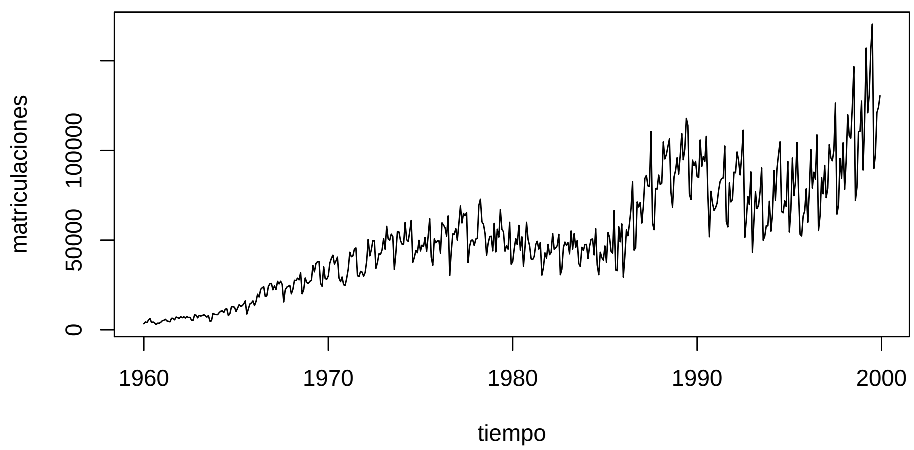
Práctica 5

1 Metodología Box-Jenkins para el análisis de series temporales

Mediante el estudio de un caso práctico ilustraremos esta metodología. Comenzaremos leyendo el fichero "matricul.dat" con datos mensuales de matriculación de automóviles de turismo en España entre 1960 y 1999.

Paso 1. Lectura y representación gráfica de los datos

```
matriculaciones_ini=read.table("matricul.dat")
matriculaciones_ini=ts(matriculaciones_ini,  frequency = 12, start = c(1960, 1))
plot(matriculaciones_ini, xlab="tiempo", ylab="matriculaciones")
```



Paso 2. Transformaciones para que la varianza sea estable en el tiempo. Se suele usar la familia de transformaciones de Box-Cox:

$$Y_t^{(\lambda)} = \begin{cases} \dfrac{Y_t^\lambda - 1}{\lambda} & \lambda \neq 0, \\ \ln(Y_t) & \lambda = 0. \end{cases}$$

```
library(TSA)
?BoxCox.ar
bc=BoxCox.ar(y=matriculaciones_ini); bc
# $lambda
#  [1] -2.0 -1.9 -1.8 -1.7 -1.6 -1.5 -1.4 -1.3 -1.2 -1.1 -1.0 -0.9 -0.8 -0.7 -0.6
# [16] -0.5 -0.4 -0.3 -0.2 -0.1  0.0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9
# [31]  1.0  1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0
#
# $loglike
#  [1]  156.2290  258.4346  359.7327  459.7609  558.6772  655.3944  750.0250
#  [8]  842.1259  931.5227 1017.7046 1099.7947 1169.8251 1244.3402 1313.1645
# [15] 1375.5714 1430.6982 1477.7695 1516.2038 1545.7522 1566.5666 1579.1724
# [22] 1584.3671 1583.0905 1576.3030 1564.8947 1549.6306 1531.1240 1509.9018
# [29] 1486.2994 1460.6115 1433.0917 1403.8569 1373.0507 1340.7710 1307.0959
# [36] 1272.0948 1235.8231 1198.3299 1159.6635 1119.8645 1078.9719
#
# $mle
# [1] 0.1
```

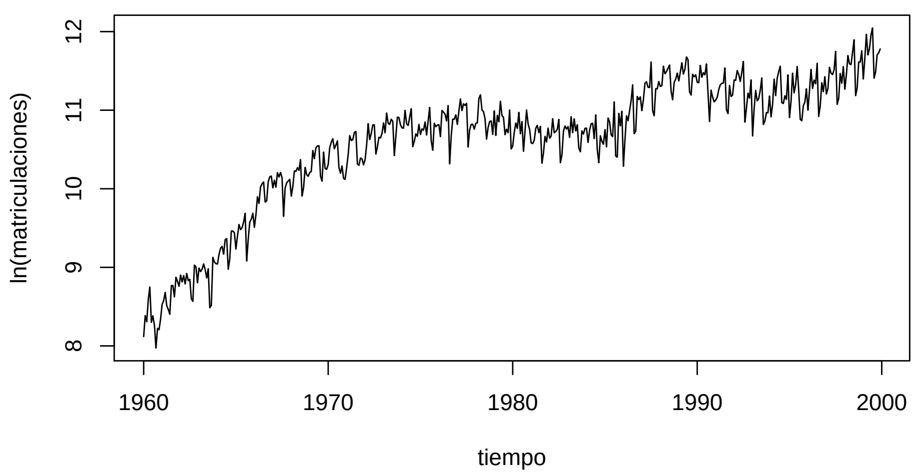
```
#
# $ci
# [1] 0.1 0.2
```

Podemos solicitar la búsqueda en una rejilla más fina (por defecto va de -2 a 2 con saltos de 0.1):

```
bc.bis=BoxCox.ar(y=matriculaciones_ini, lambda = seq(0, 0.25, 0.01) ); bc.bis
# $lambda
#  [1] 0.00 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.10 0.11 0.12 0.13 0.14
# [16] 0.15 0.16 0.17 0.18 0.19 0.20 0.21 0.22 0.23 0.24 0.25
#
# $loglike
#  [1] 1579.172 1580.010 1580.775 1581.467 1582.088 1582.639 1583.120 1583.532
#  [9] 1583.877 1584.155 1584.367 1584.514 1584.599 1584.620 1584.579 1584.477
# [17] 1584.316 1584.095 1583.817 1583.482 1583.090 1582.644 1582.144 1581.591
# [25] 1580.985 1580.328
#
# $mle
# [1] 0.13
#
# $ci
# [1] 0.06 0.20
```

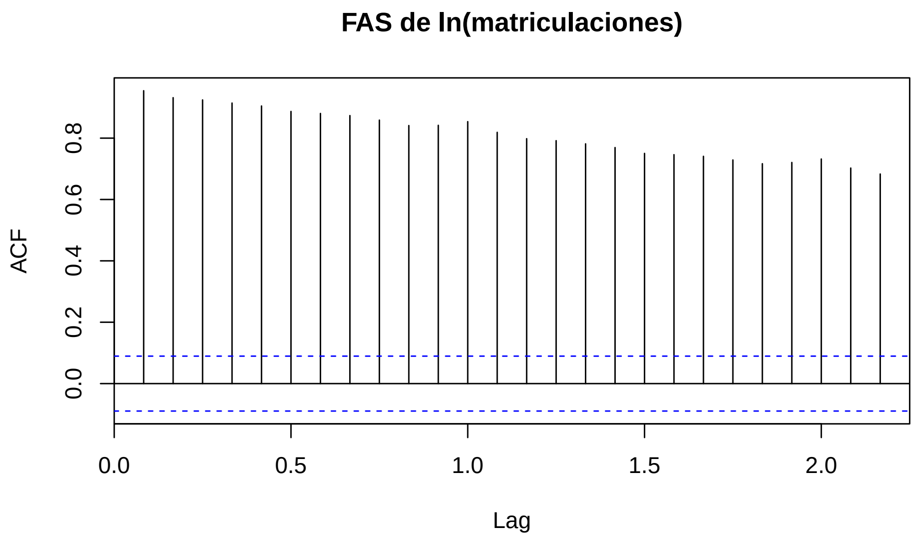
El EMV de λ es 0.13. Realizaremos la transformación para $\lambda = 0$.

```
matriculaciones=log(matriculaciones_ini)
plot(matriculaciones, xlab="tiempo", ylab="ln(matriculaciones)")
```



