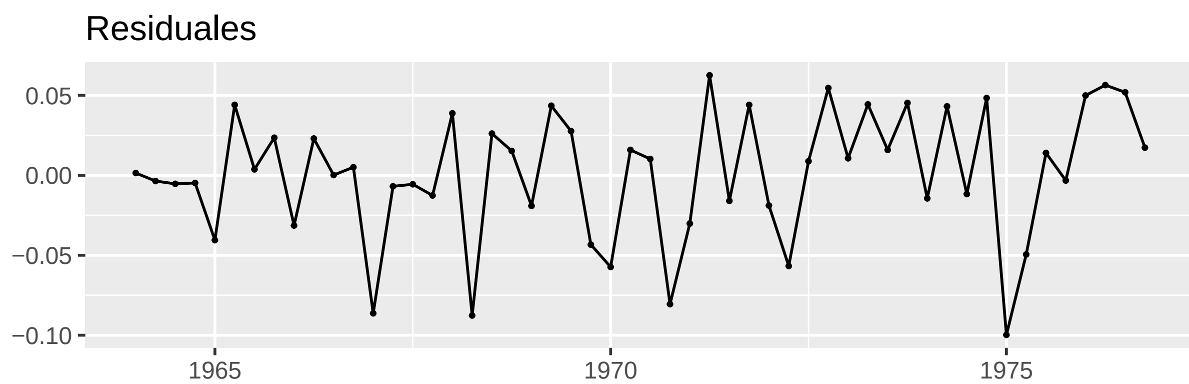
```
checkresiduals(ARIMA$residuals)
```

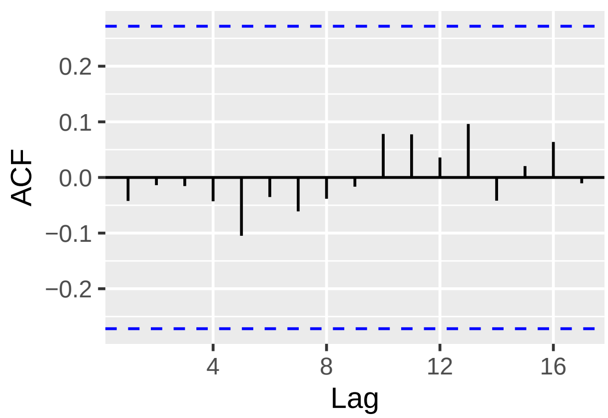
```
## Warning in modeldf.default(object): Could not find appropriate degrees of
```

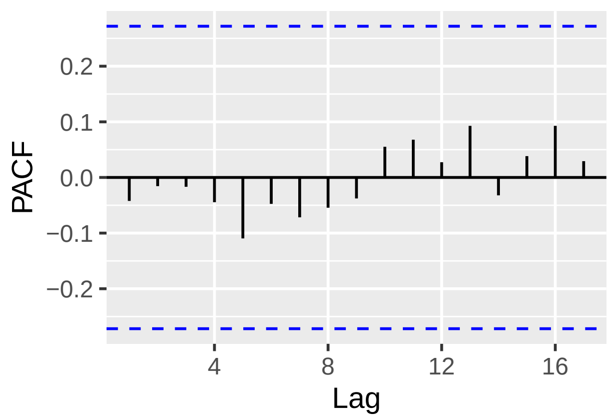
```
## freedom for this model.
```




```
ggtsdisplay(ARIMA$residuals,main="Residuales")
```



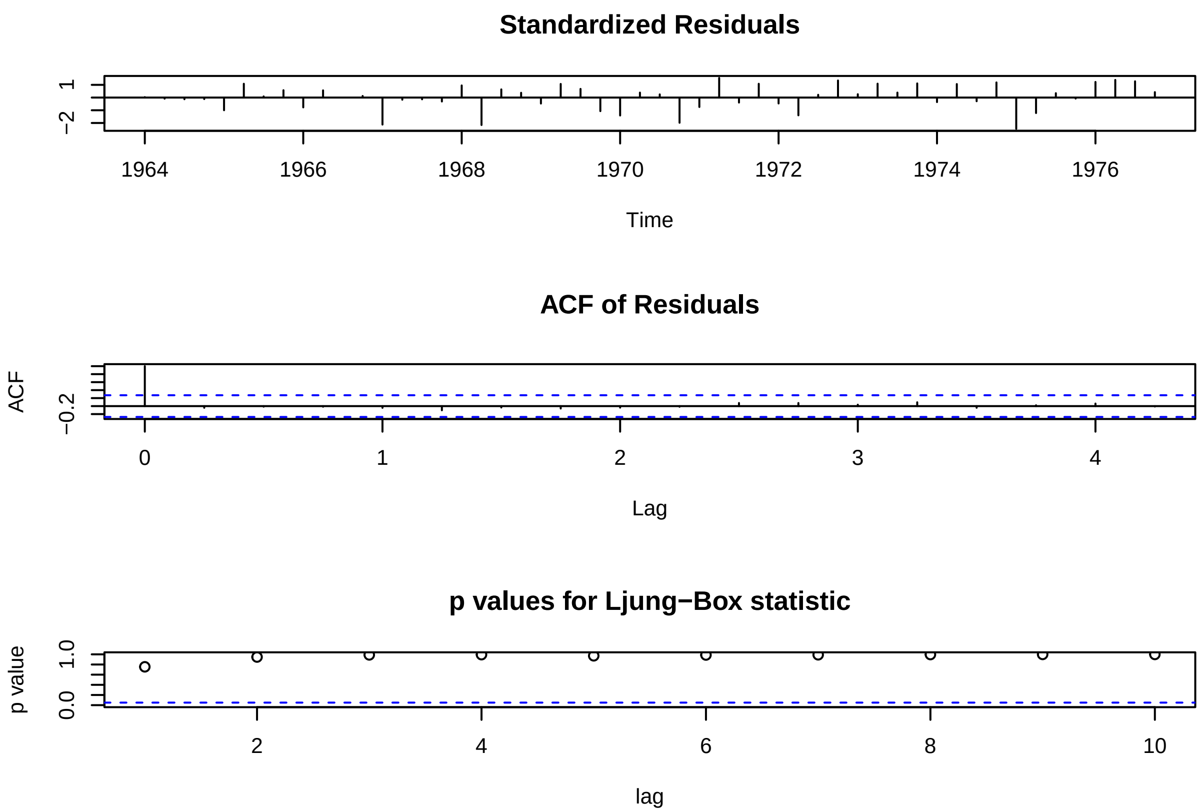





```
#No se sale en ningún lag

#Box-Pierce test contrasta
# Ho: Independencia vs. H1: Dependencia
Box.test(ARIMA$residuals, lag =10) #
```

```
##
##  Box-Pierce test
##
## data:  ARIMA$residuals
## X-squared = 1.4466, df = 10, p-value = 0.9991
```

```
#Por lo que no están relacionados de manera conjunta

#De manera conjunta, con la prueba de Ljung y Box
#HO:No estan correlacionados de manera conjunta
# vs
#H1:Estan correlacionados de manera conjunta
tsdiag(ARIMA)
```

