Tema 05: Metodología Box-Jenkins: Análisis de series temporales lineales usando R

Análisis estadístico de series económicas

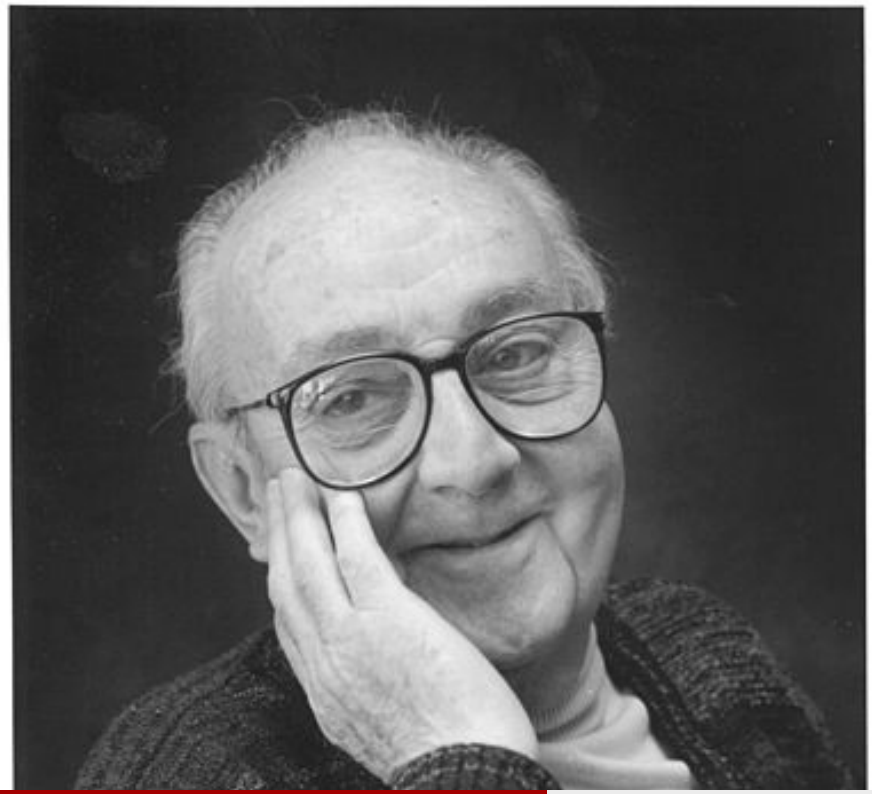
Xavier Barber

Departamento de Estadística, Matemáticas e Informática
I. Centro de Investigación Operativa
Universidad Miguel Hernández de Elche

09/Apr/2019

Licencia del Material

- Material adaptado de la Profesora Melody Ghahramanj (University of Winnipeg) http://www.ghahramani.ca/
- Basado en las enseñanzas del Profesor George E.P. Box



Leyendo la Serie temporal

La naturaleza de los datos `ts`

Los datos deberán ser:

- Continuos
- O, datos de conteos que se puedan aproximar mediante datos contínuos
  - por ejemplo, manchas solares mensuales
- Espaciados en el tiempo de forma regular
  - por ejemplo, semanalmente, trimestralmente, mensualmente o anualmente, etc.

Paquetes disponibles para Series Temporales

- Se suele utilizar el paquete **`astsa`** escrito por David Stotffer y el paquete **`forecast`** escrito por Rob Hyndman.
- Se recomienda como material de apoyo:
  - Time Series Analysis and Its Applications: With R Examples por Shumway and Stoffer
  - Forecasting: Principles and Practice por Rob J Hyndman and George Athanasopoulos ()

Leyendo datos de una serie

```
co2dat <- read.csv("co2.txt", sep="",
                   na.strings="-99.99")
```

```
    any x  decimal average interpolated  trend day
1  1958 3 1958.208  315.71       315.71 314.62  -1
2  1958 4 1958.292  317.45       317.45 315.29  -1
3  1958 5 1958.375  317.50       317.50 314.71  -1
4  1958 6 1958.458      NA       317.10 314.85  -1
5  1958 7 1958.542  315.86       315.86 314.98  -1
6  1958 8 1958.625  314.93       314.93 315.94  -1
```

```
     any  x  decimal average interpolated  trend day
729 2018 11 2018.875  408.02       408.02 410.02  24
730 2018 12 2018.958  409.07       409.07 409.77  30
731 2019  1 2019.042  410.83       410.83 410.54  27
732 2019  2 2019.125  411.75       411.75 410.90  27
```

Creando el objeto ts

```
co2 = ts(co2dat$interpolated, freq = 12, start = c(1958, 3))
```

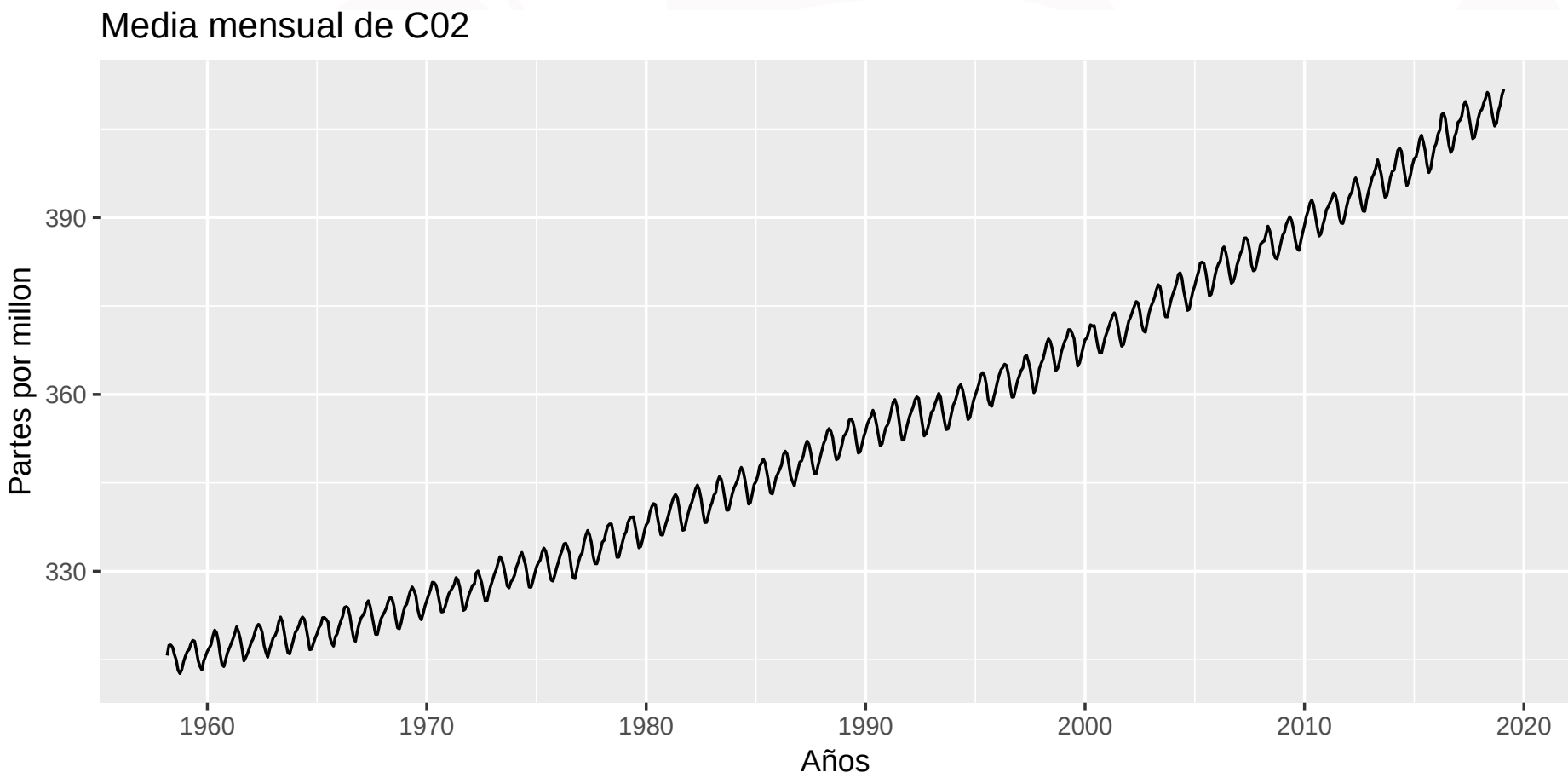
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct
1958			315.71	317.45	317.50	317.10	315.86	314.93	313.20	312.66
1959	315.62	316.38	316.71	317.72	318.29	318.15	316.54	314.80	313.84	313.26
1960	316.43	316.97	317.58	319.02	320.03	319.59	318.18	315.91	314.16	313.83
1961	316.93	317.70	318.54	319.48	320.58	319.77	318.57	316.79	314.80	315.38
1962	317.94	318.56	319.68	320.63	321.01	320.55	319.58	317.40	316.26	315.42
1963	318.74	319.08	319.86	321.39	322.25	321.47	319.74	317.77	316.21	315.99
1964	319.57	320.07	320.73	321.77	322.25	321.89	320.44	318.70	316.70	316.79
1965	319.44	320.44	320.89	322.13	322.16	321.87	321.39	318.81	317.81	317.30
1966	320.62	321.59	322.39	323.87	324.01	323.75	322.39	320.37	318.64	318.10
1967	322.07	322.50	323.04	324.42	325.00	324.09	322.55	320.92	319.31	319.31
1968	322.57	323.15	323.89	325.02	325.57	325.36	324.14	322.03	320.41	320.25
1969	324.00	324.42	325.64	326.66	327.34	326.76	325.88	323.67	322.38	321.78
1970	325.03	325.99	326.87	328.13	328.07	327.66	326.35	324.69	323.10	323.16
1971	326.17	326.68	327.18	327.78	328.92	328.57	327.34	325.46	323.36	323.57
1972	326.77	327.63	327.75	329.72	330.07	329.09	328.05	326.32	324.93	325.06
1973	328.54	329.56	330.30	331.50	332.48	332.07	330.87	329.31	327.51	327.18
1974	329.35	330.71	331.48	332.65	333.20	332.16	331.07	329.12	327.32	327.28
1975	330.73	331.46	331.90	333.17	333.94	333.45	331.98	329.95	328.50	328.34
1976	331.59	332.75	333.52	334.64	334.77	334.00	333.06	330.68	328.95	328.75
1977	332.66	333.13	334.95	336.13	336.93	336.17	334.88	332.56	331.29	331.27
1978	334.95	335.25	336.66	337.69	338.03	338.01	336.41	334.41	332.37	332.41
1979	336.14	336.69	338.27	338.95	339.21	339.26	337.54	335.75	333.98	334.19

Creando el objeto `ts`

- A veces los datos en formato serie temporal se registran de formar regular, y esto es muy importante que lo indiquemos en el commando `ts`.
- Recordad:
  - Semanal= 52
  - Mensual= 12
  - Trimestral= 4
  - Semestral= 6
- También se puede especificar el año/periodo de inicio:
  - freq=12, start(1995, 2) # febrero de 1995
  - freq=4, start(1995,2) # 2º trimestre de 1995

Dibujando la serie temporal

```
library(forecast)
autoplot(co2, xlab="Años", ylab="Partes por millon",
         main="Media mensual de CO2")
```