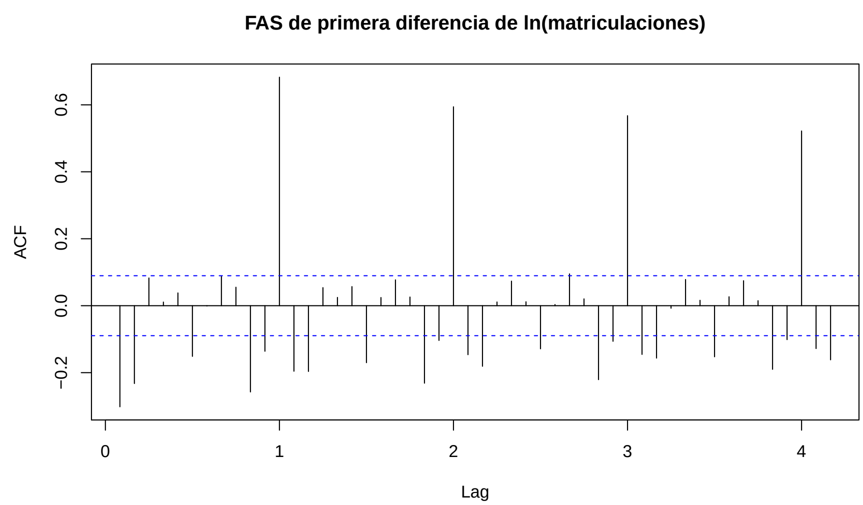
Paso 3. Transformaciones para que la media sea estable en el tiempo. Intentamos ver si la no estacionalidad en la media es debido a que se trata de un proceso integrado (la FAS decrece linealmente)

```
acf(matriculaciones, main="FAS de ln(matriculaciones)")
```



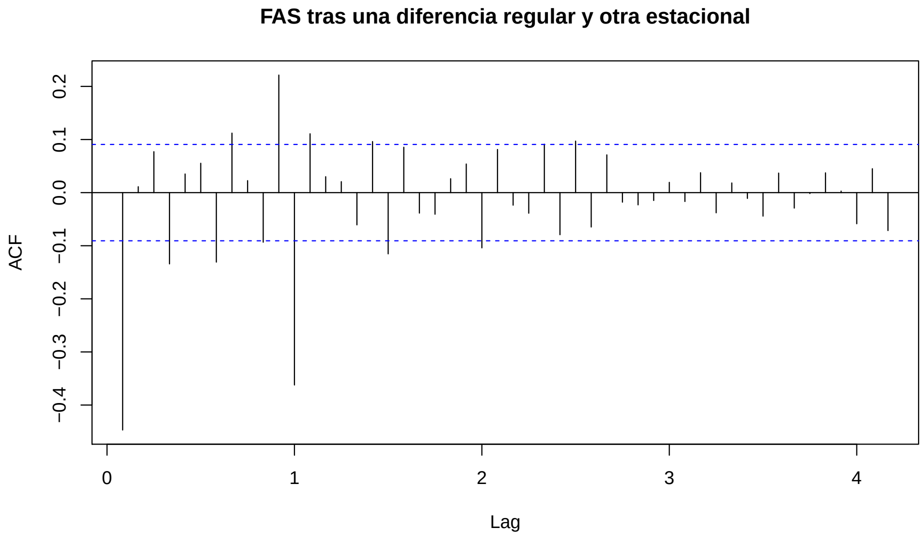
La FAS muestral desciende muy lentamente, lo que indica que se trata de un proceso integrado. Diferenciamos (diferencia regular):

```
matri_ini=diff(matriculaciones, lag=1, diff=1)
acf(matri_ini, main="FAS de primera diferencia de ln(matriculaciones)", lag=50)
```



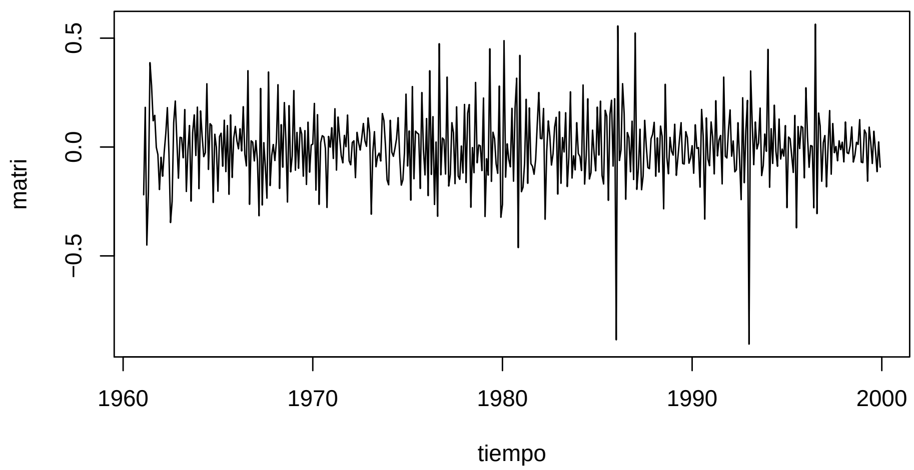
Vemos que las autocorrelaciones de periodo $s = 12$ descienden lentamente, lo que indica que es conveniente tomar una diferencia estacional:

```
matri=diff(matri_ini, lag=12, diff=1)
acf(matri, main="FAS tras una diferencia regular y otra estacional", lag=50)
```



Vemos que la FAS se corresponde con la de un proceso estacionario. También puede verse representando la serie diferenciada:

```
plot(matri)
```



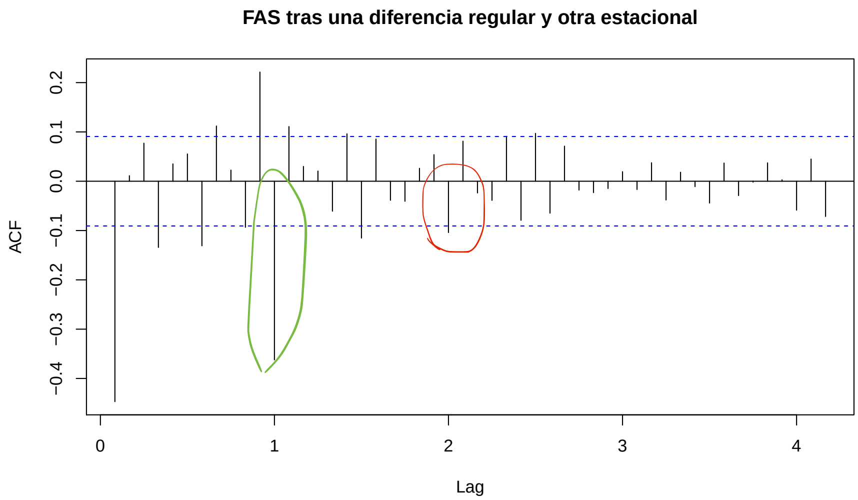
Paso 4. Contrastar estacionariedad Aplicar test de la raíz unitaria

h1: datos estacionarios

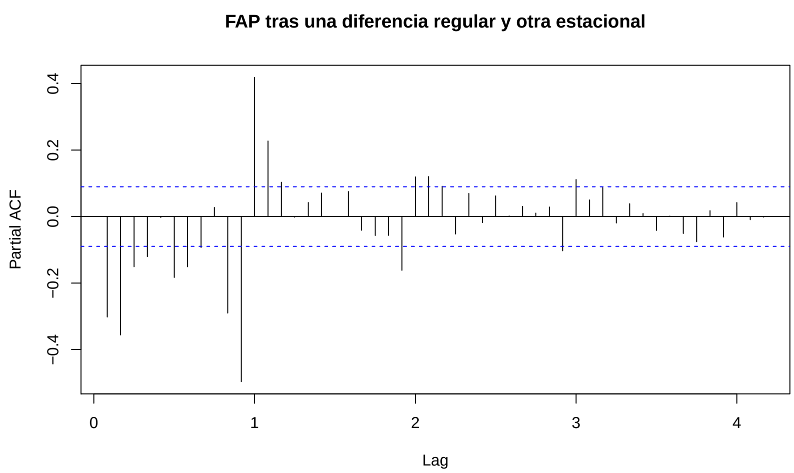
```
library(tseries)
adf.test(matri)
#
#        Augmented Dickey-Fuller Test
#
# data:  matri
# Dickey-Fuller = -10.112, Lag order = 7, p-value = 0.01
# alternative hypothesis: stationary

# Warning message:
# In adf.test(matri) : p-value smaller than printed p-value
```

Paso 5. Identificar estructura ARIMA

```
acf(matri, main="FAS tras una diferencia regular y otra estacional", lag=50)
```




```
pacf(matri_ini, main="FAP tras una diferencia regular y otra estacional", lag=50)
```



Probaremos $MA(1)$ para la parte regular y $AR(2)$ ó $ARMA(1,1)$ para la parte estacional.