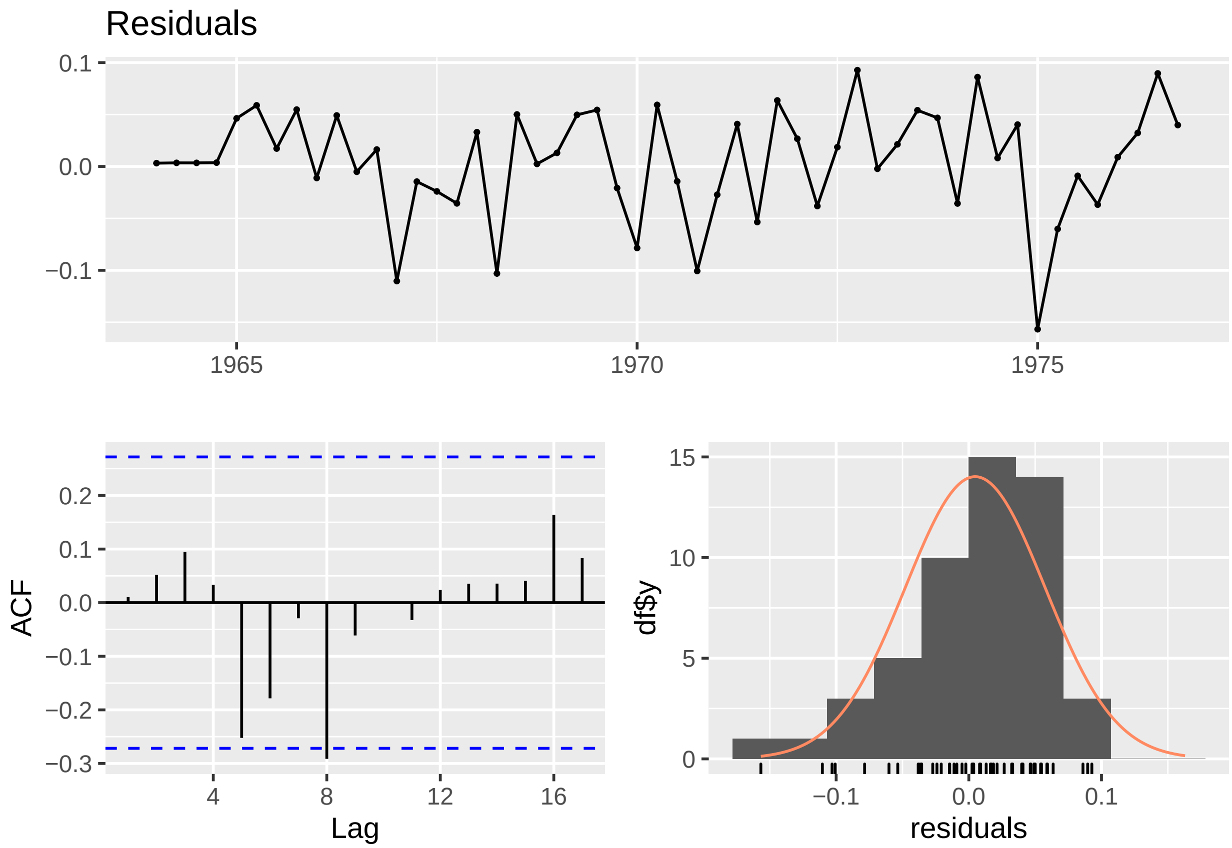


Pasa las pruebas

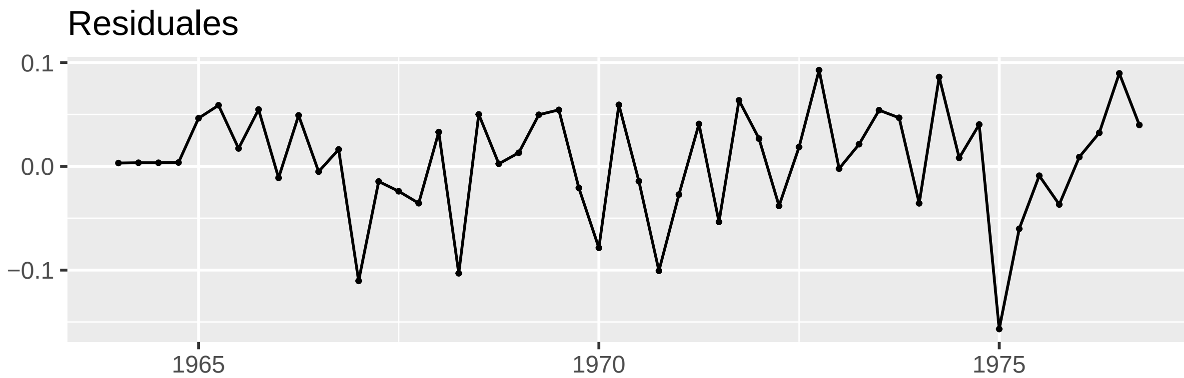
SARIMA

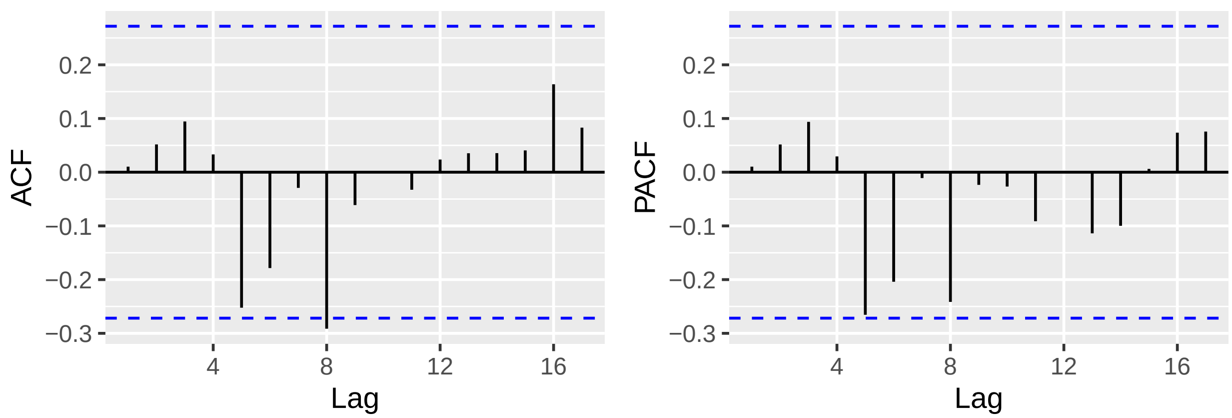
```
checkresiduals(SARIMA$residuals)
```

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```




```r
ggtsdisplay(SARIMA$residuals,main="Residuales")
```

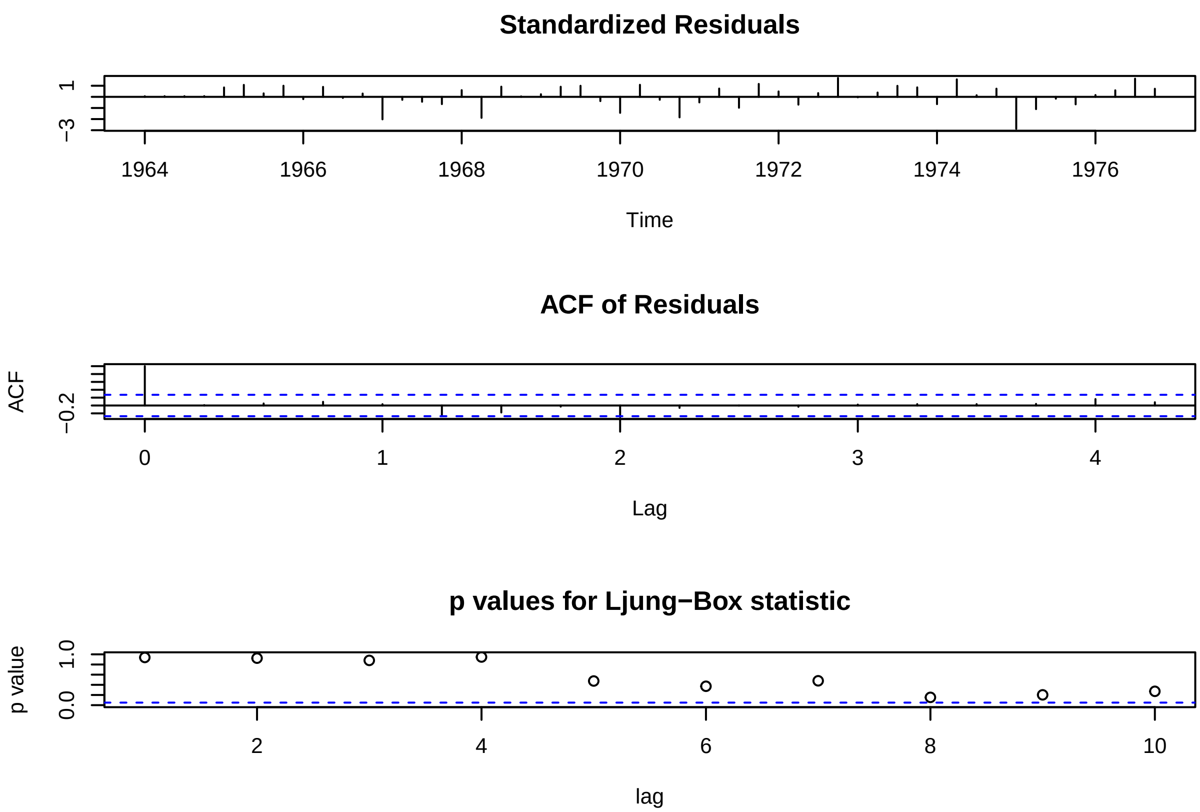





```
#Se sale en un lag cerca de 8, por poco.
#Box-Pierce test contrasta
# Ho: Independencia vs. H1: Dependencia
Box.test(SARIMA$residuals, lag =10)
```

```
##
##  Box-Pierce test
##
## data:  SARIMA$residuals
## X-squared = 10.287, df = 10, p-value = 0.4157
```

```
#De manera conjunta, con la prueba de Ljung y Box
#HO:No estan correlacionados de manera conjunta
# vs
#H1:Estan correlacionados de manera conjunta
tsdiag(SARIMA)
```