Asunciones necesarias para el ajuste

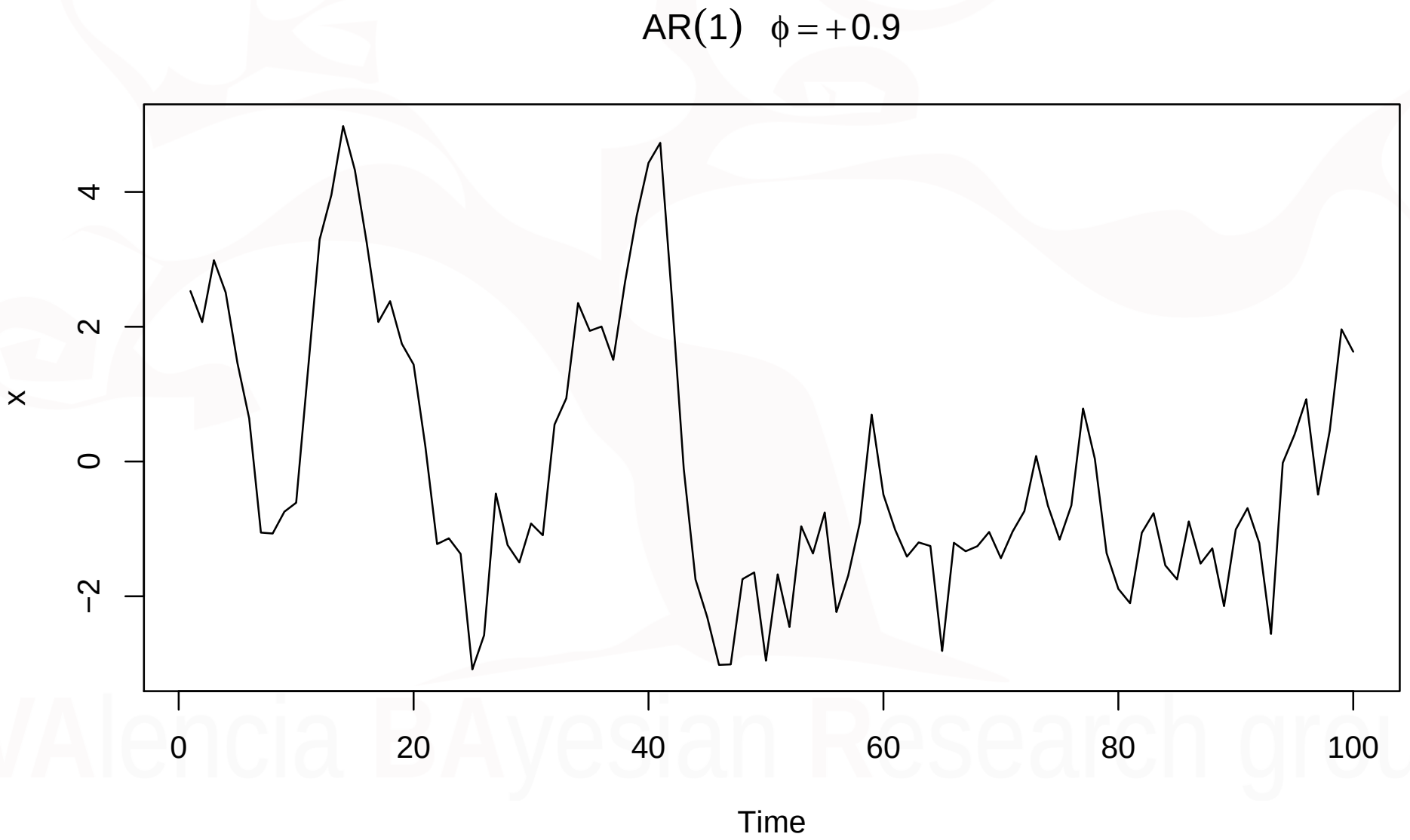
- Se necesita la condición de estacionariedad débil para las series:
  - Media Constante
  - La función de la covarianza sólo depende de los retardos
- La segunda condición implica una varianza constanste.
- Gráficamente: debemos observar esa varinza y esa media de forma constante.
  - Si se observa en el grádico, procederemos al análisis
  - En caso contrario, deberemos realizar transformaciones: Diferenciación
- El primer modelo a ajustar será un `ARMA(p,q)`

Simulando procesos ARMA(p,q) en R

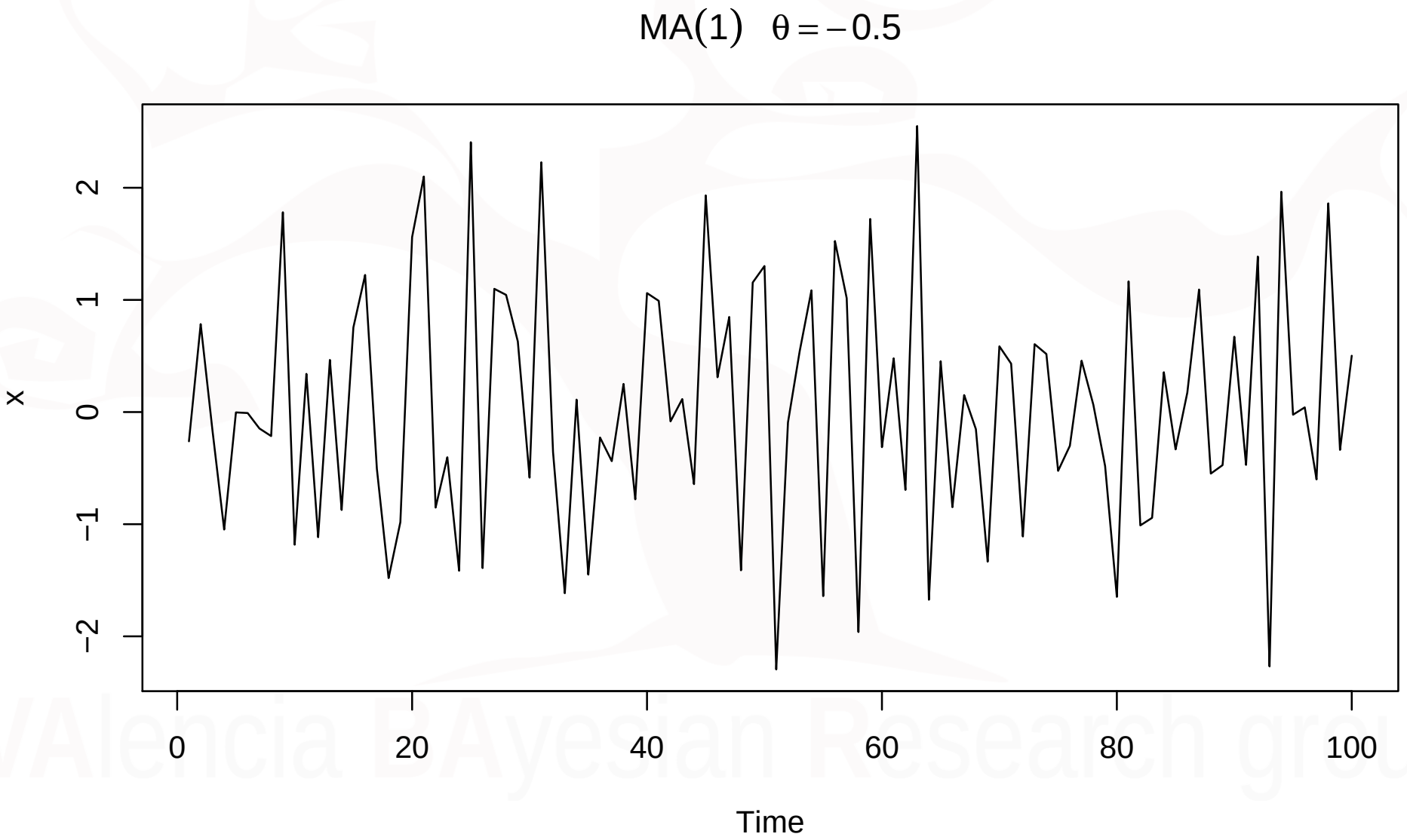
- Supongamos que queremos simular los siguientes procesos

```
# AR(1)
out1 = arima.sim(list(order = c(1, 0, 0), ar = 0.9), n = 100)

# MA(1)
out4 = arima.sim(list(order = c(0, 0, 1), ma = -0.5), n = 100)

# ARMA(1,1)
out6 = arima.sim(list(order = c(1, 0, 1), ar = 0.9, ma = -0.5),
    n = 100)
```