Pasos 6 y 7. Estimación de parámetros y diagnóstico

```
ajuste1=arima(matri, order=c(0,0,1), seasonal=list(order=c(2,0,0), period=12)); ajuste1
# Call:
# arima(x = matri, order = c(0, 0, 1), seasonal = list(order = c(2, 0, 0), period = 12))
#
# Coefficients:
#           ma1     sar1     sar2  intercept
#       -0.6044  -0.5031  -0.2656    -0.0003
# s.e.   0.0408   0.0465   0.0465     0.0014
#
# sigma^2 estimated as 0.01674:  log likelihood = 290.16,  aic = -572.32
```

Miramos si los coeficientes estimados son significativamente no nulos:

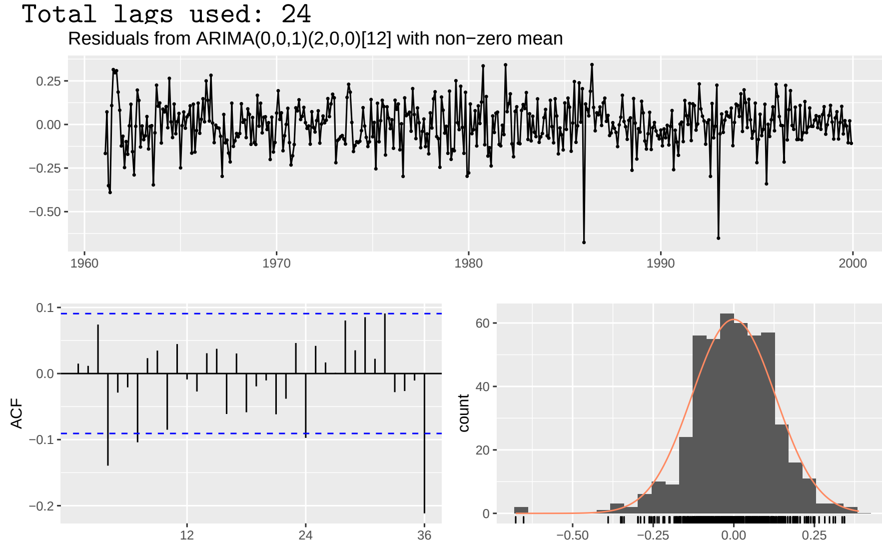
```
confint(ajuste1)
#                   2.5 %       97.5 %
# ma1         -0.684459655 -0.524434120
# sar1        -0.594212665 -0.412044884
# sar2        -0.356804417 -0.174340206
# intercept   -0.002952072  0.002399943
```

Eliminamos la media del modelo:

```
ajuste1=arima(matri, order=c(0,0,1), seasonal=list(order=c(2,0,0), period=12), include.mean=FALSE)
ajuste1
# Call:
# arima(x = matri, order = c(0, 0, 1), seasonal = list(order = c(2, 0, 0), period = 12),
#     include.mean = FALSE)
#
# Coefficients:
#           ma1     sar1     sar2
#       -0.6044  -0.5031  -0.2656
# s.e.   0.0408   0.0465   0.0465
#
# sigma^2 estimated as 0.01675:  log likelihood = 290.14,  aic = -574.28
```

Diagnosis del modelo:

```
library(forecast)
checkresiduals(ajuste1)
#
#         Ljung-Box test
#
# data:  Residuals from ARIMA(0,0,1)(2,0,0)[12] with non-zero mean
# Q* = 37.016, df = 20, p-value = 0.01165
#
# Model df: 4.   Total lags used: 24
```



El modelo no es adecuado, ya que se rechaza que las innovaciones están incorreladas, y en la FAS de las innovaciones hay muchos "palos" fuera de las bandas. Intentamos otro

```
ajuste2=arima(matri, order=c(0,0,1), seasonal=list(order=c(1,0,1), period=12)); ajuste2
# Call:
# arima(x = matri, order = c(0, 0, 1), seasonal = list(order = c(1, 0, 1), period = 12))
#
# Coefficients:
#           ma1    sar1     sma1  intercept
#       -0.6119  0.2093  -0.8882     -4e-04
# s.e.   0.0393  0.0576   0.0336      4e-04
#
#sigma^2 estimated as 0.01448:  log likelihood = 318.89,  aic = -629.78
```

Miramos si los coeficientes estimados son significativamente no nulos:

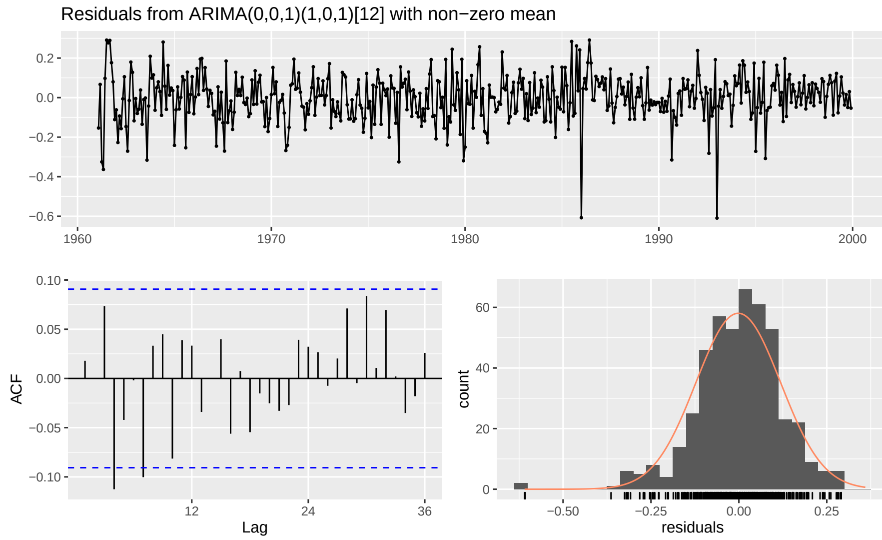
```
confint(ajuste2)
#                  2.5 %        97.5 %
# ma1       -0.688880770 -0.5348721676
# sar1       0.096494765  0.3221132470
# sma1      -0.953952817 -0.8223660700
# intercept -0.001139184  0.0004107751
```

Eliminamos la media del modelo:

```
ajuste2=arima(matri, order=c(0,0,1), seasonal=list(order=c(1,0,1), period=12), include.mean=FALSE)
ajuste2
# Call:
# arima(x = matri, order = c(0, 0, 1), seasonal = list(order = c(1, 0, 1), period = 12),
#     include.mean = FALSE)
#
# Coefficients:
#           ma1    sar1     sma1
#       -0.6087  0.2071  -0.8833
# s.e.   0.0392  0.0579   0.0338
#
# sigma^2 estimated as 0.01452:  log likelihood = 318.48,  aic = -630.95
```

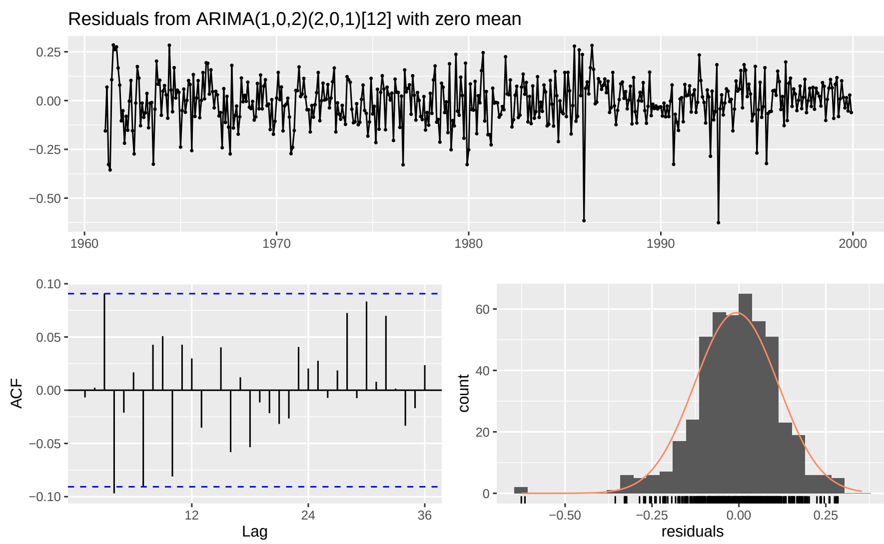
Diagnosis del modelo:

```
checkresiduals(ajuste2)
#
#        Ljung-Box test
#
# data:  Residuals from ARIMA(0,0,1)(1,0,1)[12] with non-zero mean
# Q* = 27.168, df = 20, p-value = 0.1306
#
# Model df: 4.   Total lags used: 24
```



El paquete `forecast` busca modelo que minimiza AIC

```
(ajuste3=auto.arima(matri))
```
hemos metido los datos diferenciados, sino saldría q=Q=1
```
# Series: matri
# ARIMA(1,0,2)(2,0,1)[12] with zero mean
#
# Coefficients:
#          ar1      ma1     ma2    sar1    sar2     sma1
#       0.7224  -1.3138  0.4083  0.2108  0.0088  -0.8833
# s.e.  0.3612   0.3714  0.2392  0.0597  0.0558   0.0382
#
# sigma^2 estimated as 0.01468:  log likelihood=319.04
# AIC=-624.08   AICc=-623.83   BIC=-595.05

checkresiduals(ajuste3)

#
#        Ljung-Box test
#
# data:  Residuals from ARIMA(1,0,2)(2,0,1)[12] with zero mean
# Q* = 25.79, df = 18, p-value = 0.1046
#
# Model df: 6.   Total lags used: 24
```



La función `auto.arima` hace una búsqueda secuencial (paso a paso), que no necesariamente examina todos los modelos. Podemos decirle que genere todos los modelos (tiene unas cotas superiores para q, d, q, P, D y Q):

```
(ajuste4=auto.arima(matri, step=FALSE))
# Series: matri
# ARIMA(0,0,1)(1,0,1)[12] with zero mean
#
# Coefficients:
#            ma1    sar1     sma1
#        -0.6087  0.2071  -0.8833
# s.e.    0.0392  0.0579   0.0338
#
# sigma^2 estimated as 0.01461:  log likelihood=318.48
# AIC=-628.95   AICc=-628.87   BIC=-612.37
```

que coincide con el ajuste 2.