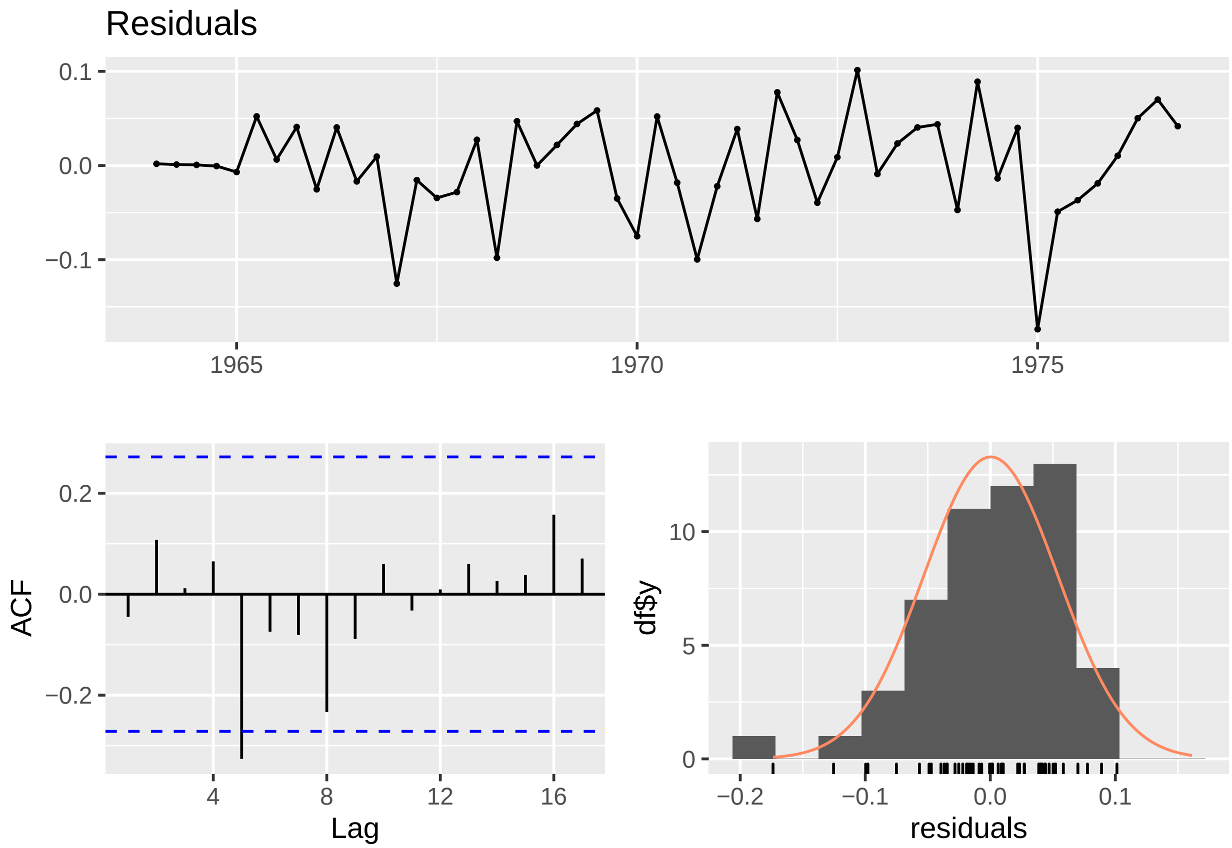


Pasa las pruebas

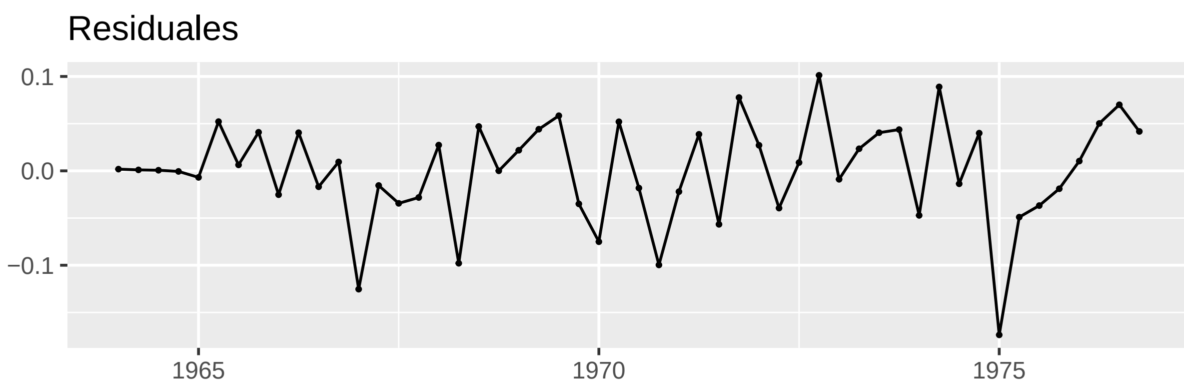
SARIMA AUTOMÁTICO

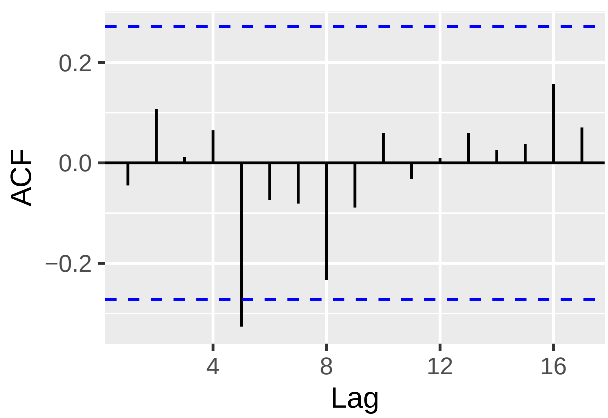
```
checkresiduals(modelo_automatico$residuals)
```

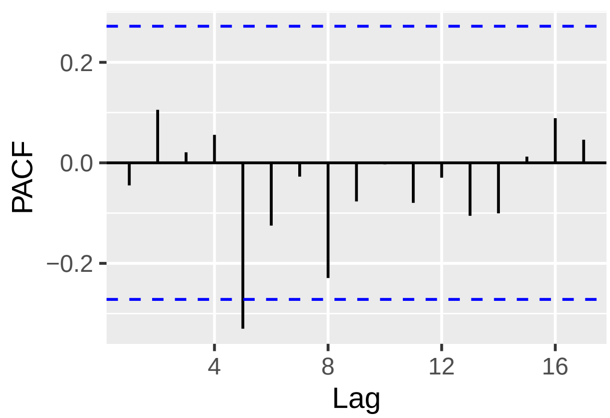
```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```




```
ggtsdisplay(modelo_automatico$residuals,main="Residuales")
```



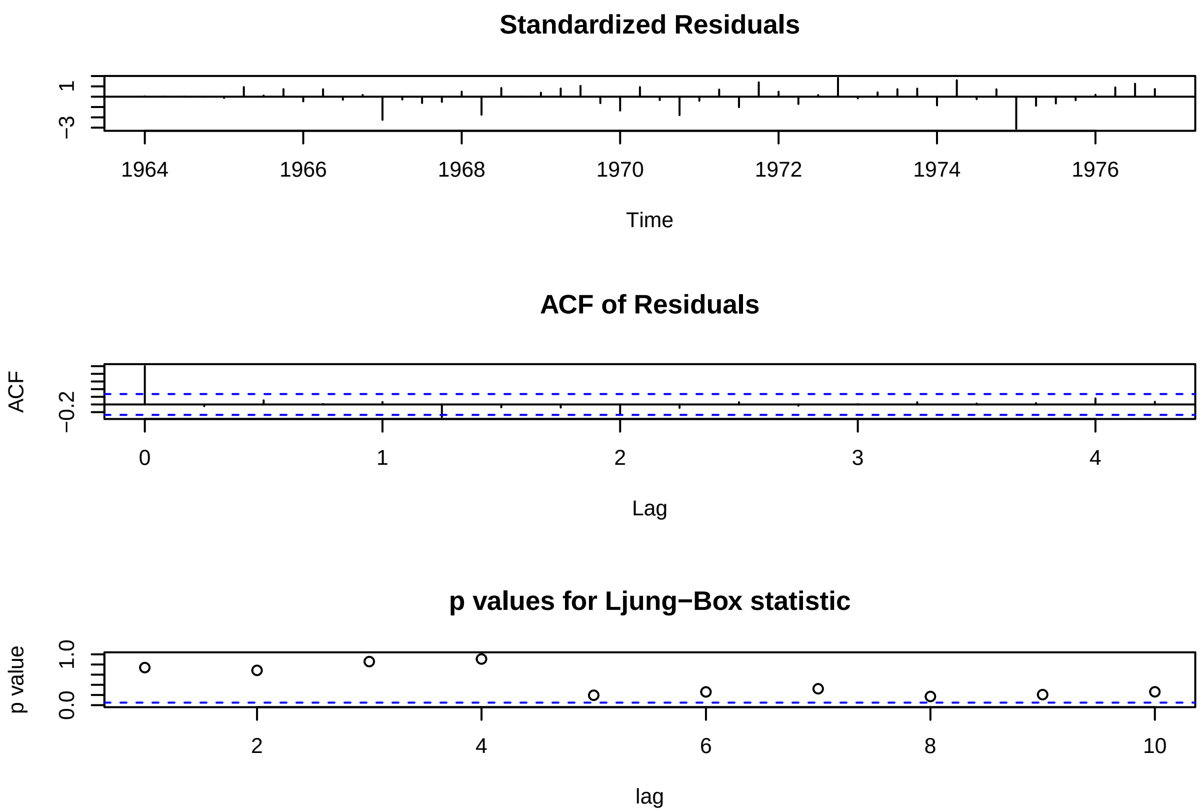





```r
#Se sale cerca del lag 5
#Box-Pierce test contrasta
# Ho: Independencia vs. H1: Dependencia
Box.test(modelo_automatico$residuals, lag =10)
```