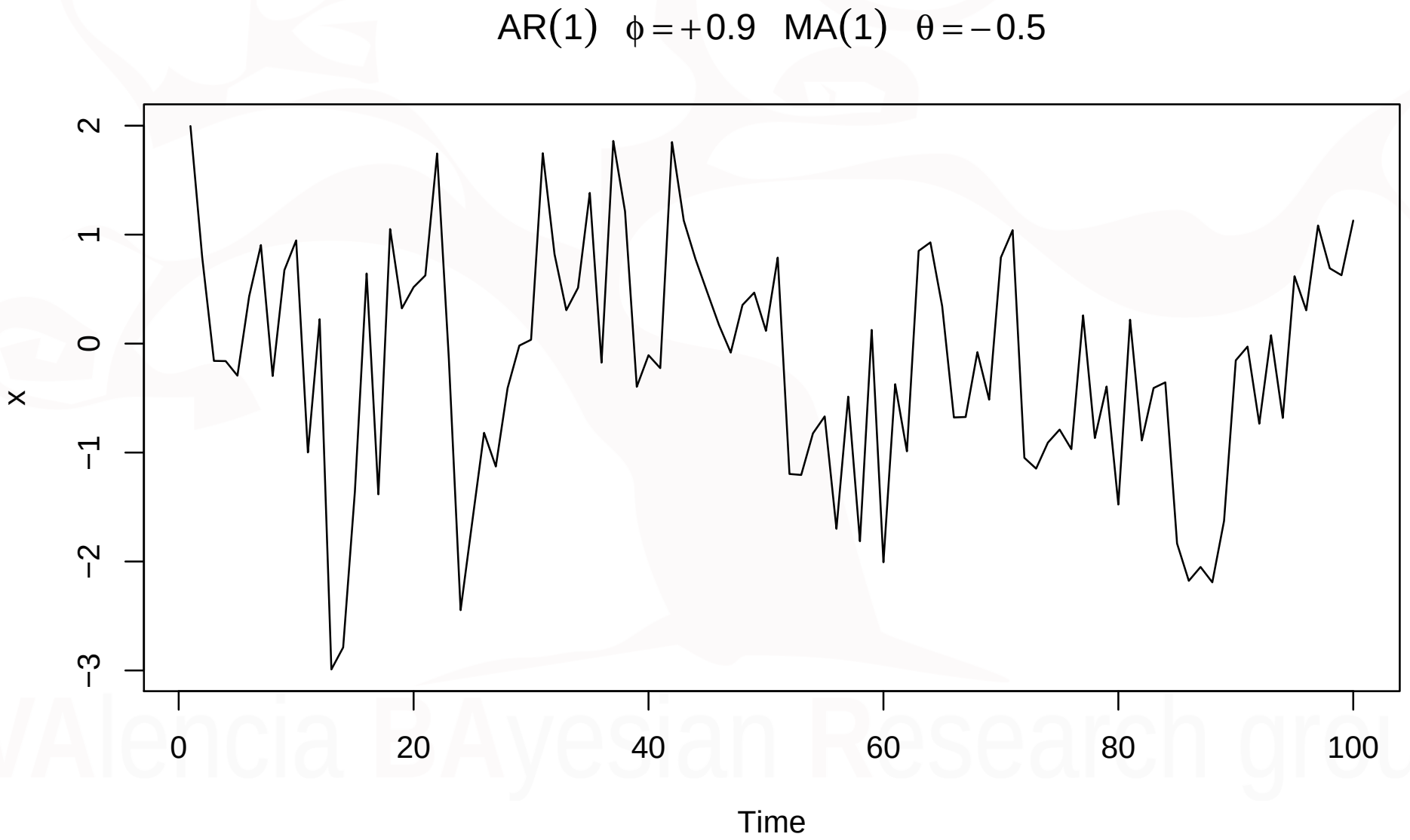
Simulando procesos ARMA(1,0) en R



Simulando procesos ARMA(0,1) en R



Simulando procesos ARMA(1,1) en R

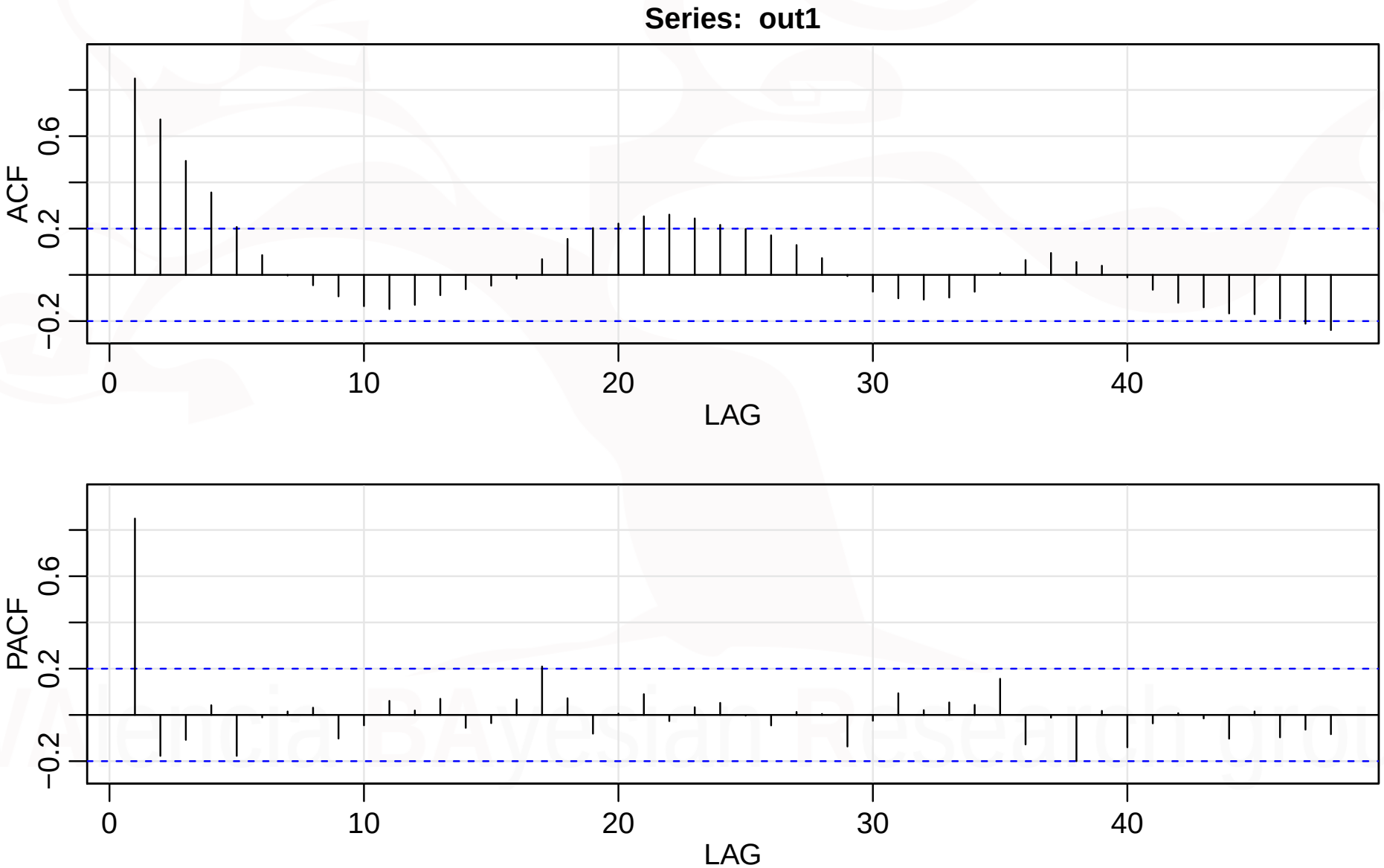


Modelo ARMA(p,q)

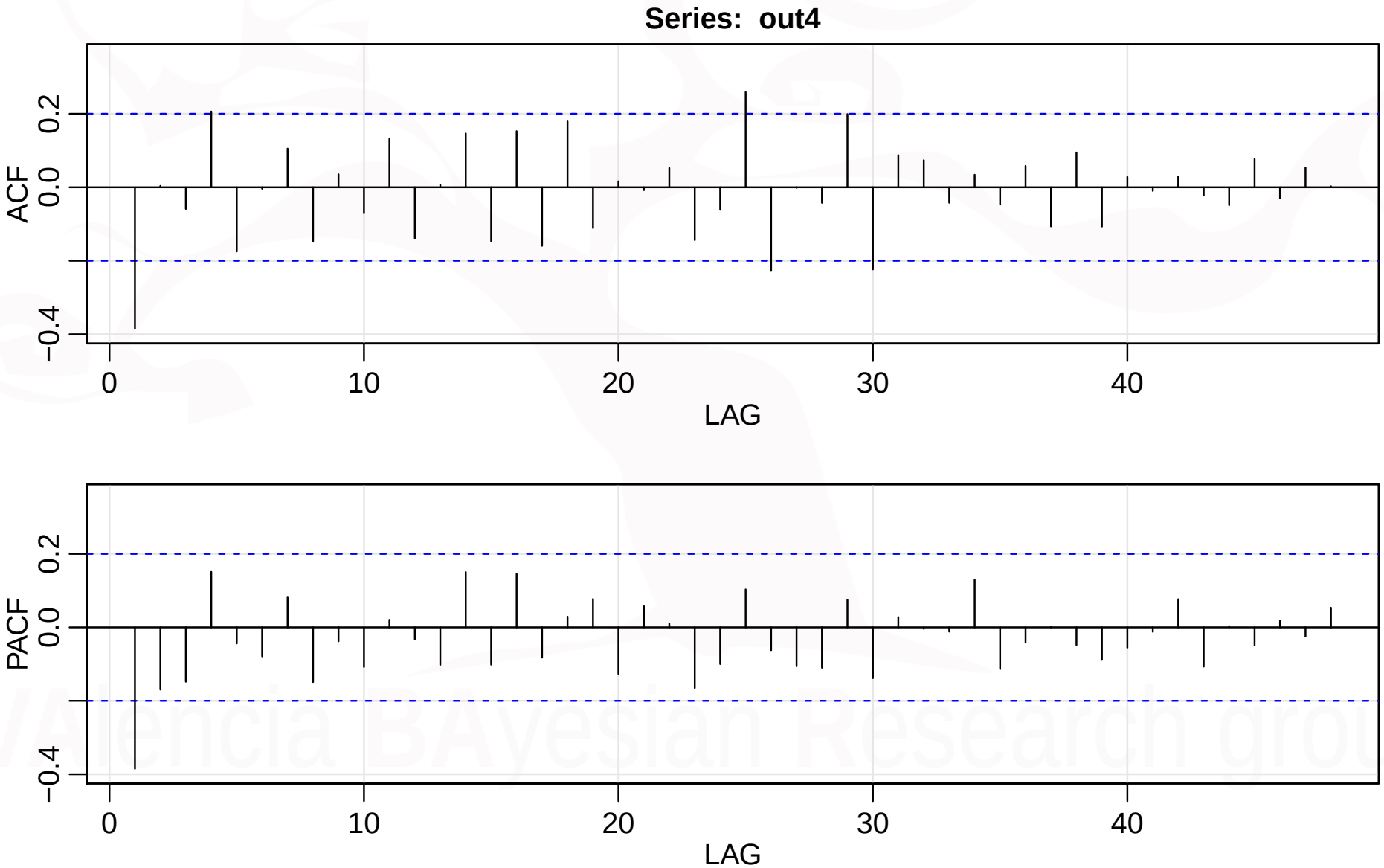
Identificando los modelos ARMA(1,0)

```
# install.packages('astsa')
library(astsa)
acf2(out1, 48)  #prints values and plots
```

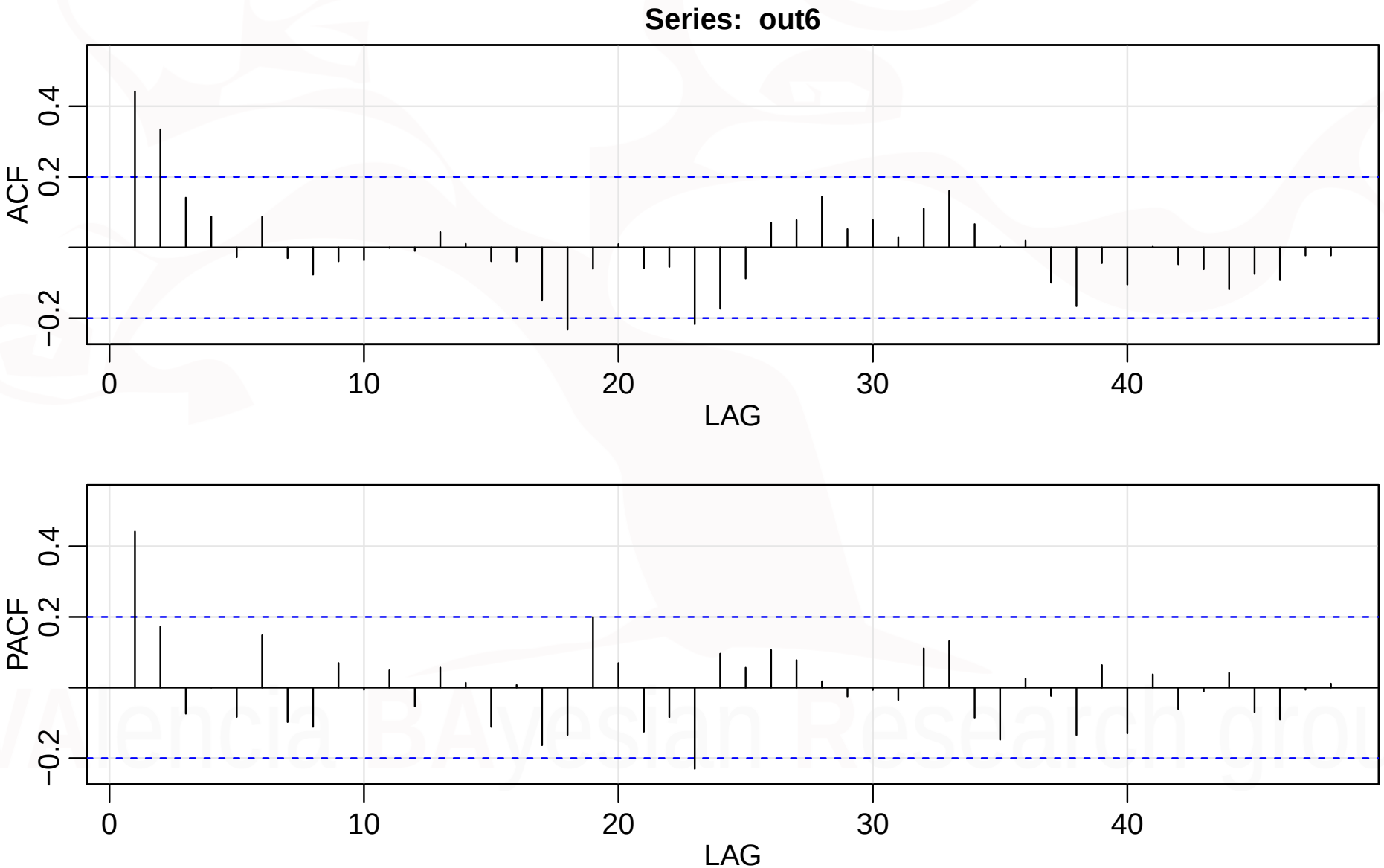
Identificando los modelos ARMA(1,0)



Identificando los modelos ARMA(0,1)



Identificando los modelos ARMA(1,1)