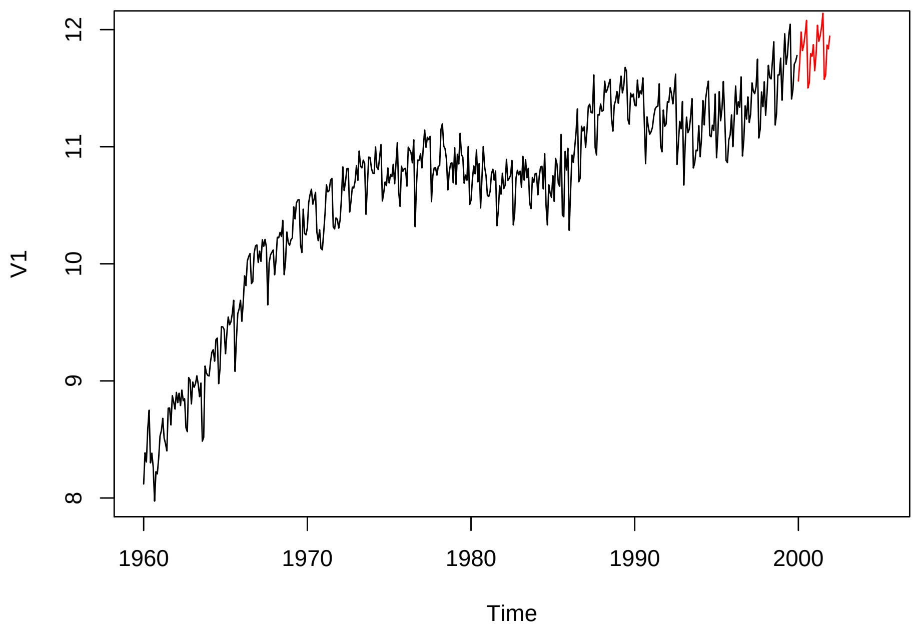
Hemos ajustado la serie diferenciada. Equivalentemente,

```
(ajuste.bueno=arima(matriculaciones, order=c(0,1,1), seasonal=list(order=c(1,1,1), period=12)))
# arima(x = matriculaciones, order = c(0, 1, 1), seasonal = list(order = c(1,
#    1, 1), period = 12))
#
# Coefficients:
#           ma1    sar1     sma1
#       -0.6087  0.2071  -0.8833
# s.e.   0.0392  0.0579   0.0338
#
# sigma^2 estimated as 0.01452:  log likelihood = 318.48,  aic = -630.95
```

Por defecto, para series diferenciadas, la función `arima` no incluye término independiente, pero se le puede solicitar que lo incluya.

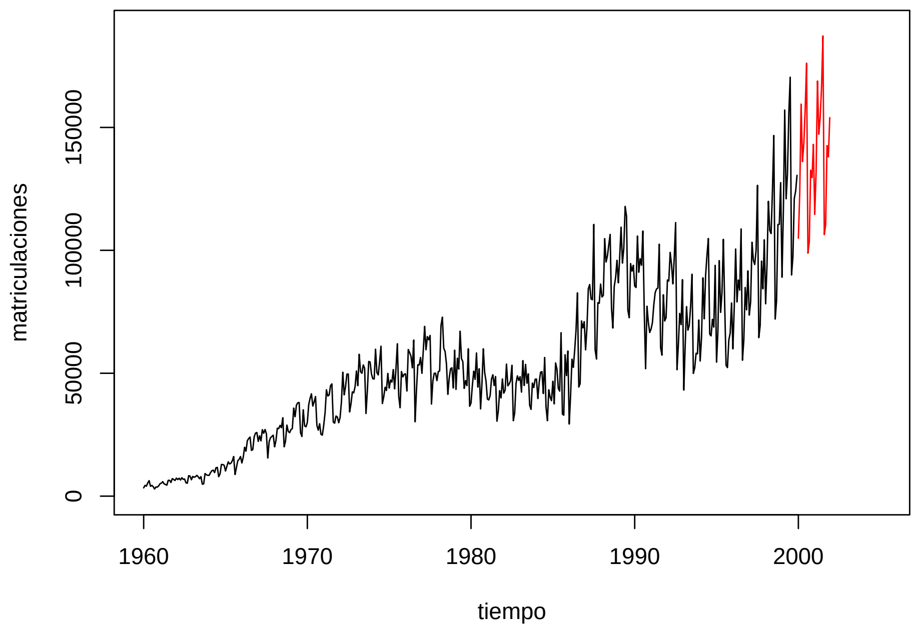
Paso 8. Predicción de resultados

```
pred=predict(ajuste.bueno,n.ahead=24) # da prediccin + error estndar de la prediccin
plot(matriculaciones, xlim=c(1960, 2005), ylim=c(8,12))
lines(pred$pred, col="red")
```



Finalmente, hay que deshacer los cambios hechos para obtener predicciones sobre los datos originales:

```
plot(matriculaciones_ini, xlab="tiempo", ylab="matriculaciones", xlim=c(1960, 2005), ylim=c(0,190000
lines(exp(pred$pred), col="red")
```



También podemos usar el paquete `forecast`:

```
plot((forecast(ajuste.bueno,h=24))) #muy similar al anterior
```

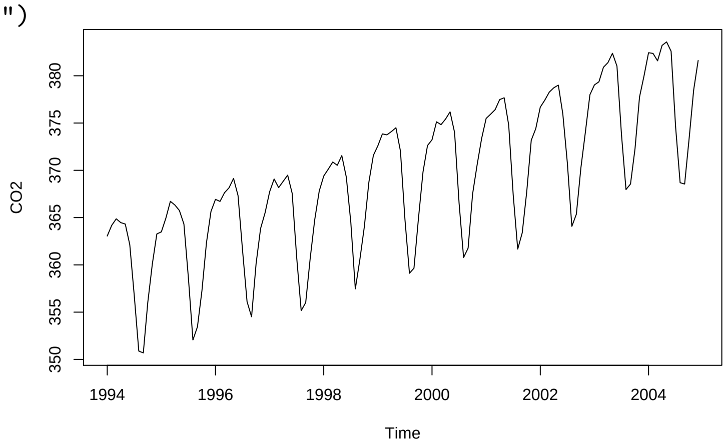
Ejercicio 1: Analizar el *dataset* co2, contenido en la librería `TSA`, que contiene una serie de tiempo de 132 observaciones mensuales de 1994 a 2005, relativa a la concentración de CO2 en Alert (Canadá).

Ejercicio 2: Analizar la serie mensual de pasajeros (en cientos) de aerolíneas internacionales entre 1949 to 1960. (*dataset* de `R` AirPassengers).