```
##
##  Box-Pierce test
##
## data:  modelo_automatico$residuals
## X-squared = 10.52, df = 10, p-value = 0.3962
```

```r
#De manera conjunta, con la prueba de Ljung y Box
#H0:No estan correlacionados de manera conjunta
# vs
#H1:Estan correlacionados de manera conjunta
tsdiag(modelo_automatico)
```

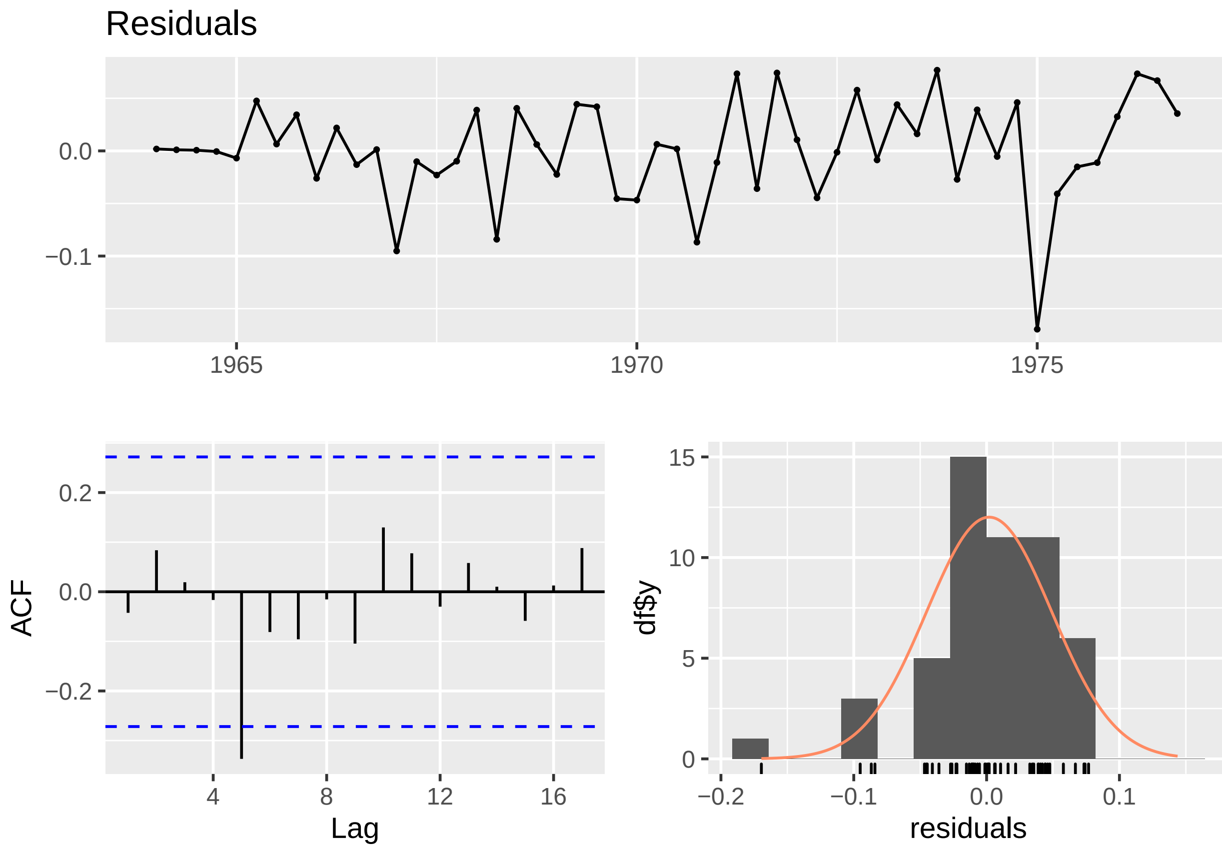


Pasa las pruebas

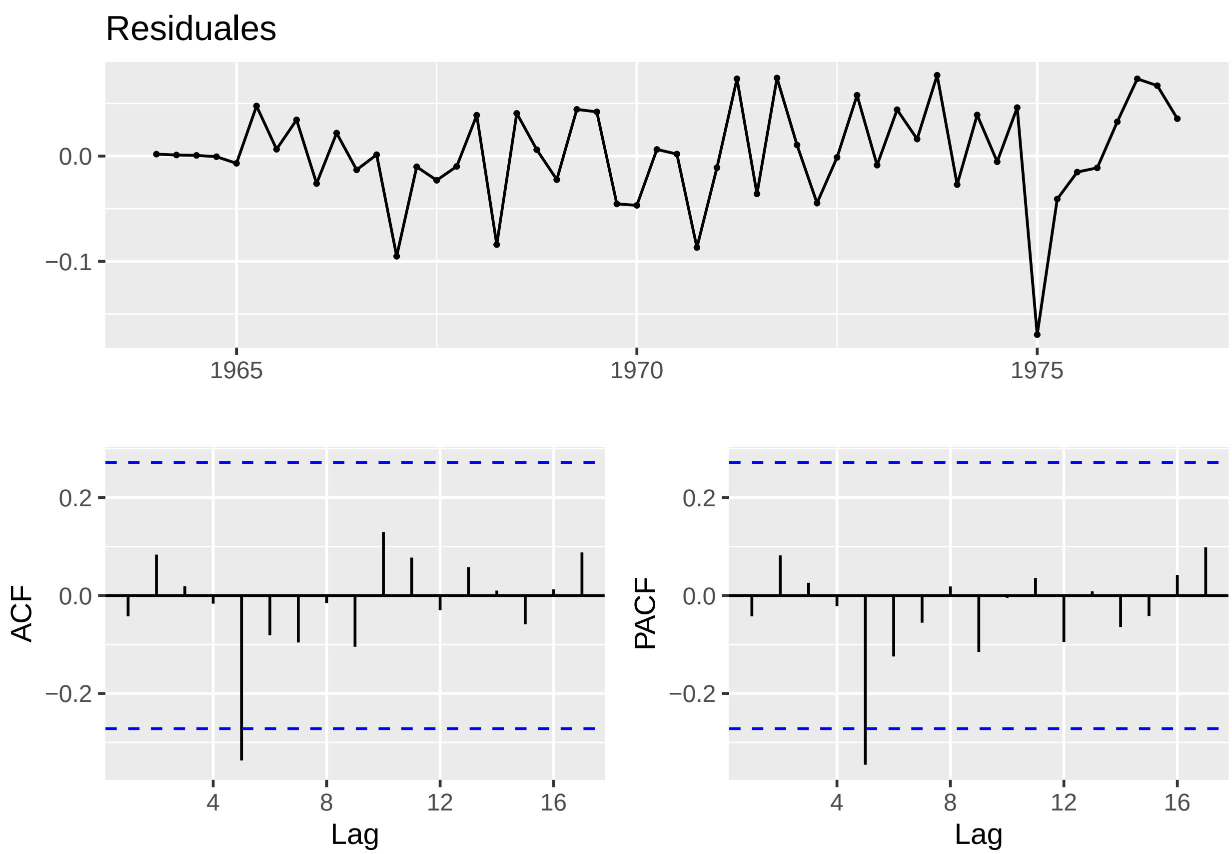
SARIMA2

```
checkresiduals(SARIMA2$residuals)
```

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```



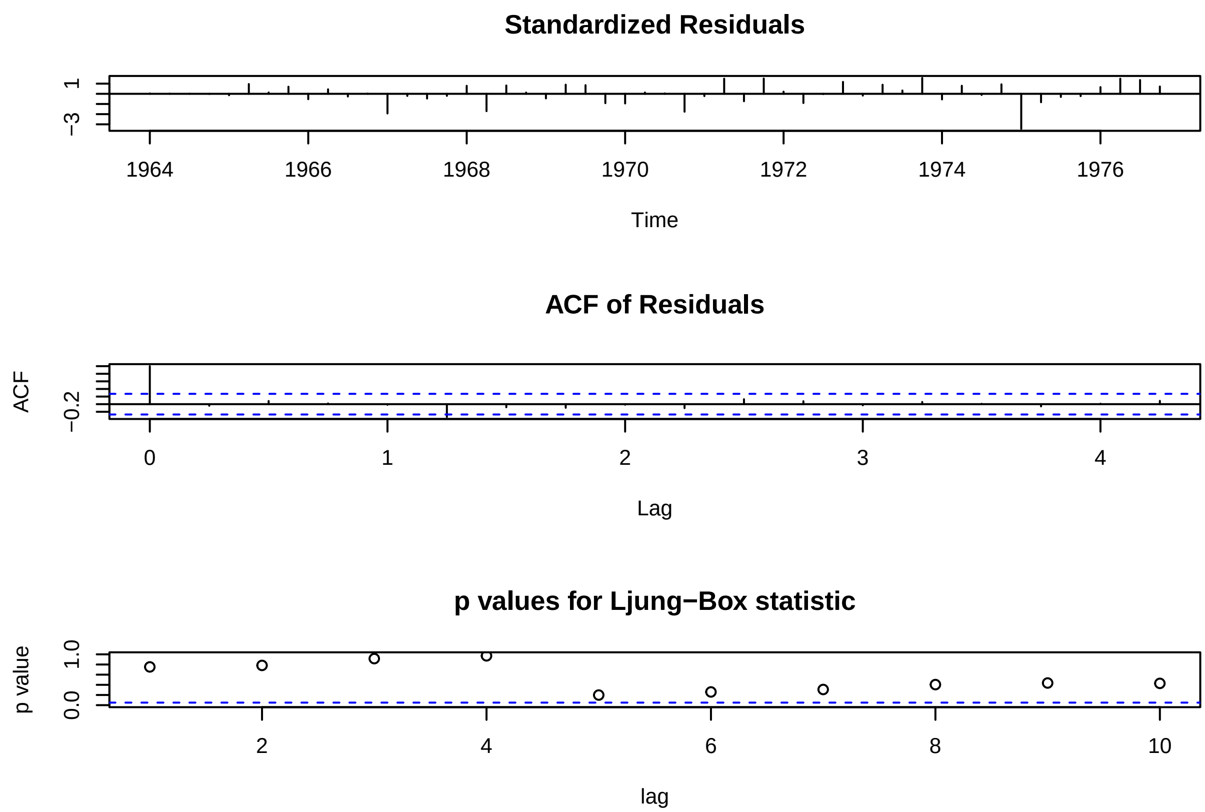