Identificación de los modelos ARMA(p,q)

	AR(p)	MA(q)	ARMA(p,q)
ACF	Colas largas	Se corta en q	Colas largas
PACF	Se corta en p	Colas largas	Colas largas

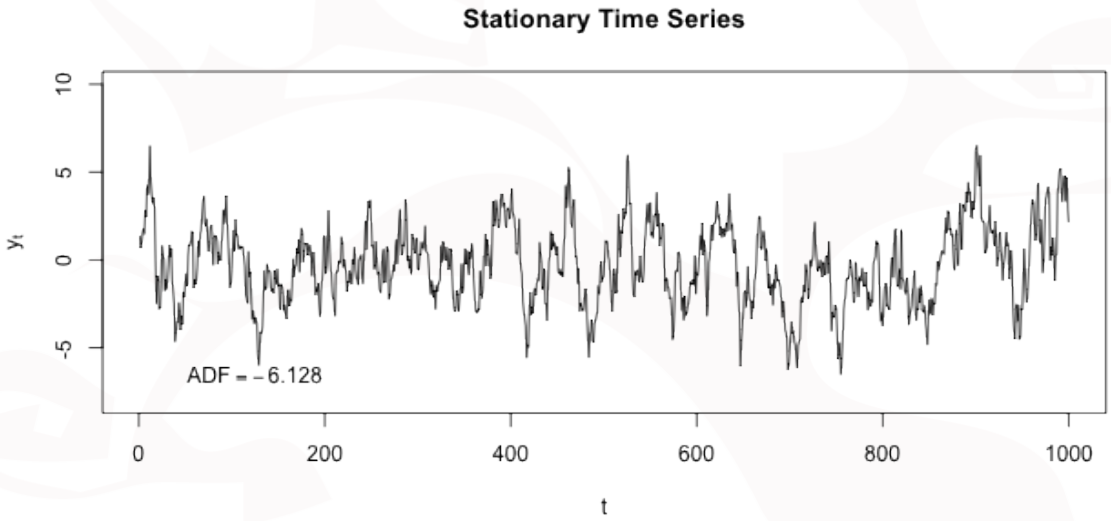
Trasnformando ts en R

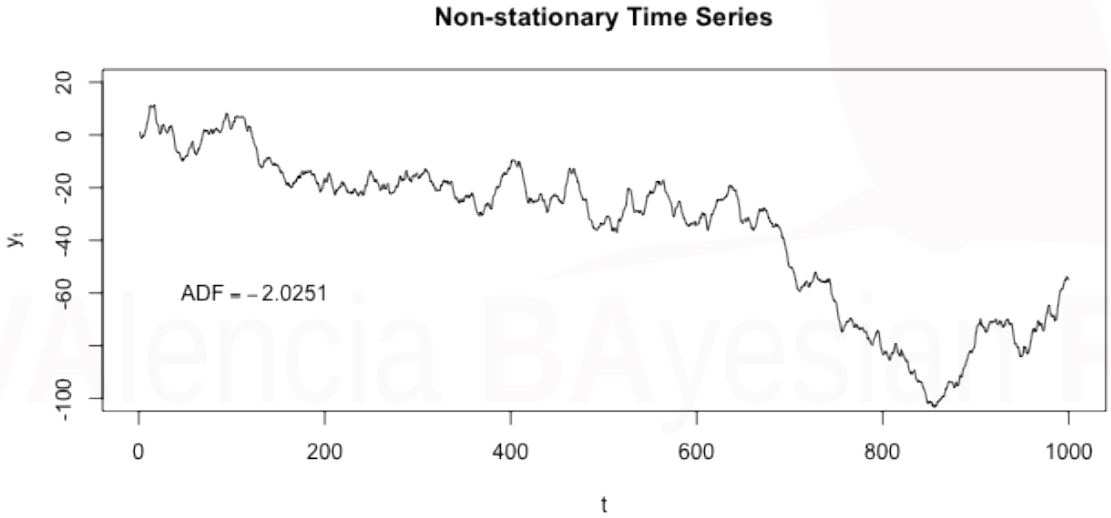
- Los modelos ARMA asumen que el proceso es de estacionariedad "débil".

- Un gráfico puede revelar la falta de esta estacionariedad, como por ejemplo:
  - La existencia de una tendencia, ya sea lineal, cuadráticas, cúbica, etc.
  - La varianza no es contante a lo largo del tiempo.

Entonces, se necesita trasnformar los datos para ajustar este modelo ARMA(p,q).

Series NO estacionarias





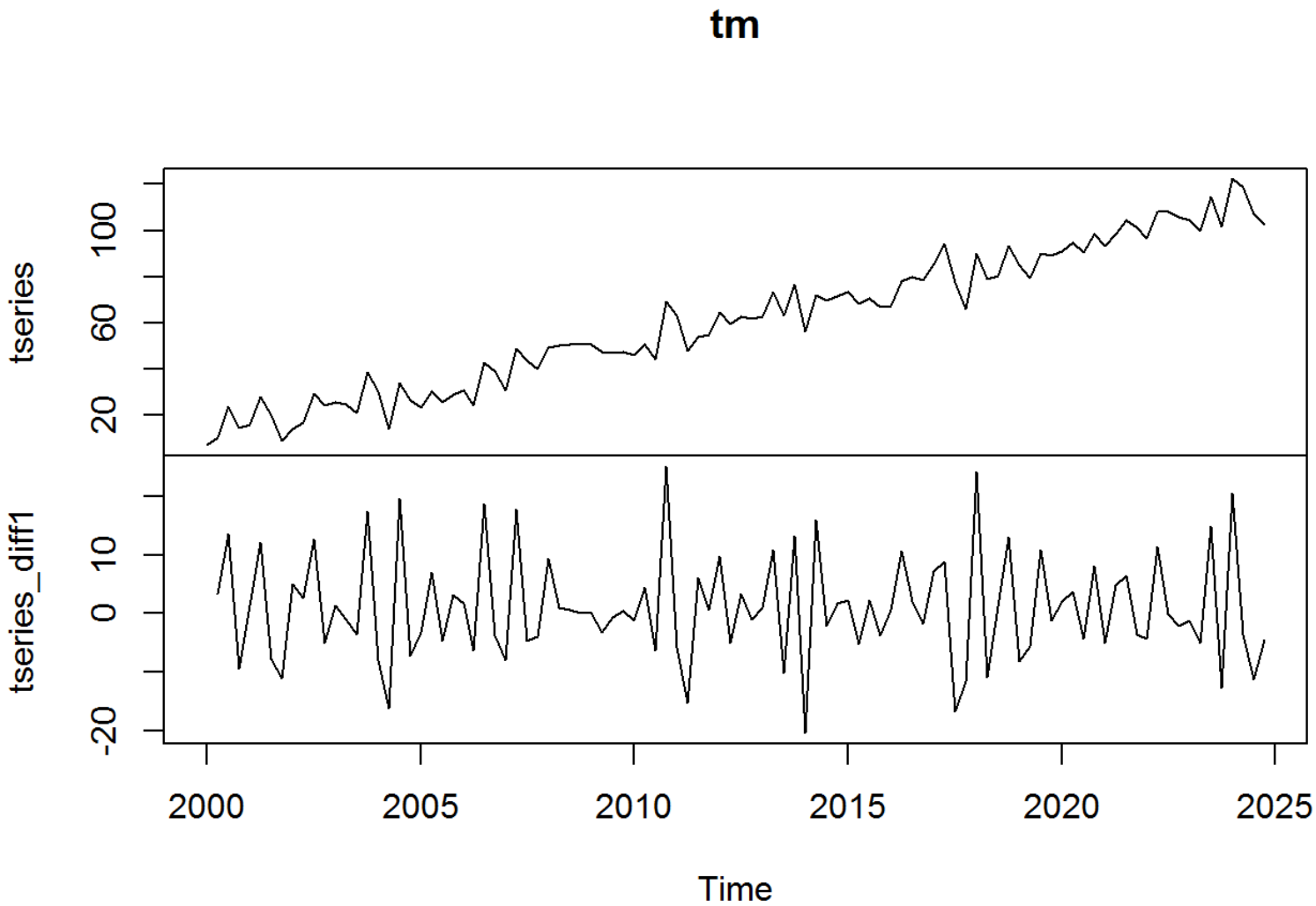
Trasnformando ts en R

Tendencias lineales

- Tomar la primera diferencia: $w_t = \bigtriangledown Y_t = y_t - y_{t-1}$. Y entonces ajustamos un ARMA al modelo w_t

- Ajustar el modelo $y_t = \beta_o + \beta_1 \times t + a_t$. Y entonces ajustar los residuos utilizando un modelo ARMA(p,q).

Diferenciando



Trasnformando ts en R

Tendencia cuadráticas

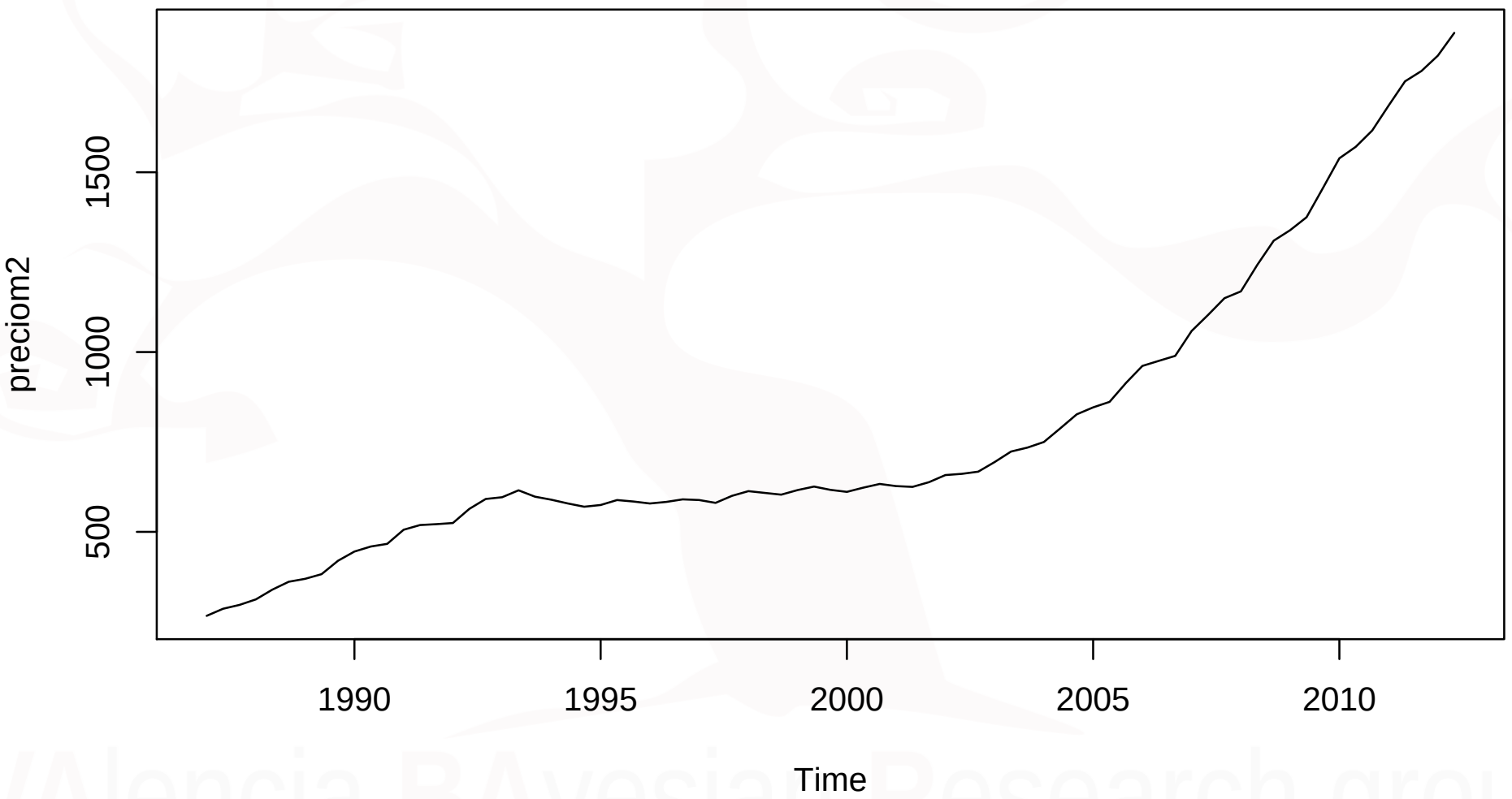
- Tomar la segunda diferencia:

$$v_t = \bigtriangledown^2 Y_t = \bigtriangledown(\bigtriangledown y_t) = y_t - 2y_{t-1} + y_{t-2}.$$