Análisis de los datos de co2

Paso 1. Lectura y representación gráfica de los datos

```
library(TSA); library (tseries); library(forecast) #cargamos todas las libreras que vamos a usar
data(co2)
str(co2)
plot (co2, ylab="CO2")
```



Paso 2. Transformaciones para que la varianza sea estable en el tiempo

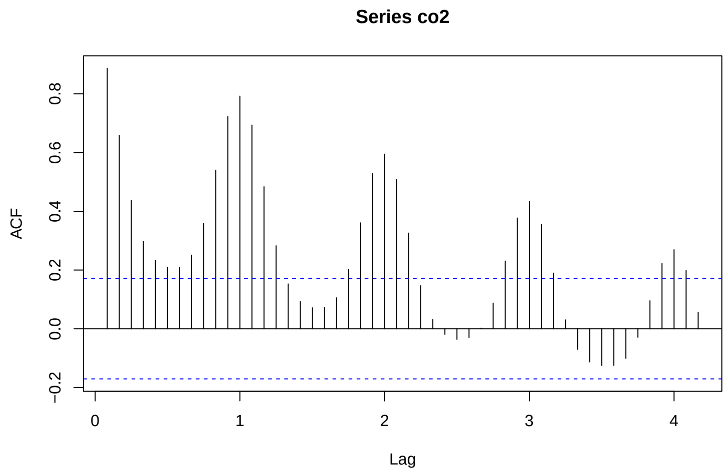
Del gráfico de la serie, no parece que haya que aplicar alguna trasnformación.

```
bc=BoxCox.ar(y=co2); bc
# $lambda
#  [1] -2.0 -1.9 -1.8 -1.7 -1.6 -1.5 -1.4 -1.3 -1.2 -1.1 -1.0 -0.9 -0.8 -0.7 -0.6 -0.5
# [17] -0.4 -0.3 -0.2 -0.1  0.0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.0  1.1
# [33]  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0
#
# $loglike
#  [1] 764.9878 765.1478 765.2733 765.4660 765.6224 765.7770 765.9292 766.0768 766.2266
# [10] 766.3696 766.5153 766.6602 766.7996 766.9391 767.0781 767.2136 767.3454 767.4786
# [19] 767.6073 767.7309 767.8604 767.9759 768.0986 768.2251 768.3405 768.4571 768.5717
# [28] 768.6814 768.7919 768.9001 769.0082 769.0625 769.2070 769.3118 769.4059 769.5019
# [37] 769.5952 769.6870 769.7629 769.8643 769.9229
#
# $mle
# [1] 2
#
# $ci
# [1] 0.2 2.0
```

Como 1 está en el intervalo de confiaza para λ, no transformamos los datos.

Paso 3. Transformaciones para que la media sea estable en el tiempo.

```
acf(co2, lag=50)
```

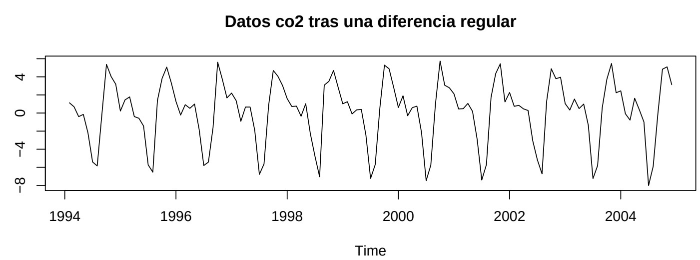


Claramente hay que aplicar una diferencia regular y otra estacional. Aplcamos primero una regular:

```
co2.d1=diff(co2)
```

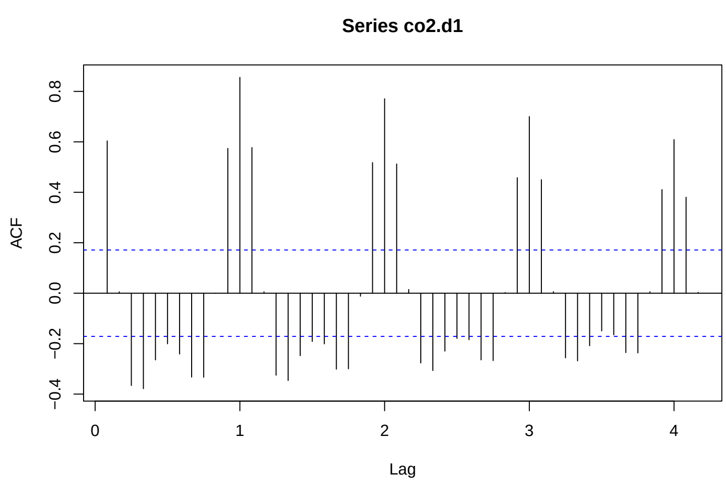
Veamos qué aspecto tiene la serie tras una diferencia regular:

```
plot(co2.d1, main="Datos co2 tras una diferencia regular", ylab=" ")
```



Miramos ahora la FAS:

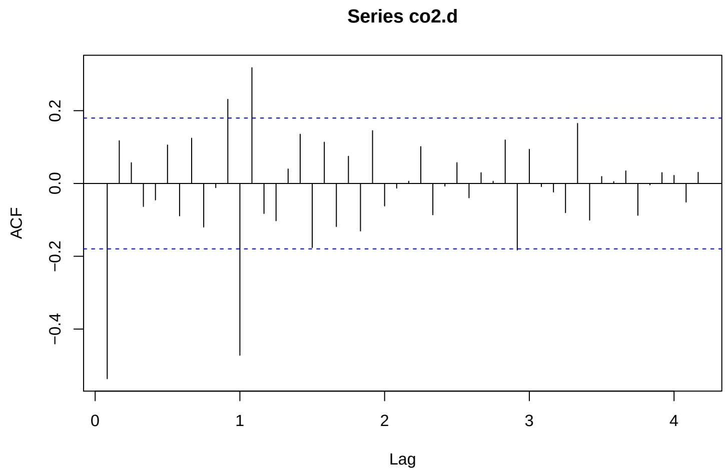
```
acf(co2.d1, lag=50)
```



Las autocorrelaciones estacionales desciende lentamente, aplicamos una diferencia estacional.

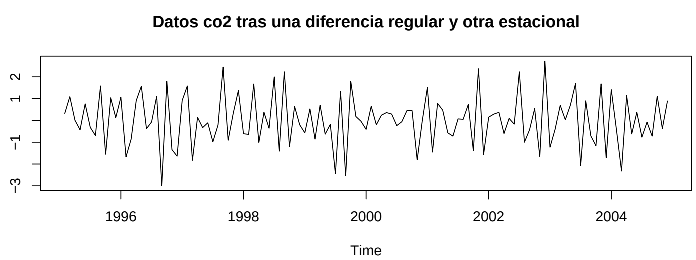
```
co2.d=diff(co2.d1, lag=12)
```

```
acf(co2.d, lag=50)
```



Ahora el gráfico de la FAS tiene el aspecto que debe tener un proceso estacionario (no integrado). Vemos qué aspecto tiene la serie tras una diferencia regular y otra estacional:

```
plot(co2.d, main="Datos co2 tras una diferencia regular y otra estacional", ylab=" ")
```

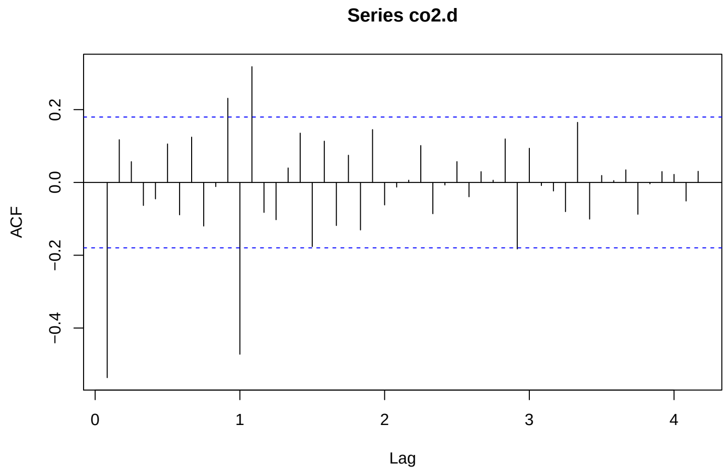


Paso 4. Contrastar estacionariedad Aplicar test de la raíz unitaria

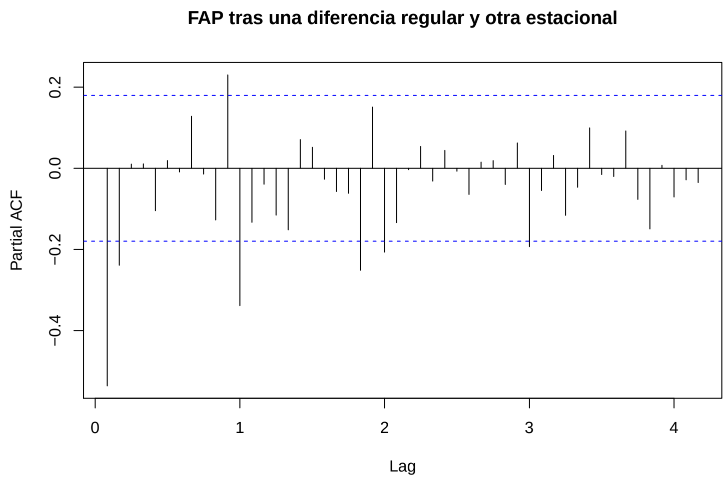
```
adf.test(co2.d)
#
#         Augmented Dickey-Fuller Test
#
# data:  co2.d
# Dickey-Fuller = -5.9106, Lag order = 4, p-value = 0.01
# alternative hypothesis: stationary
#
# Warning message:
# In adf.test(co2.d) : p-value smaller than printed p-value
```