```
ggtsdisplay(SARIMA2$residuals,main="Residuales")
```




```r
#Se sale en el lag 5;
#Box-Pierce test contrasta
# Ho: Independencia vs. H1: Dependencia
Box.test(SARIMA2$residuals, lag =10)
```

```
##
##  Box-Pierce test
##
## data:  SARIMA2$residuals
## X-squared = 8.6676, df = 10, p-value = 0.5639
```

```r
#De manera conjunta, con la prueba de Ljung y Box
#H0:No estan correlacionados de manera conjunta
# vs
#H1:Estan correlacionados de manera conjunta
tsdiag(SARIMA2)
```



Pasa los test

Significancia de los coeficientes

ARIMA

```
ARIMA_int<-confint(ARIMA)
ARIMA_int
```

```
##              2.5 %       97.5 %
## ar1  -1.00862641 -0.980272709
## ar2  -1.00769140 -0.978709545
## ar3  -1.00209914 -0.995346337
## ma1  -0.26602909  0.449156386
## ma2  -0.08125679  0.782926858
## ma3  -0.29018023  0.596448780
## ma4  -1.04713433 -0.001366895
## ma5  -0.80658372  0.194540758
## ma6  -0.60300938  0.098162739
## ma7  -0.67709203  0.110569510
## ma8  -1.21515818 -0.115601890
## ma9  -0.79038082  0.474220236
## ma10 -0.68907677  0.438275283
## ma11 -0.41681849  0.520709182
## ma12  0.05951077  1.012705480
## ma13 -0.09375056  0.814269836
## ma14 -0.30496484  0.515268122
```

```
## ma15 -0.48960995  0.369774301
## ma16 -0.41846529  0.544986781
```

```r
k=length(confint(ARIMA))/2
no_sign<-c()
for(i in 1:k){
  no_sign[i]<-(ARIMA_int[i]<0 & ARIMA_int[i+k]>0)
}
#No significativos
sum(no_sign)
```

```
## [1] 13
```

```r
#Porcentaje no significativos
sum(no_sign)/k
```

```
## [1] 0.6842105
```

SARIMA

```r
SARIMA_int<-confint(SARIMA)
SARIMA_int
```

```
##           2.5 %     97.5 %
## ar1  -1.0011218  1.1569074
## ar2  -0.1756916  2.0093433
## ma1   0.2387241  2.0433917
## ma2  -0.1281688  0.7236648
## sma1 -1.1870104 -0.4232753
```

```r
k=length(confint(SARIMA))/2
no_sign<-c()
for(i in 1:k){
  no_sign[i]<-(SARIMA_int[i]<0 & SARIMA_int[i+k]>0)
}
#No significativos
sum(no_sign)
```

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

```r
#Porcentaje no significativos
sum(no_sign)/k
```

```
## [1] 0.6
```

SARIMA AUTOMÁTICO

```r
SARIMA_DRIFT_int<-confint(modelo_automatico)
SARIMA_DRIFT_int
```

```
##           2.5 %     97.5 %
## ar1  -0.0195618  0.5560452
## sma1 -0.9272078 -0.4205821
```

```r
k=length(confint(modelo_automatico))/2
no_sign<-c()
for(i in 1:k){
  no_sign[i]<-(SARIMA_DRIFT_int[i]<0 & SARIMA_DRIFT_int[i+k]>0)
```

```
}
#No significativos
sum(no_sign)
```

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

```
#Porcentaje no significativos
sum(no_sign)/k
```

```
## [1] 0.5
```

SARIMA2

```
SARIMA2_int<-confint(SARIMA2)
SARIMA2_int
```

```
##           2.5 %      97.5 %
## ar1  -0.1077112  1.11718746
## ma1  -0.8933044  0.37395118
## sar1 -1.5230961 -0.38888560
## sar2 -1.1437314  0.10671376
## sar3 -0.7943175  0.12091960
## sma1 -0.2783978  1.30557906
## sma2 -1.0582232  0.08599028
```

```
k=length(confint(SARIMA2))/2
no_sign<-c()
for(i in 1:k){
  no_sign[i]<-(SARIMA2_int[i]<0 & SARIMA2_int[i+k]>0)
}
#No significativos
sum(no_sign)
```

```
## [1] 6
```

```
#Porcentaje no significativos
sum(no_sign)/k
```

```
## [1] 0.8571429
```

Por la simplicidad del modelo (Al tener solo 2 coeficientes), ser el que poseeel menor porcentaje de coeficientes no significativos y además, pasar todos los tests (En normalidad pasó al menos Anderson-Darling) y tener los mejores índices, elegimos el ajustado por auto.arima

$$\therefore \textbf{Elegimos el } SARIMA(1,1,0)\times(1,1,0)_{[4]}$$

¿Ajustará mejor si le quitamos el coeficiente no significativo?

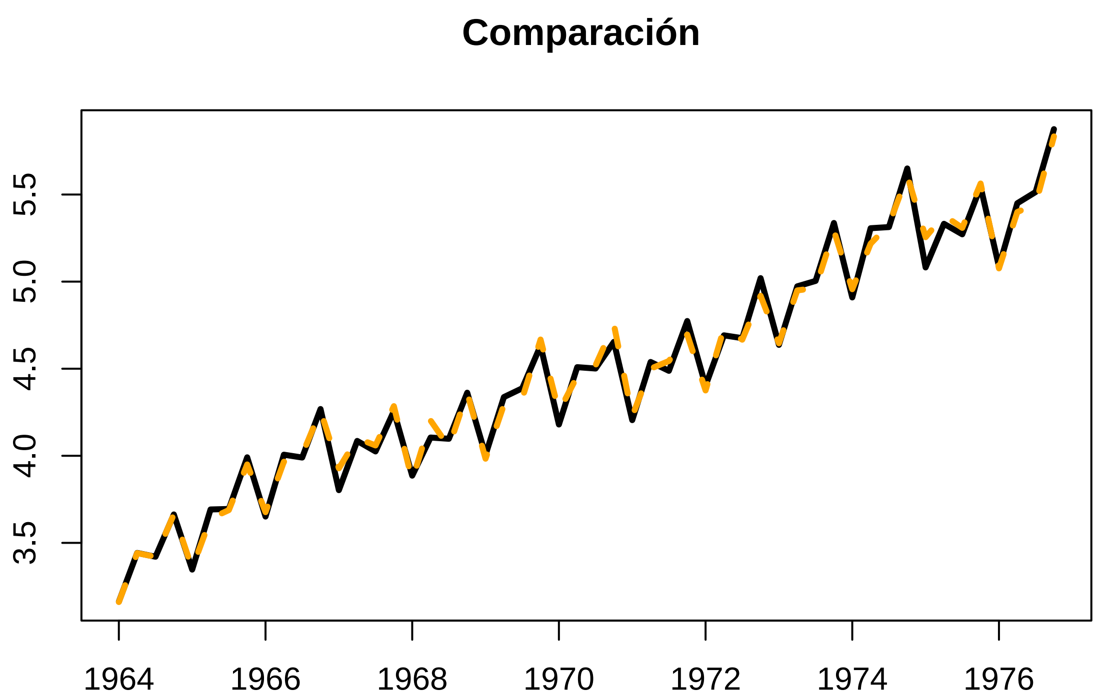
```
modelo_automatico2<-arima(Serie_sq,order=c(0,1,0),seasonal=list(order=c(0,1,1),period=4))
modelo_automatico2
```