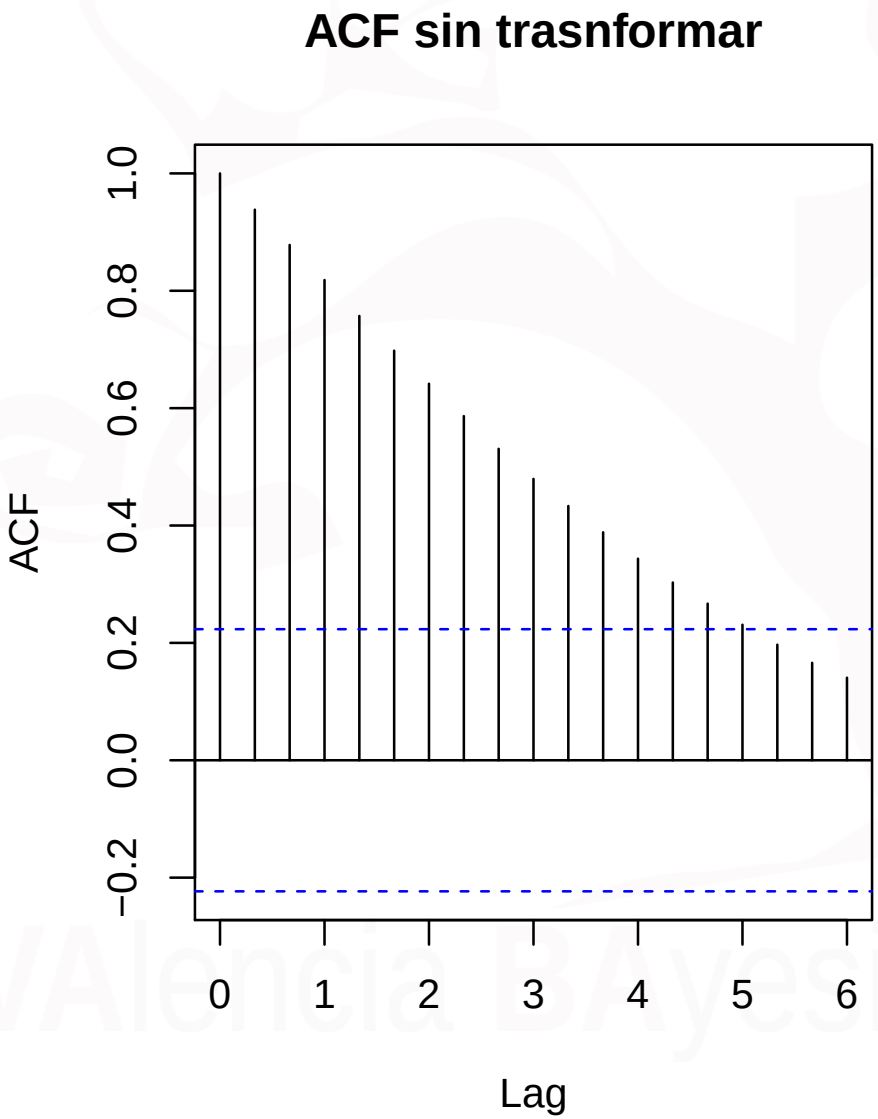
  Y entonces ajustamos un ARMA al modelo v_t

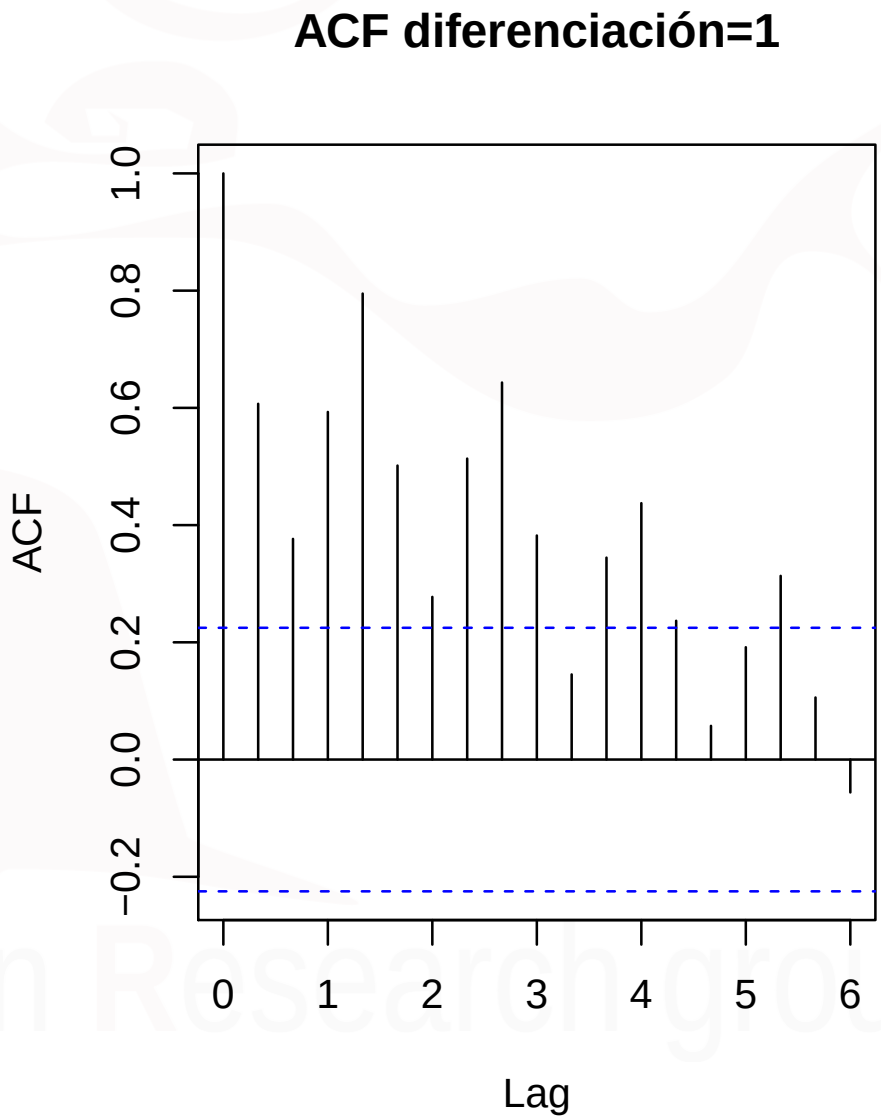
- Detrimendo: Ajustar el modelo $y_t = \beta_o + \beta_1 \times t + \beta_2 \times t^2 + a_t$. Y entonces ajustar los residuos utilizando un modelo ARMA.