Paso 5. Identificar estructura ARIMA

```
acf(co2.d, main="FAS tras una diferencia regular y otra estacional", lag=50)
```




```
pacf(co2.d, main="FAP tras una diferencia regular y otra estacional", lag=50)
```

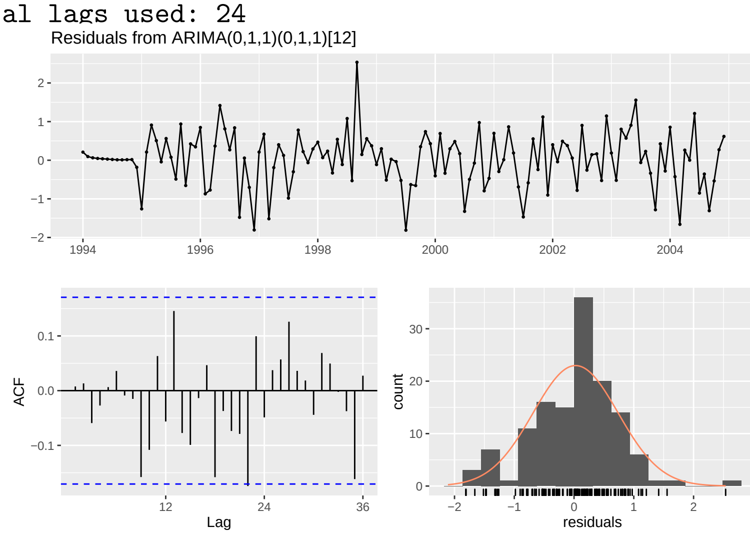


Vemos una autocorrelación no nula en el retardo 1 y otra en el retardo 12. La del retardo 12 tiene una autocorrelación no nula a cada lado (correspondientes a la parte regular). Esto nos sugere considerar un modelo $MA(1)$ para la parte regular y $MA(1)$ para la parte estacional.

Pasos 6 y 7. Estimación de parámetros y diagnóstico

```
(ajuste1=arima(co2, order=c(0,1,1), seasonal=list(order=c(0,1,1), period=12)))
# Call:
# arima(x = co2, order = c(0, 1, 1), seasonal = list(order = c(0, 1, 1), period = 12))
#
# Coefficients:
#           ma1     sma1
#       -0.5792  -0.8206
# s.e.   0.0791   0.1137
#
# sigma^2 estimated as 0.5446:  log likelihood = -139.54,  aic = 283.08
checkresiduals(ajuste1)
#
#   Ljung-Box test
#
# data:  Residuals from ARIMA(0,1,1)(0,1,1)[12]
```

```
# Q* = 25.891, df = 22, p-value = 0.2564
#
# Model df: 2.   Total lags used: 24
```



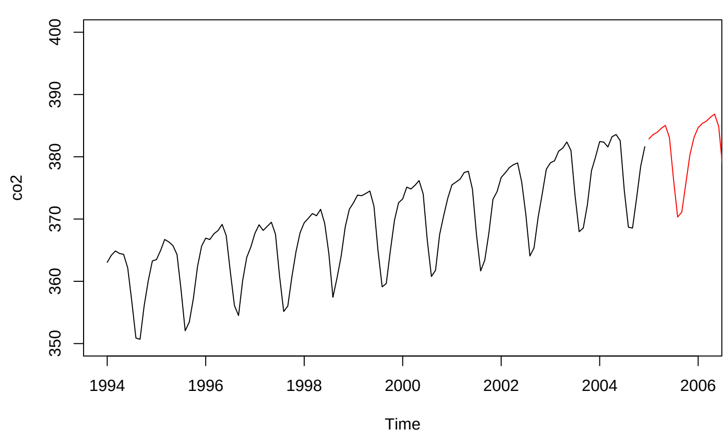
Veamos qué modelo selecciona la función `auto.arima`. Para que no tarde demasiado fijamos $d = 1$ y $D = 1$:

```
(ajuste2=auto.arima(co2, d=1, D=1, step=FALSE))
```

Selecciona el ya ajustado.

Paso 8. Predicción de resultados

```
pred=predict(ajuste1,n.ahead=24)
plot(co2, xlim=c(1994, 2006), ylim=c(350,400))
lines(pred$pred, col="red")
```

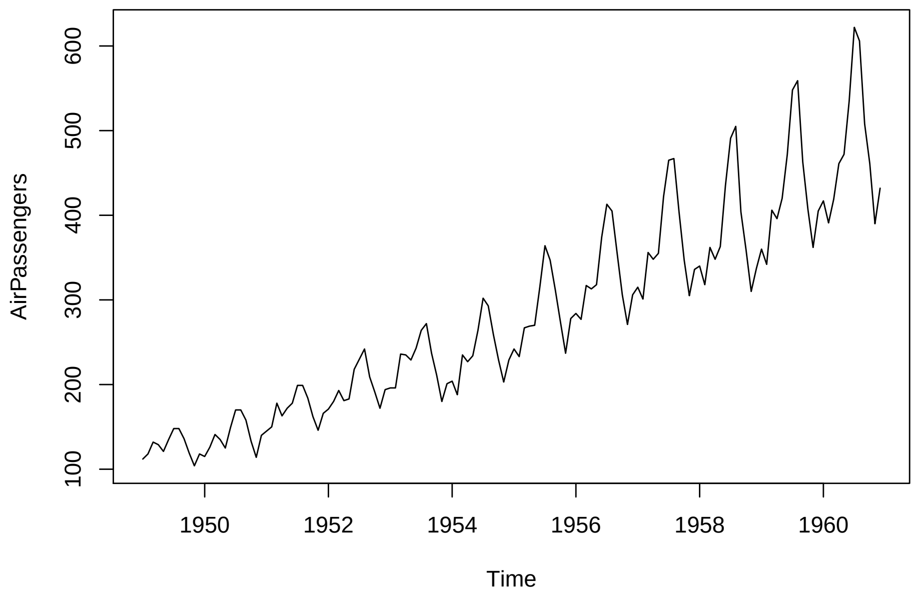



```
forecast(ajuste2,h=24))
plot(forecast(ajuste2,h=24))
```

Análisis de los datos de pasajeros de aerolíneas: AirPassengers

Paso 1. Lectura y representación gráfica de los datos

```
data(AirPassengers)
?AirPassengers
plot(AirPassengers)
```

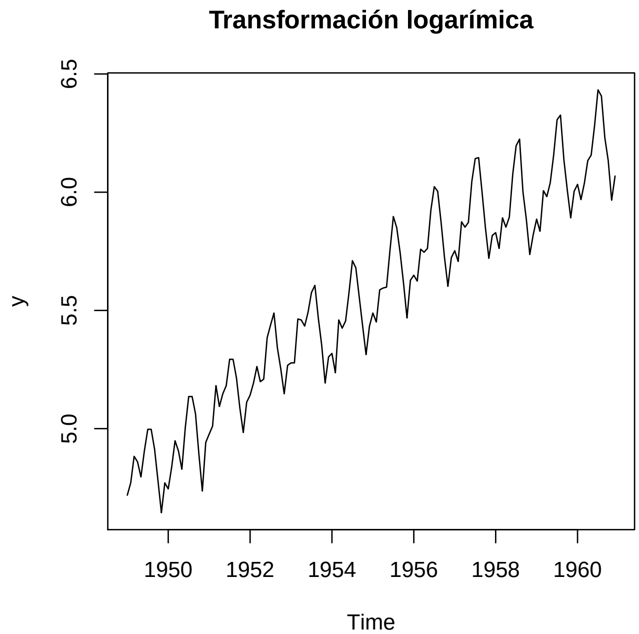


Paso 2. Transformaciones para que la varianza sea estable en el tiempo Claramente hay que aplicar alguna transformación que estabilice la varianza.

```
bc=BoxCox.ar(y=AirPassengers); bc

#da error, por defecto usa mle, podemos intentar otro metodo, a ver si no da error

bc=BoxCox.ar(y=AirPassengers, method = "ols")
# $lambda
#  [1] -2.0 -1.9 -1.8 -1.7 -1.6 -1.5 -1.4 -1.3 -1.2 -1.1 -1.0 -0.9 -0.8 -0.7 -0.6 -0.5
# [17] -0.4 -0.3 -0.2 -0.1  0.0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.0  1.1
# [33]  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0
#
# $loglike
#  [1] 440.3195 447.1485 453.8219 460.3306 466.6652 472.8152 478.7698 484.5175 490.0461
# [10] 495.3425 500.3933 505.1840 509.6995 513.9243 517.8418 521.4352 524.6871 527.5798
# [19] 530.0956 532.2172 533.9279 535.2123 536.0567 536.4499 536.3837 535.8533 534.8578
# [28] 533.4005 531.4887 529.1337 526.3507 523.1576 519.5752 515.6259 511.3336 506.7224
# [37] 501.8167 496.6400 491.2152 485.5635 479.7049
#
# $mle
# [1] 0.3
#
#$ci
#[1] 0.1 0.6
BoxCox.lambda(AirPassengers,method = "loglik") #funcion del paquete forecast
# 0.2
```