```
##
## Call:
## arima(x = Serie_sq, order = c(0, 1, 0), seasonal = list(order = c(0, 1, 1),
##     period = 4))
##
## Coefficients:
```

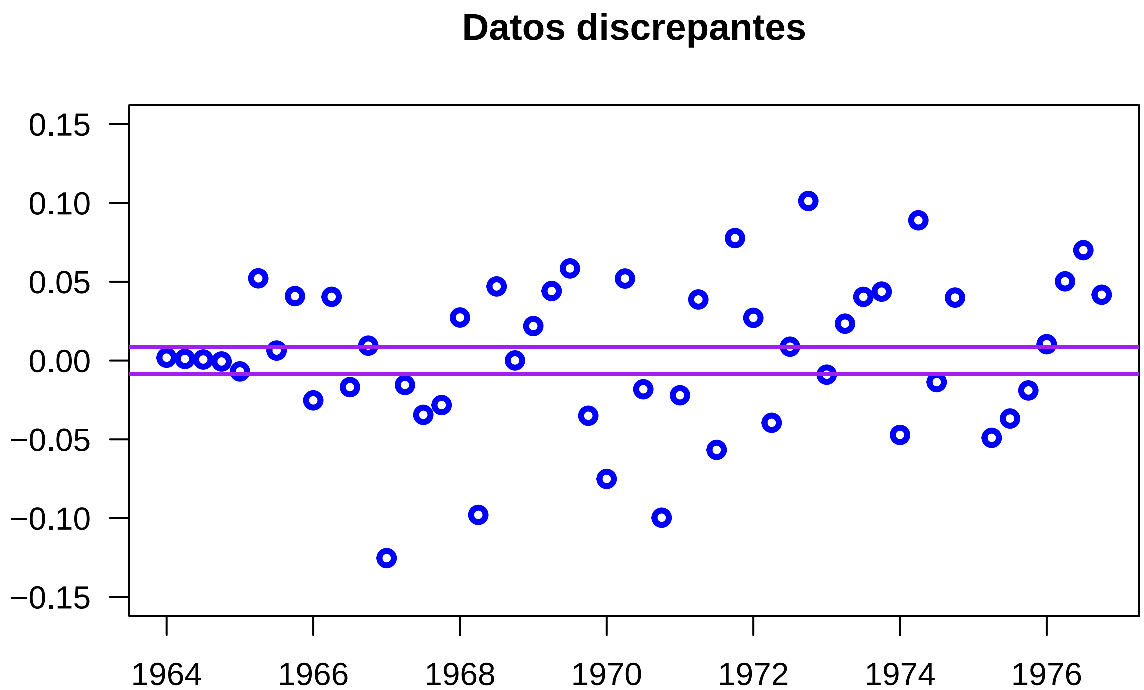
```
##          sma1
##       -0.7373
## s.e.   0.1242
##
## sigma^2 estimated as 0.003296:  log likelihood = 66.04,  aic = -130.09
```

Gráfica

```
AUTOMATICO_ajuste <- Serie_sq - residuals(modelo_automatico)
ts.plot(Serie_sq, lwd=3, main="Comparación ", ylab="", xlab="")
points(AUTOMATICO_ajuste, type = "l", col ="orange", lty = 2, lwd=3)
```

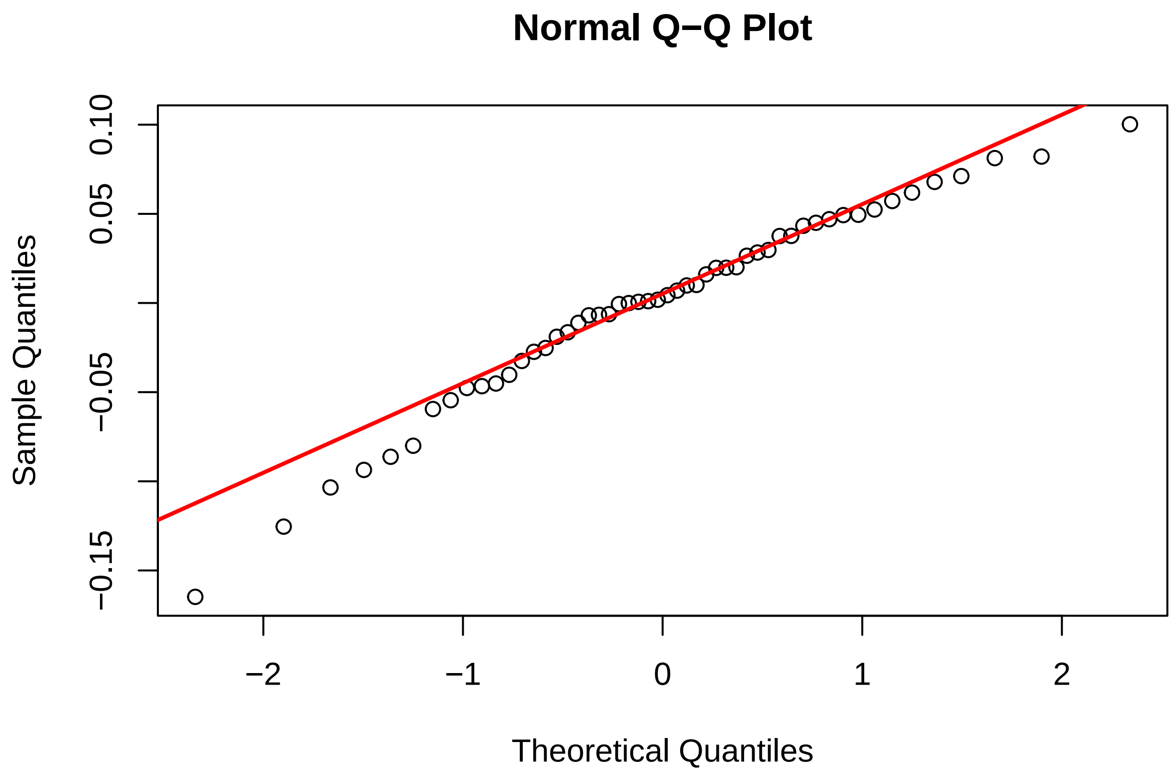



```
###Y ahora los residuales
plot(modelo_automatico$residuals, type="p", col="blue", ylim=c(-.15,.15), ylab="", xlab="", main="Datos
abline(h=3*(var(modelo_automatico$residuals)), col="purple", lwd=2)
abline(h=-3*(var(modelo_automatico$residuals)), col="purple",lwd=2)
```



Normalidad

```
#SARIMA AUTOMÁTICO
qqnorm(modelo_automatico2$residuals)
qqline(modelo_automatico2$residuals, col="red", lwd=2)
```




```r
#Prueba Anderson-Darling
ad.test(modelo_automatico2$residuals)
```

```
##
##  Anderson-Darling normality test
##
## data:  modelo_automatico2$residuals
## A = 0.50391, p-value = 0.1954
```

```r
#Prueba de Shapiro
shapiro.test(modelo_automatico2$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  modelo_automatico2$residuals
## W = 0.9657, p-value = 0.1382
```

```r
#Jarque-Bera Test. tseries
jarque.bera.test(modelo_automatico2$residuals)
```

```
##
##  Jarque Bera Test
##
## data:  modelo_automatico2$residuals
## X-squared = 5.0001, df = 2, p-value = 0.08208
```

Varianza constante

```
#SARIMA AUTOMÁTICO
Y <- as.numeric(modelo_automatico2$residuals)
X <- 1:length(modelo_automatico2$residuals)
bptest(Y ~ X)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  Y ~ X
## BP = 3.2205, df = 1, p-value = 0.07272
```

Media 0

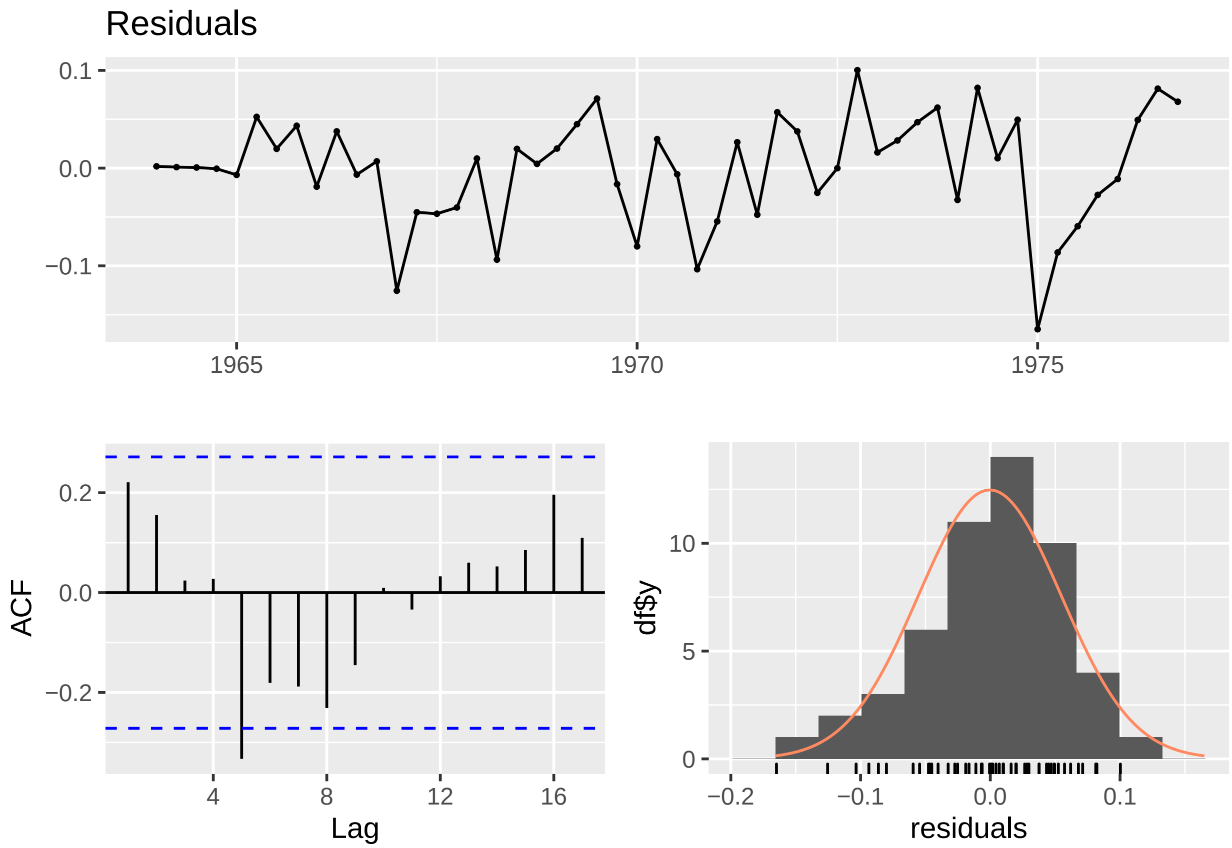
```
t.test(modelo_automatico2$residuals,mu=0)
```