Serie temporal con tendencia

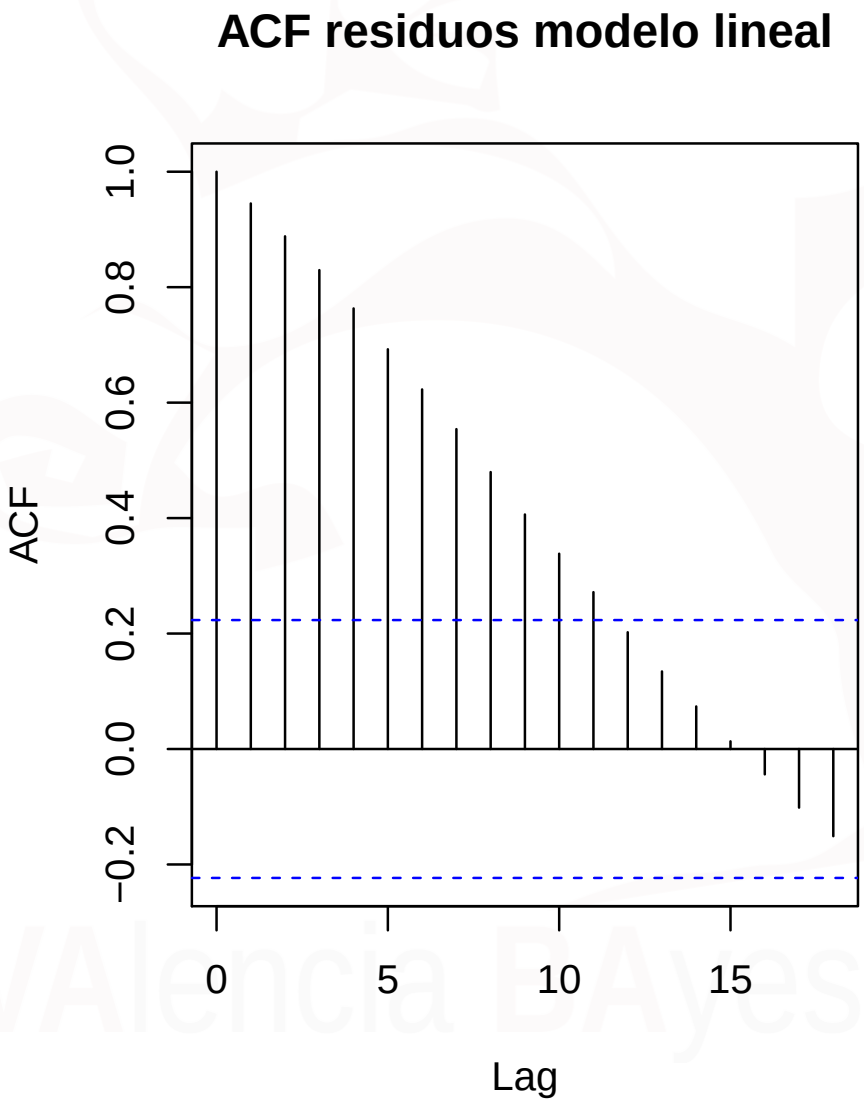


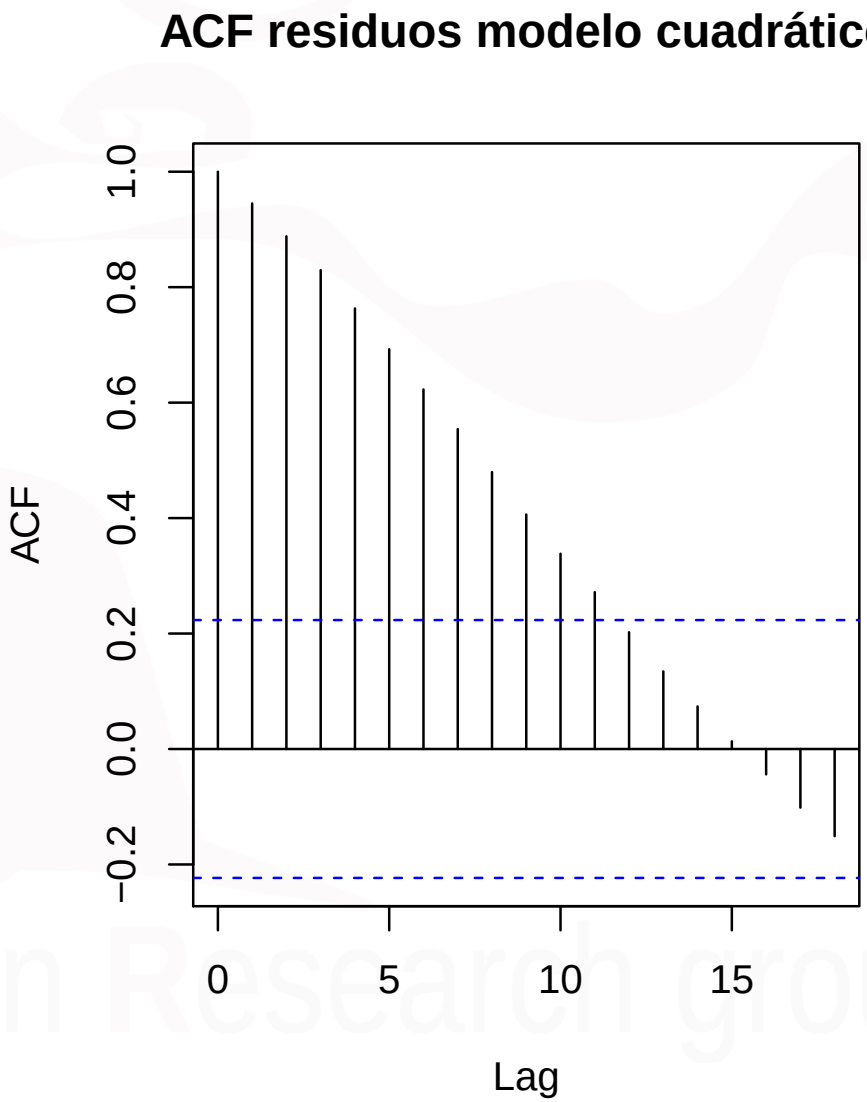
Serie temporal con tendencia



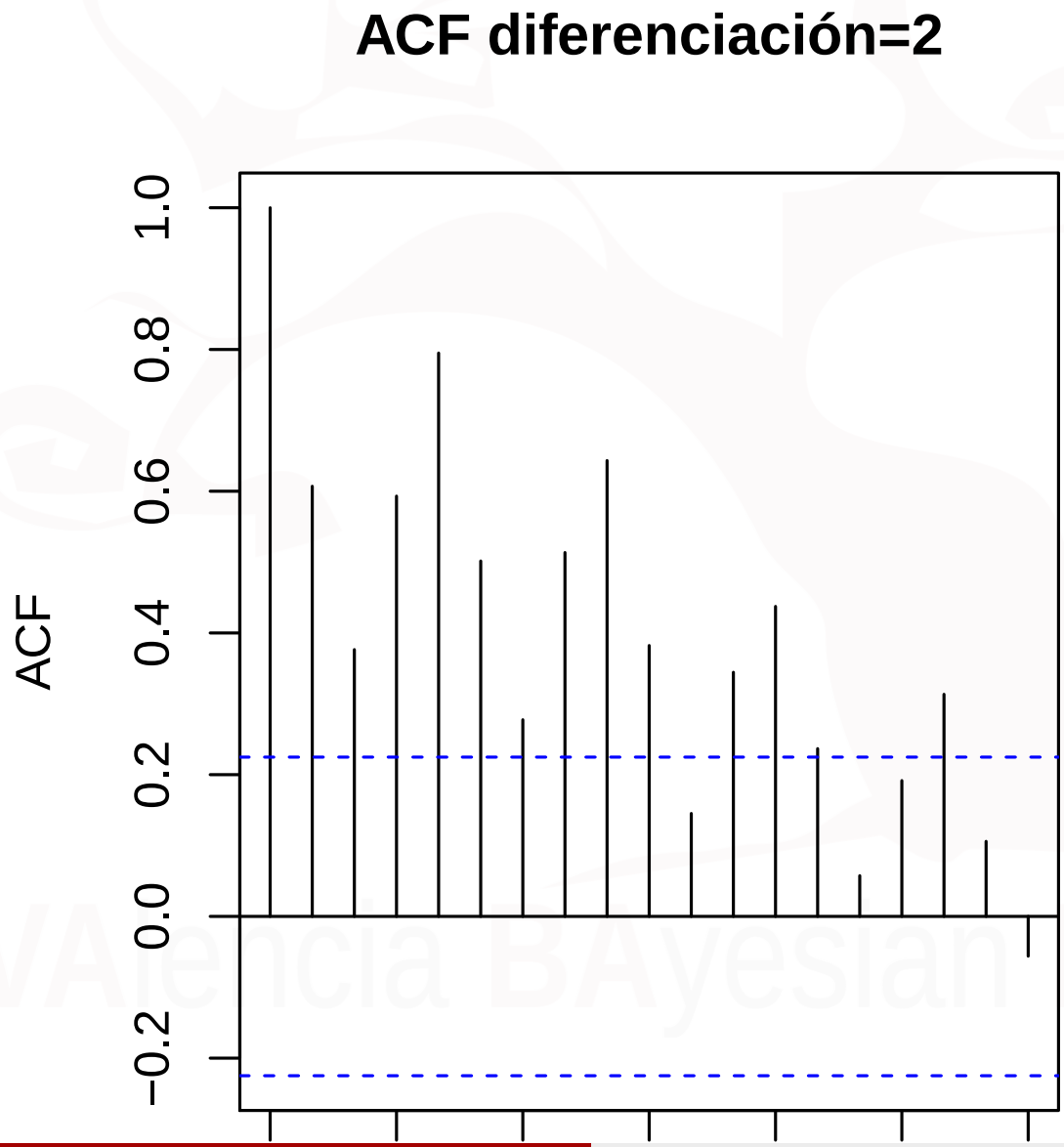


Serie temporal con tendencia

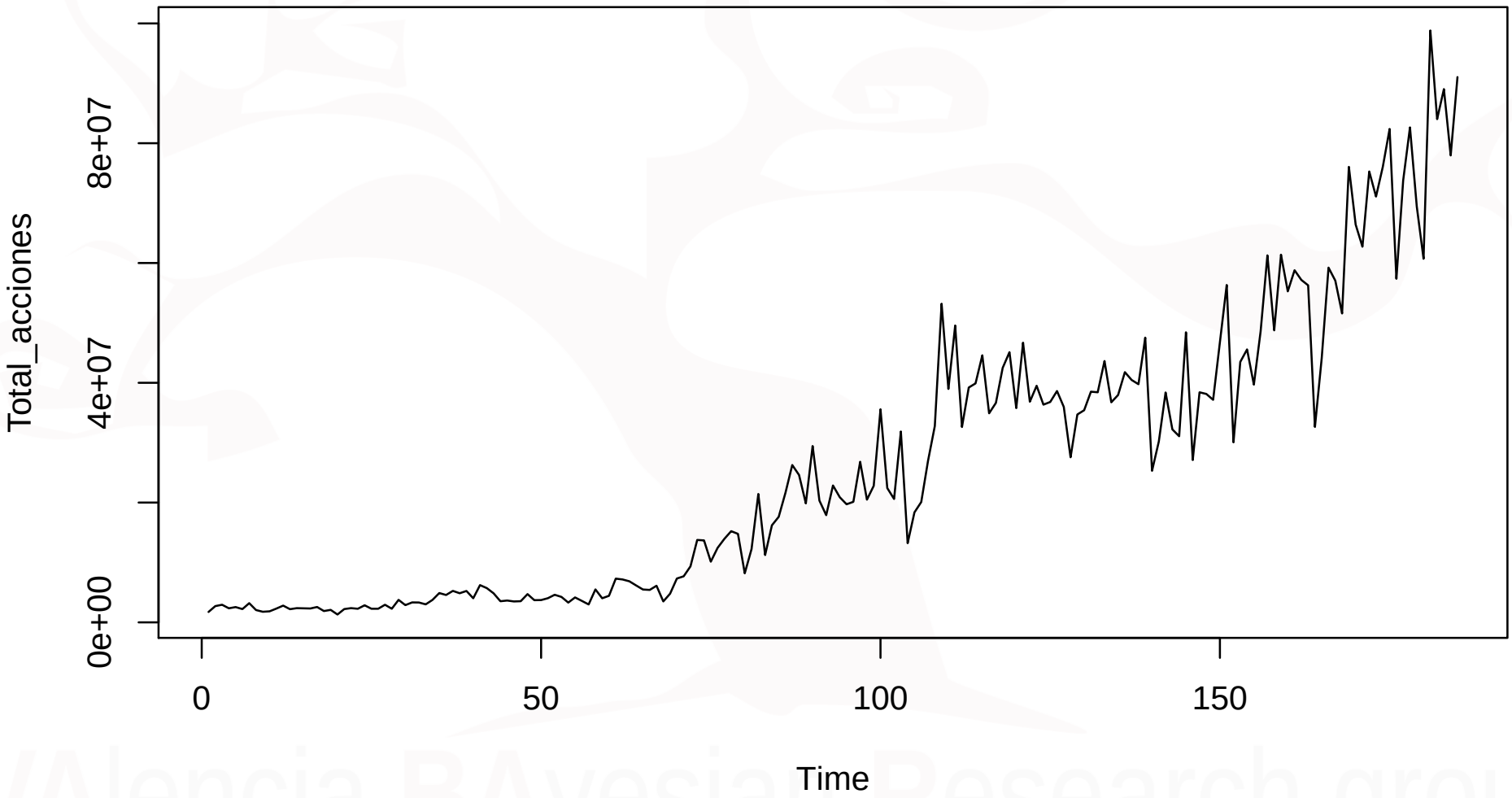




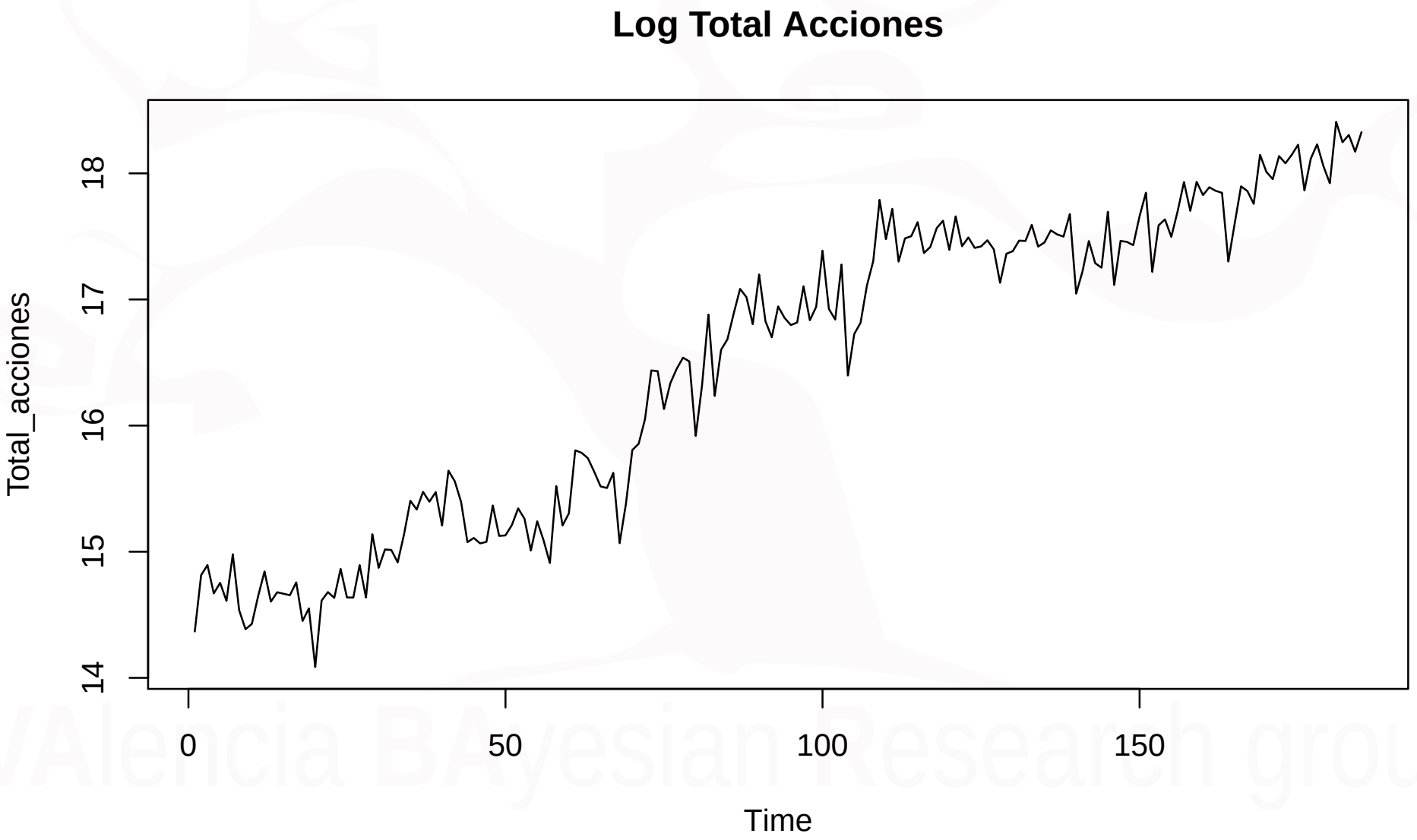