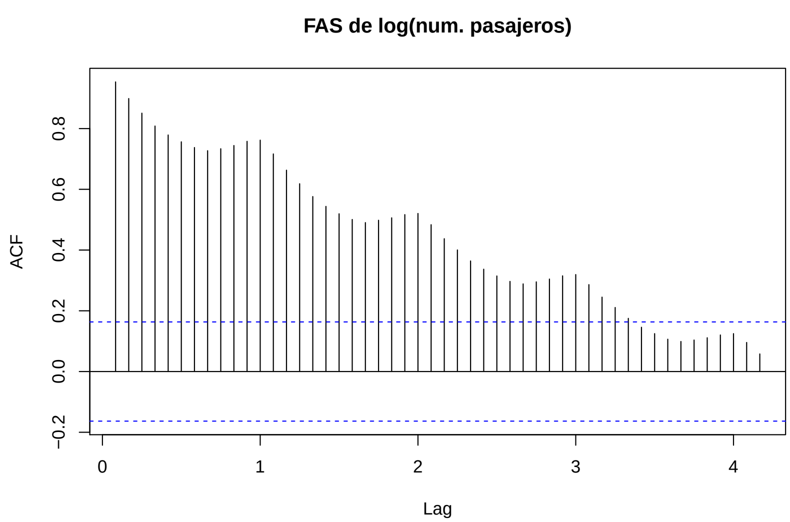
Intentamos la transformación logarítmica

```
y=log(AirPassengers)
plot(y, main="Transformacin logarmica")
```



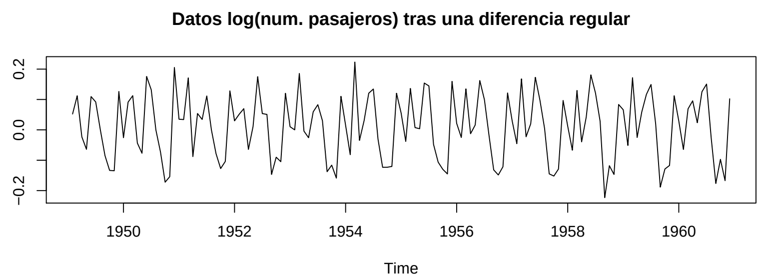
Paso 3. Transformaciones para que la media sea estable en el tiempo.

```
acf(y, main="FAS de log(num. pasajeros)", lag=50)
```

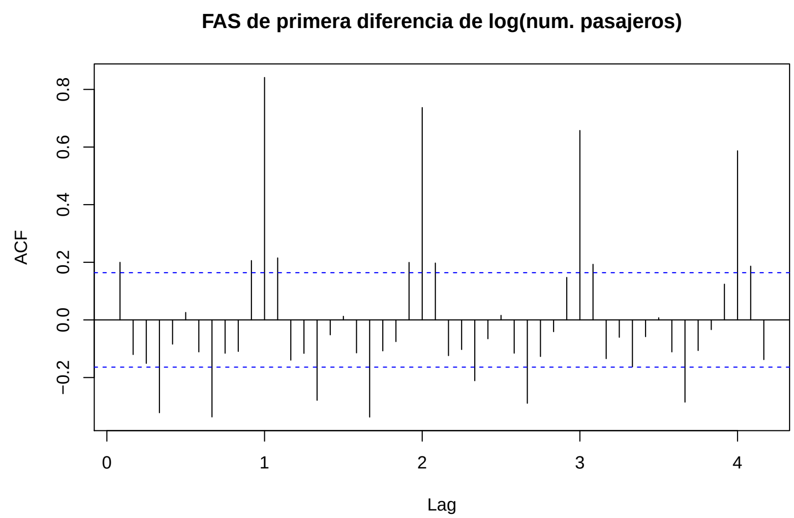


Desciende muy lentamente, lo que indica que se trata de un proceso integrado. Además también se aprecia el efecto estacional. Aplicamos primero una diferencia regular:

```
y1=diff(y, lag=1, diff=1)
plot(y1,  main="Datos log(num. pasajeros) tras una diferencia regular", ylab=" ")
```

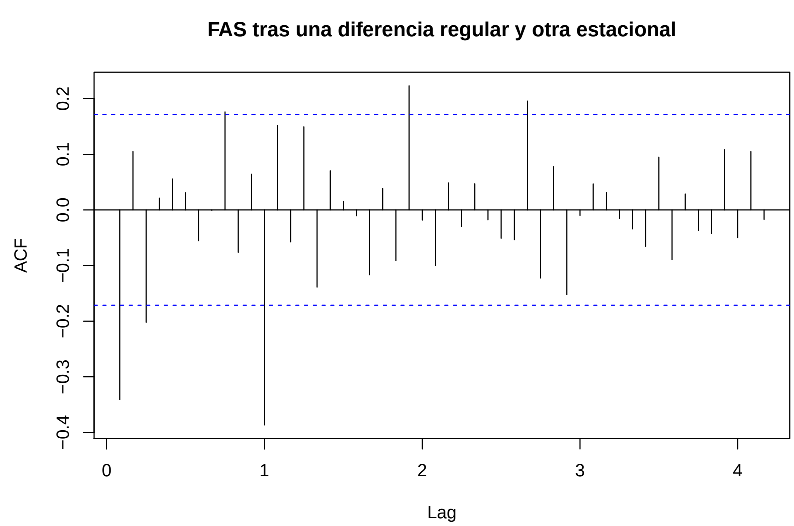



```
acf(y1, main="FAS de primera diferencia de log(num. pasajeros)", lag=50)
```

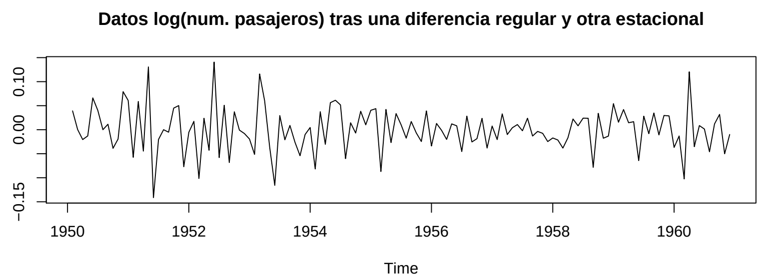


Las autocorrelaciones estacionales descienden muy lentamente, aplicamos una diferencia estacional:

```
y2=diff(y1, lag=12, diff=1)
acf(y2, main="FAS tras una diferencia regular y otra estacional")
```



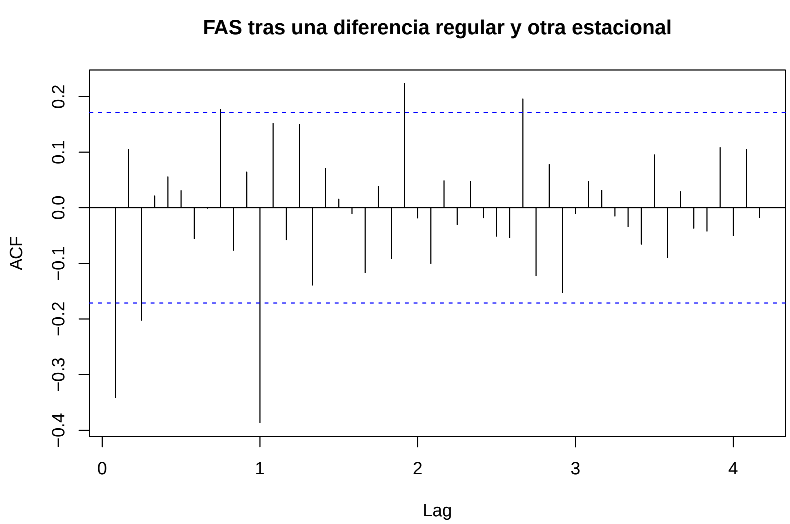

```
plot(y2, main="Datos log(num. pasajeros) tras una diferencia regular y otra estacional", ylab=" ")
```