```
##
##  One Sample t-test
##
## data:  modelo_automatico2$residuals
## t = -0.052502, df = 51, p-value = 0.9583
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.01574752  0.01494485
## sample estimates:
##     mean of x
## -0.0004013333
```

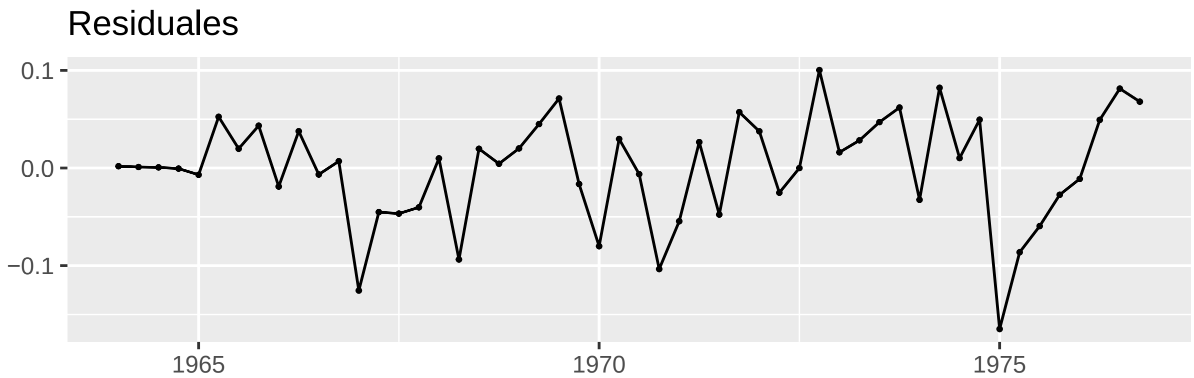
Residuales no correlacionados

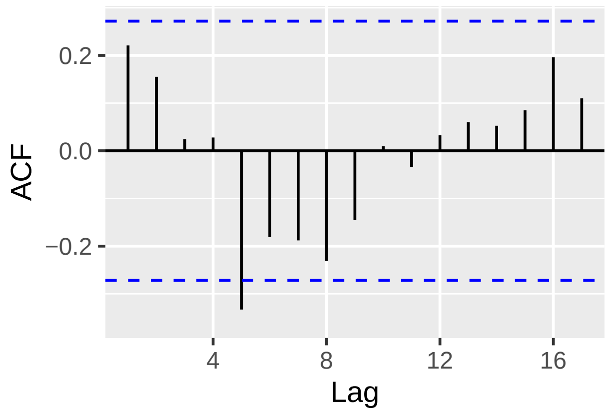
```
checkresiduals(modelo_automatico2$residuals)
```

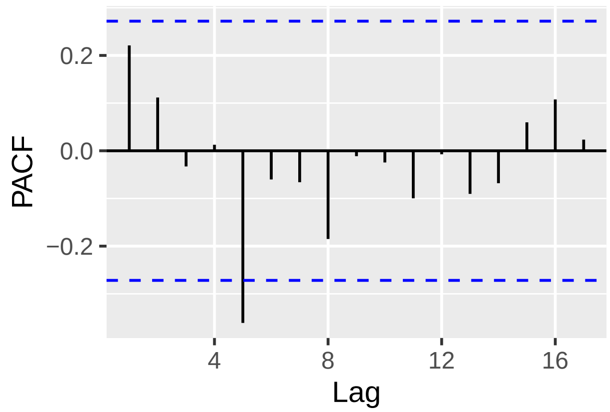
```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```




```
ggtsdisplay(modelo_automatico2$residuals,main="Residuales")
```



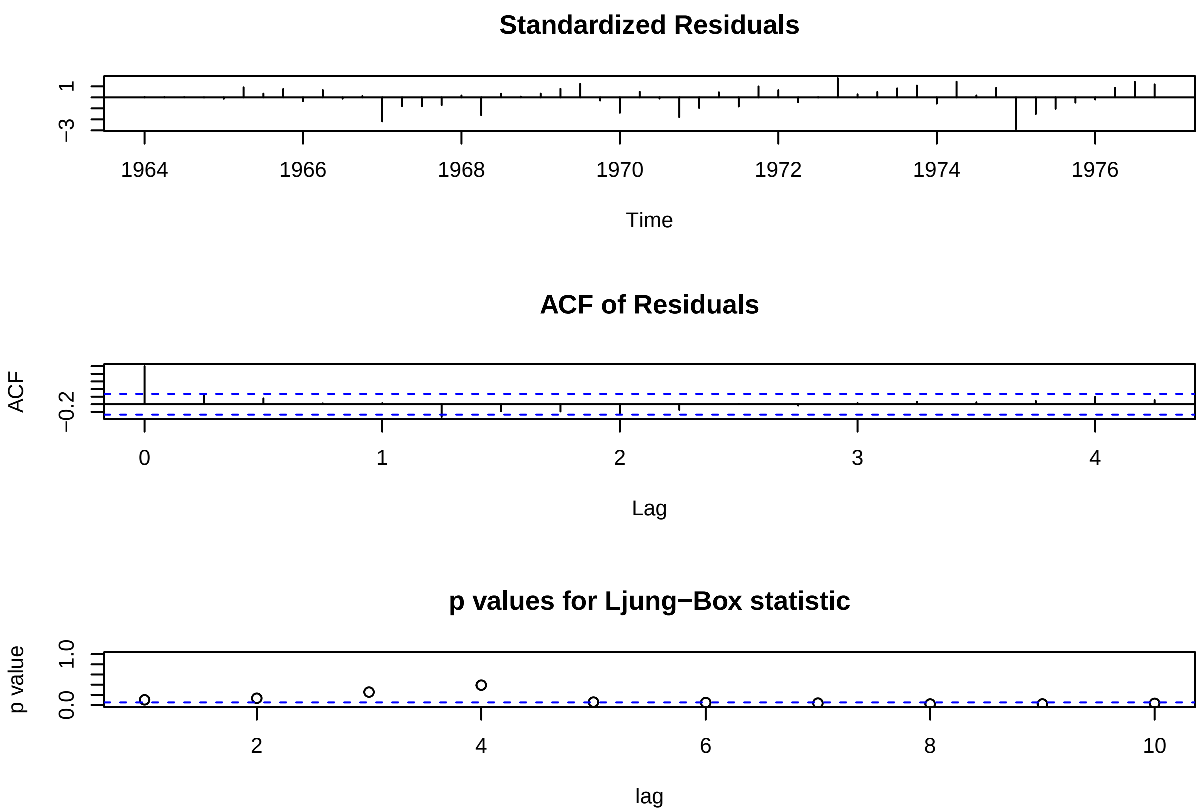





```
#Se sale cerca del lag 5
#Box-Pierce test contrasta
# Ho: Independencia vs. H1: Dependencia
Box.test(modelo_automatico2$residuals, lag =10)
```

```
##
##  Box-Pierce test
##
## data:  modelo_automatico2$residuals
## X-squared = 17.046, df = 10, p-value = 0.07335
```

```
#De manera conjunta, con la prueba de Ljung y Box
#H0:No estan correlacionados de manera conjunta
# vs
#H1:Estan correlacionados de manera conjunta
tsdiag(modelo_automatico2)
```



Coeficiente Significativo

```
SARIMA_DRIFT_int<-confint(modelo_automatico2)
SARIMA_DRIFT_int
```

```
##            2.5 %     97.5 %
## sma1 -0.9806686 -0.4939985
```

```
k=length(confint(modelo_automatico2))/2
no_sign<-c()
for(i in 1:k){
  no_sign[i]<-(SARIMA_DRIFT_int[i]<0 & SARIMA_DRIFT_int[i+k]>0)
}
#No significativos
sum(no_sign)
```

```
## [1] 0
```

```
#Porcentaje no significativos
sum(no_sign)/k
```