Serie temporal con tendencia



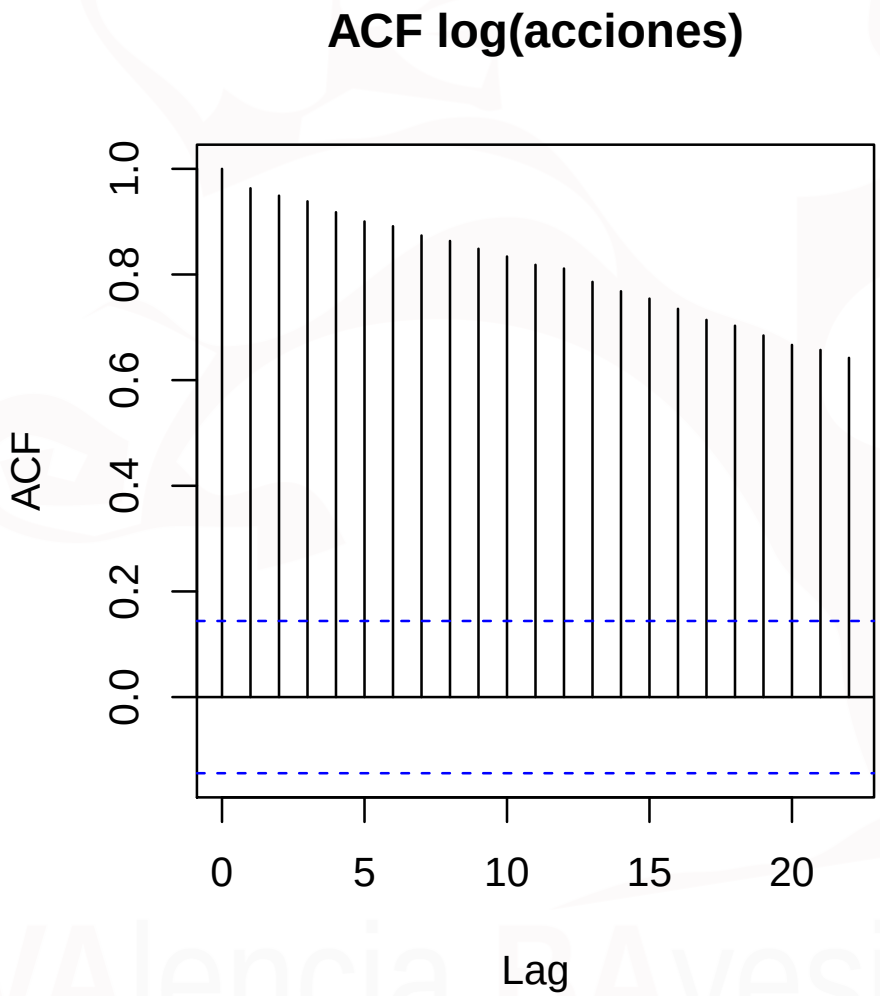
Serie con varianza no cte. y tendencia

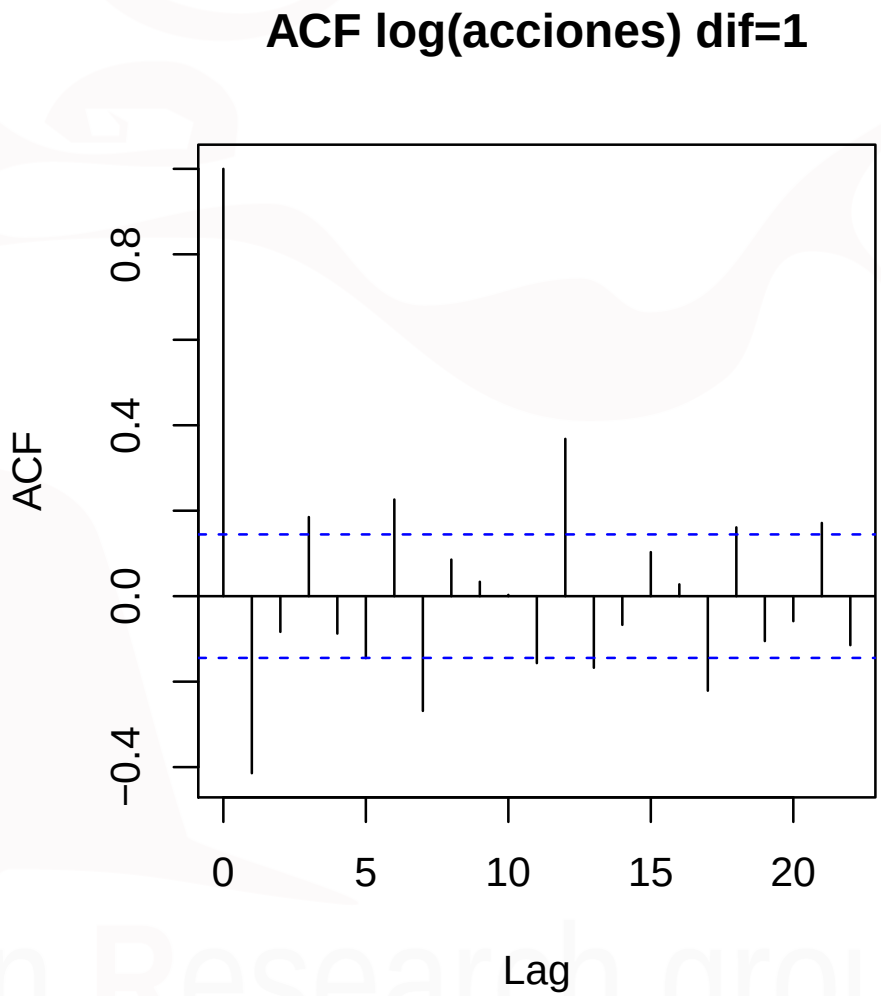


Serie con varianza no cte. y tendencia



Serie con varianza no cte. y tendencia





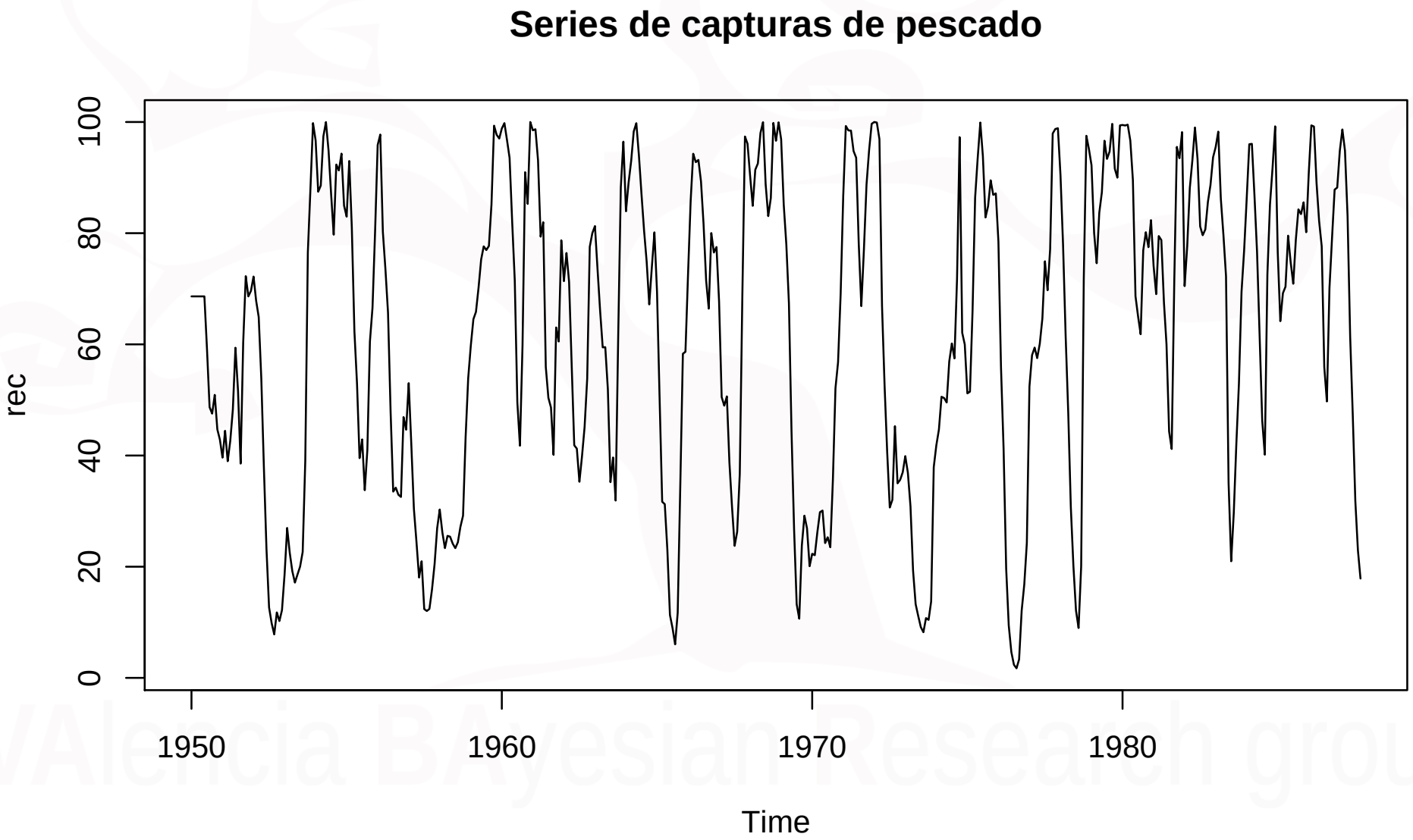
Modelos ARIMA

ARIMA(p,d,q)