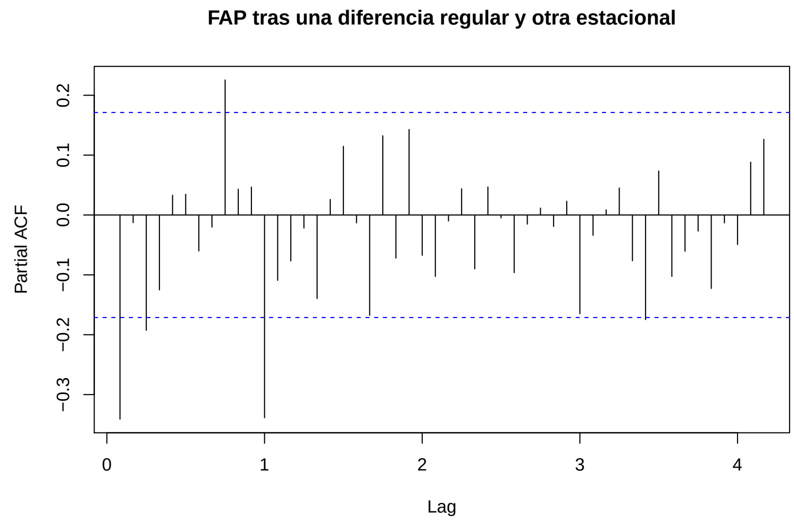
Paso 4. Contrastar estacionariedad Aplicar test de la raíz unitaria

```
library(tseries)
adf.test(y2)
#
#        Augmented Dickey-Fuller Test
#
# data:  y2
# Dickey-Fuller =  -5.1993, Lag order = 5, p-value = 0.01
# alternative hypothesis: stationary

# Warning message:
# In adf.test(y2) : p-value smaller than printed p-value
```

Paso 5. Identificar estructura ARIMA

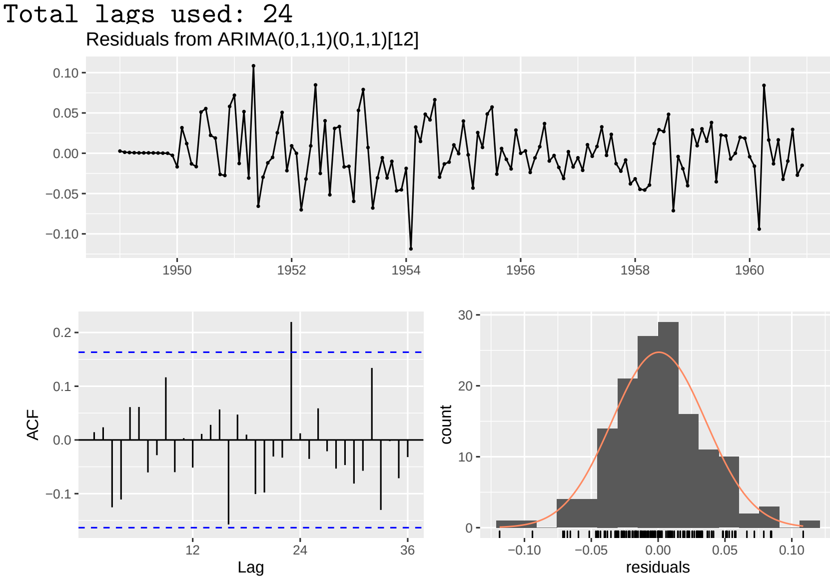



```
pacf(y2, main="FAP tras una diferencia regular y otra estacional", lag=50)
```



Del gráfico FAS parace que la parte regular y la estacionaria son ambas MA.

```
ajuste1=arima(y, order=c(0,1,1), seasonal=list(order=c(0,1,1), period=12)); ajuste1
# Call:
# arima(x = y, order = c(0, 1, 1), seasonal = list(order = c(0, 1, 1), period = 12))
#
# Coefficients:
#          ma1      sma1
#      -0.4018   -0.5569
# s.e.  0.0896    0.0731
#
# sigma^2 estimated as 0.001348:  log likelihood = 244.7,  aic = -485.4
checkresiduals(ajuste1)
#
#       Ljung-Box test
#
# data:  Residuals from ARIMA(0,1,1)(0,1,1)[12]
# Q* = 26.446, df = 22, p-value = 0.233
#
# Model df: 2.   Total lags used: 24
```



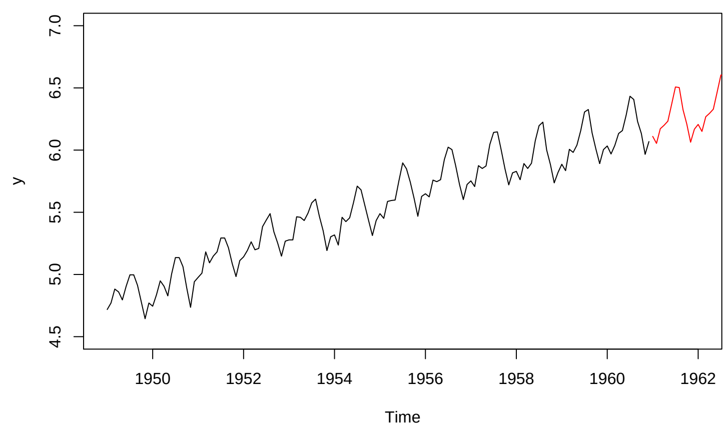
Parece un ajuste adecuado. Veamos qué hace `auto.arima`

```
(ajuste2=auto.arima(y, d=1, D=1, step=FALSE))
```

Selecciona el mismo modelo.

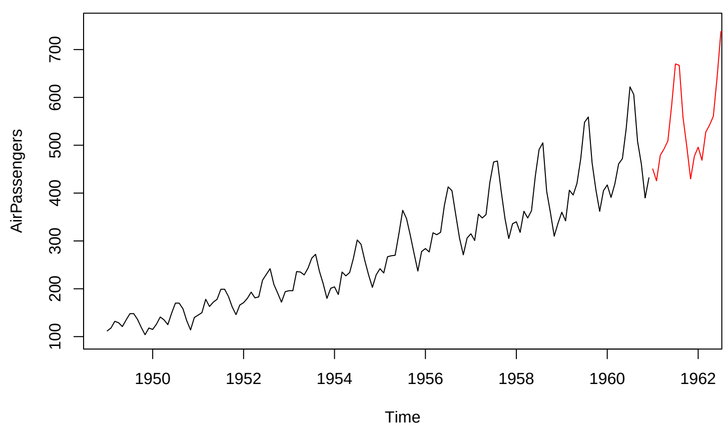
Paso 8. Predicción de resultados

```
pred=predict(ajuste1,n.ahead=24)
plot(y, xlim=c(1949, 1962), ylim=c(4.5, 7))
lines(pred$pred, col="red")
```

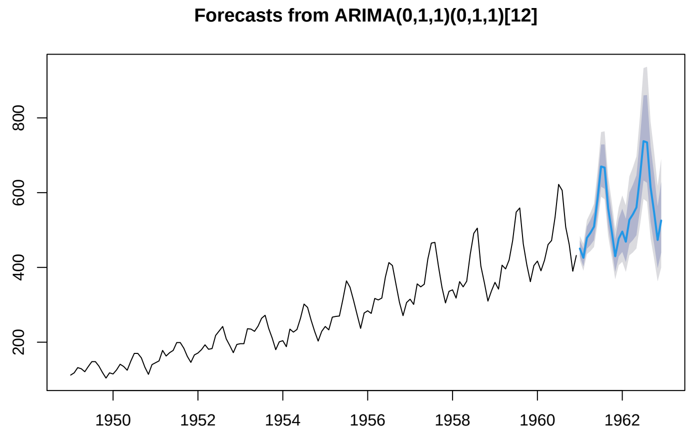


Finalmente hay que deshacer los cambios:

```
plot(AirPassengers, xlim=c(1949, 1962), ylim=c(100,750))
lines(exp(pred$pred), col="red")
```




```
(ajuste2=auto.arima(AirPassengers, max.p=0, d=1, max.P=0, D=1, step=FALSE, lambda=0))
forecast(ajuste2,h=24)
plot(forecast(ajuste2,h=24))
```


Forecasts from ARIMA(0,1,1)(0,1,1)[12]
200
400
600
800
1950
1952
1954
1956
1958
1960
1962