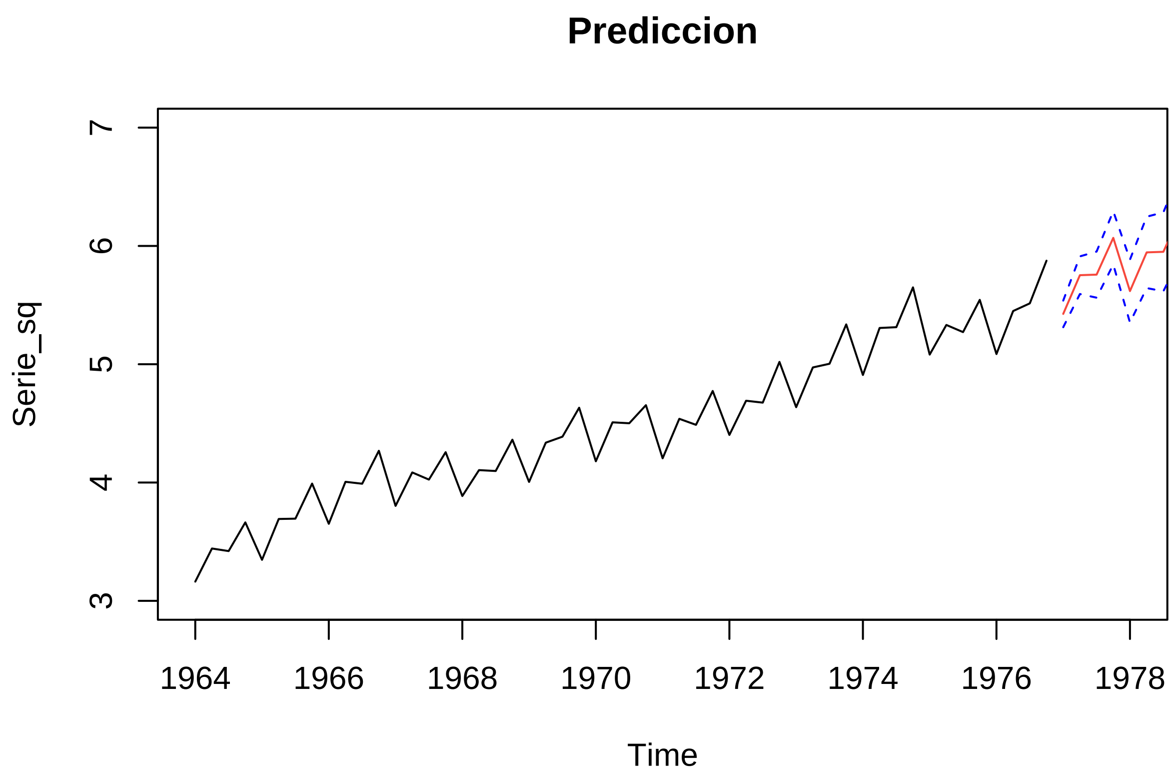
```
## [1] 0
```

4. Forecasting

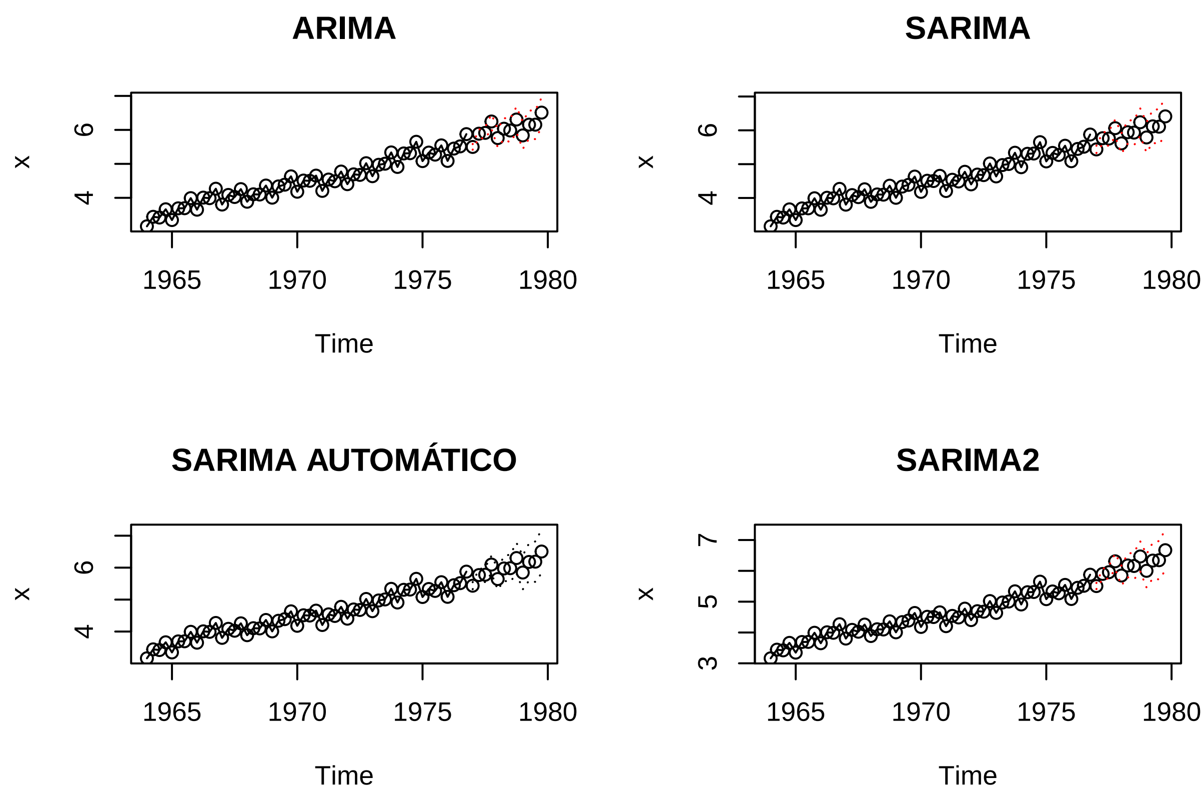
Con el modelo estimado, pronostique $n_{new} = 8$ (2 años) valores futuros (obtenga intervalos de predicción).

Usando el modelo mencionado anteiormente hacemos los pronosticos con la serie a la que le aplicamos la raíz cuadrada.

```
inicio<-start(Serie_sq)[1]
final<-end(Serie_sq)[1]
SARIMA_forecast <- predict(modelo_automatico2, n.ahead =8)$pred
SARIMA_forecast_se <- predict(modelo_automatico2, n.ahead = 8)$se
lower<-SARIMA_forecast - qnorm(0.975)*SARIMA_forecast_se
upper<-SARIMA_forecast + qnorm(0.975)*SARIMA_forecast_se
ts.plot(Serie_sq, xlim=c(inicio,final+2),ylim=c(3,7),  main="Prediccion")
points(SARIMA_forecast, type = "l", col = 2)
points(lower, type = "l", col ="blue", lty = 2)
points(upper, type = "l", col ="blue", lty = 2)
```




```
par(mfrow=c(2,2))
plot(ARIMA, col="red", main="ARIMA")
plot(SARIMA, col="red", main="SARIMA")
plot(modelo_automatico, main="SARIMA AUTOMÁTICO")
plot(SARIMA2, col="red", main="SARIMA2")
```




```
par(mfrow=c(1,1))
```

Entonces, usando la transformación inversa que es elevar al cuadrado, tenemos:

```
Predicciones_raiz<-data.frame('Puntual'=SARIMA_forecast,'Banda_inf'=lower,
                              'Banda_sup'=upper)
Predicciones_raiz
```

```
##    Puntual Banda_inf Banda_sup
## 1 5.425457  5.312915  5.538000
## 2 5.752631  5.593478  5.911785
## 3 5.757476  5.562556  5.952396
## 4 6.068250  5.843177  6.293324
## 5 5.618335  5.352146  5.884525
## 6 5.945509  5.643764  6.247255
## 7 5.950354  5.616822  6.283887
## 8 6.261128  5.898585  6.623672
```

```
Predicciones_normales<-Predicciones_raiz**2
Predicciones_normales
```

```
##    Puntual Banda_inf Banda_sup
## 1 29.43559  28.22707  30.66944
## 2 33.09277  31.28699  34.94920
## 3 33.14853  30.94203  35.43102
## 4 36.82366  34.14272  39.60592
## 5 31.56569  28.64547  34.62763
## 6 35.34908  31.85207  39.02819
```

```
## 7 35.40671  31.54869  39.48723
## 8 39.20173  34.79331  43.87302
```

```
ts.plot(Serie, xlim=c(inicio[1],final[1]+2.5),ylim=c(9,50),  main="Predicción normal")
points(SARIMA_forecast**2, type = "l", col = 3)
points(lower**2, type = "l", col ="blue", lty = 2)
points(upper**2, type = "l", col ="blue", lty = 2)
```