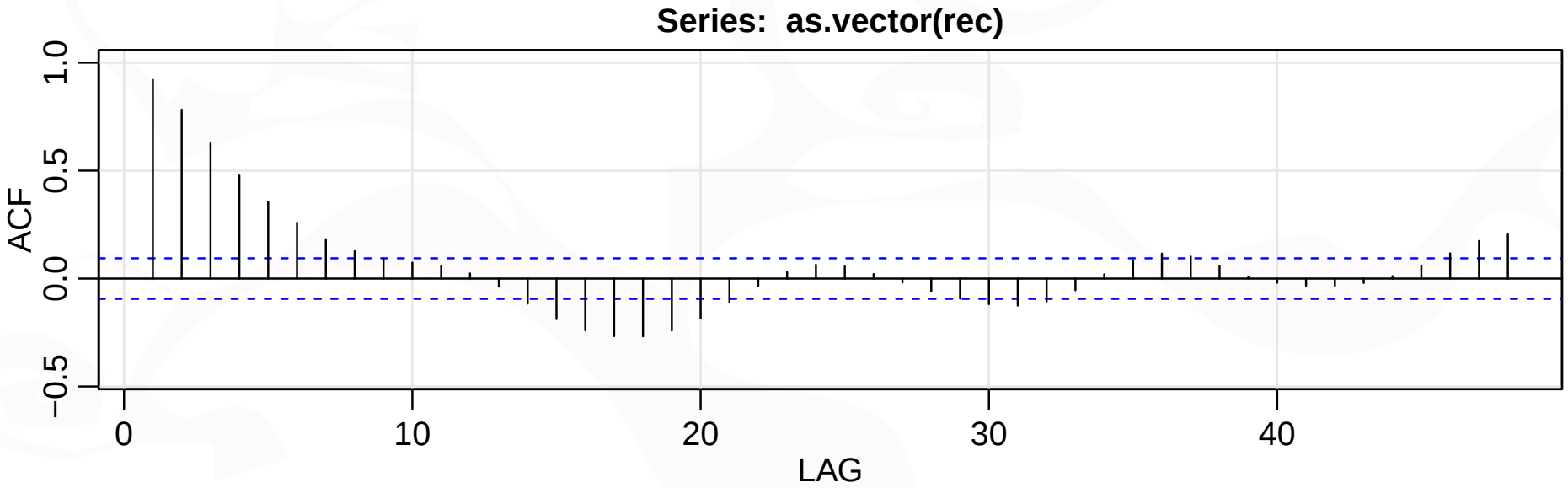
```
arima(x, order = c(0L, 0L, 0L),
      seasonal = list(order = c(0L, 0L, 0L),
      period = NA),
      xreg = NULL, include.mean = TRUE,
      transform.pars = TRUE,
      fixed = NULL, init = NULL,
      method = c("CSS-ML", "ML", "CSS"), n.cond,
      SSinit = c("Gardner1980", "Rossignol2011"),
      optim.method = "BFGS",
      optim.control = list(), kappa = 1e6)
```

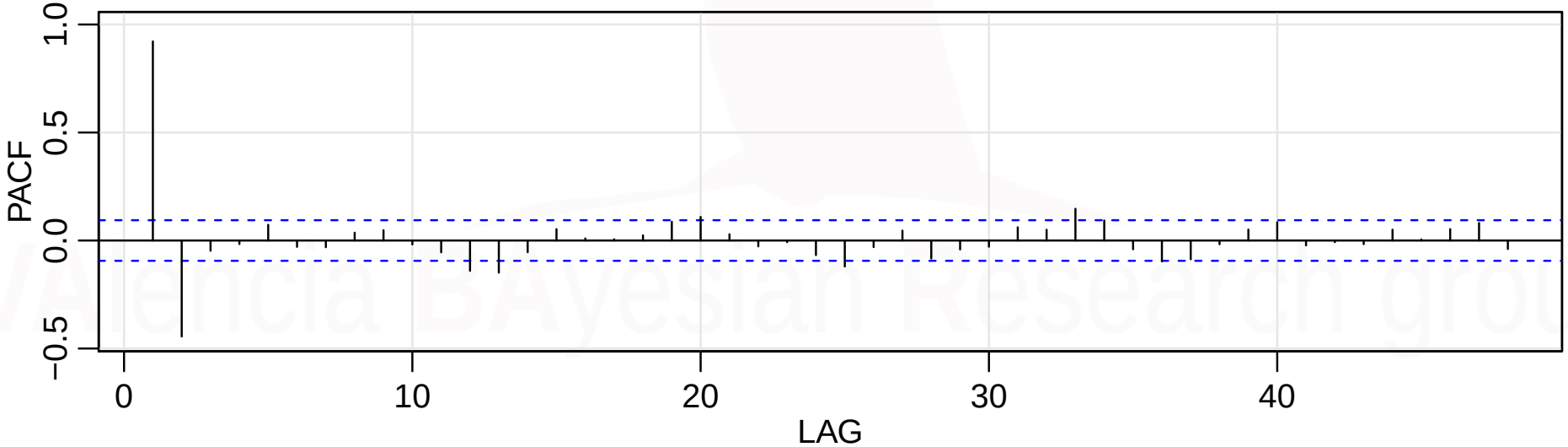
Se recomienda el uso de paquetes como **sarima** o **astsa** que pueden facilitar los gráficos y otros aspectos del análisis y ajuste.

ARIMA(p,d,q)


Series de capturas de pescado
rec
0
20
40
60
80
100
1950
1960
1970
1980
Time

ARIMA(p,d,q)


Series: as.vector(rec)
ACF
1.0
0.5
0.0
−0.5
0
10
20
30
40
LAG


PACF
1.0
0.5
0.0
−0.5
0
10
20
30
40
LAG

ARIMA(p,d,q)

```
fit1 <- arima(rec, order = c(2, 0, 0))
fit1
```

```
##
## Call:
## arima(x = rec, order = c(2, 0, 0))
##
## Coefficients:
##          ar1      ar2  intercept
##       1.3512  -0.4612    61.8585
## s.e.  0.0416   0.0417     4.0039
##
## sigma^2 estimated as 89.33:  log likelihood = -1661.51,  aic = 3331.02
```

ARIMA(p,d,q)

- El "intercepto" en el **arima** es la estimación de la media
- El modelo ajustado será pues:

$$Y_t - 61.68 = 1.35(Y_{t-1} - 61.86) - 0.46(Y_{t-2} - 61.86) + \hat{a}_t$$

Estacionalidad

SARIMA(P,D,Q)

- Cuando los datos muestran un comportamiento **estacional** y esto queda patente también en el ACF, entonces incorporaremos al modelo ajustado esta componente utilizando los miodelos `SARIMA(P,D,Q)`

$$ARIMA(p,q,d) \times SARIMA(P,Q,D)_S$$

ARIMA(p,d,q) y SARIMA(P,D,Q)

```
sarima(xdata, p, d, q, P = 0, D = 0, Q = 0, S = -1,
       details = TRUE, xreg=NULL, Model=TRUE,
       tol = sqrt(.Machine$double.eps),
       no.constant = FALSE)
```

```
p: AR order (must be specified)
d: difference order (must be specified)
q: MA order (must be specified)
P: SAR order; use only for seasonal models
D: seasonal difference; use only for seasonal models
Q: SMA order; use only for seasonal models
S: seasonal period; use only for seasonal models
```

SARIMA(P,D,Q)

La opción **nop.constant**:

- Controla si el modelo SARIMA(P,D,Q) incluye la constante en el modelo
- En particular, si no hay difefrenciación (d=0 and D=0) se obtiene al estimaciónd de la media.
- Si hay diferenciación de orden 1 (d=1 o D=1, pero no ambas), se incluye un término constante en modelo
- lo dicho anteriomente se obvia si el modelo inclute `no.constant=TRUE`.

SARIMA(P,D,Q)

- La idea es que la diferenciación total sea de primer orden (d+D≤ 1)
- Trabajaremos pues con modelos del estilo:
  - ARIMA(0,0,1) $\times$ SARIMA(0,1,1)$_1$2

Test de estacionariedad

La Prueba de Dickey-Fuller busca determinar la existencia o no de raíces unitarias en una serie temporal.

H_0: No hay estacionariedad

H_a: Existe estacionariedad

```
##
##  Augmented Dickey-Fuller Test
##
## data:  rec
## Dickey-Fuller = -4.0878, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

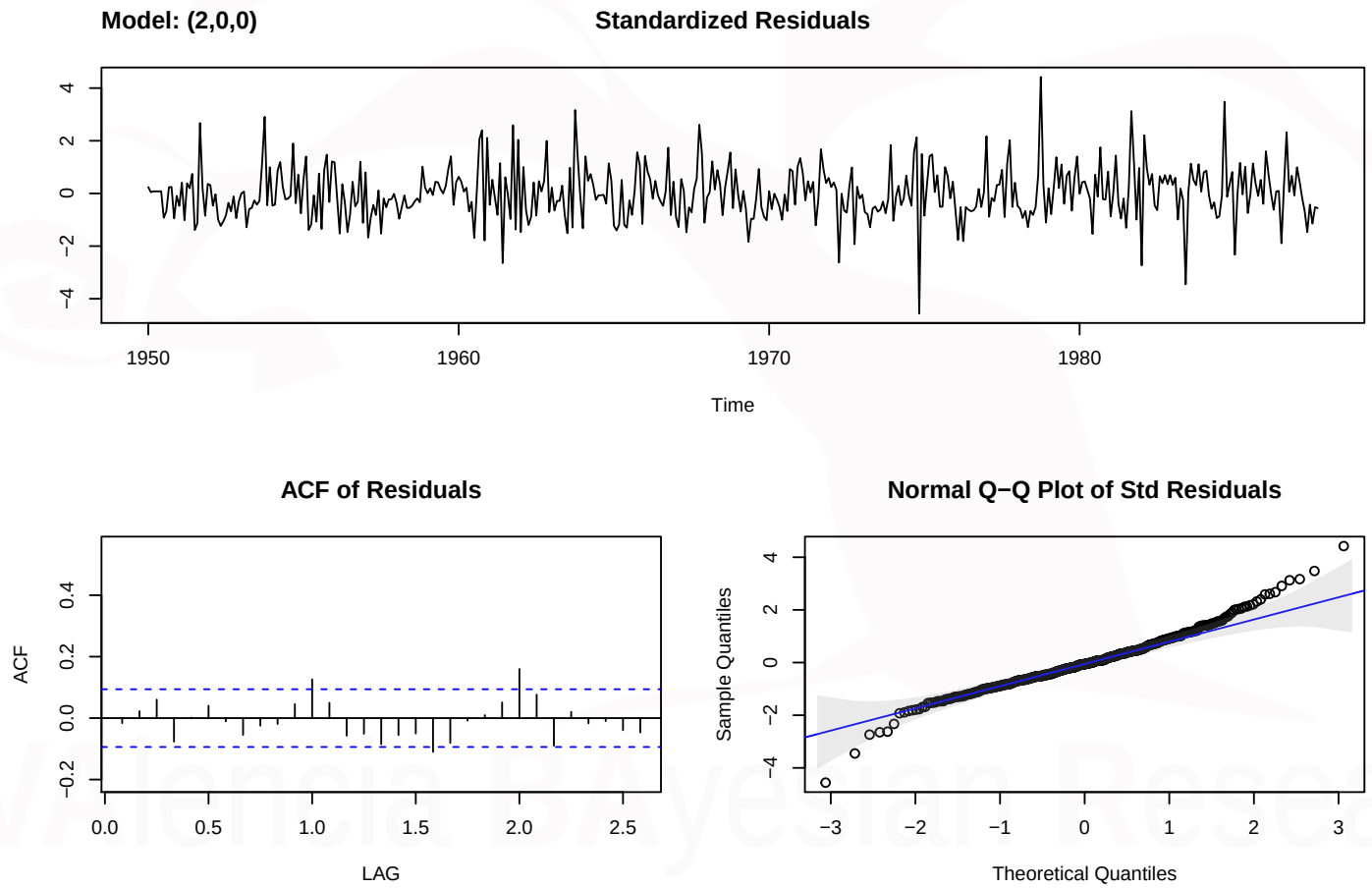
Test de Estacionariedad-Tendencia

H_0: Serie estacionaria con una tendencia o nivel

```
##
##  KPSS Test for Trend Stationarity
##
## data:  serie
## KPSS Trend = 0.46997, Truncation lag parameter = 4, p-value = 0.01
```

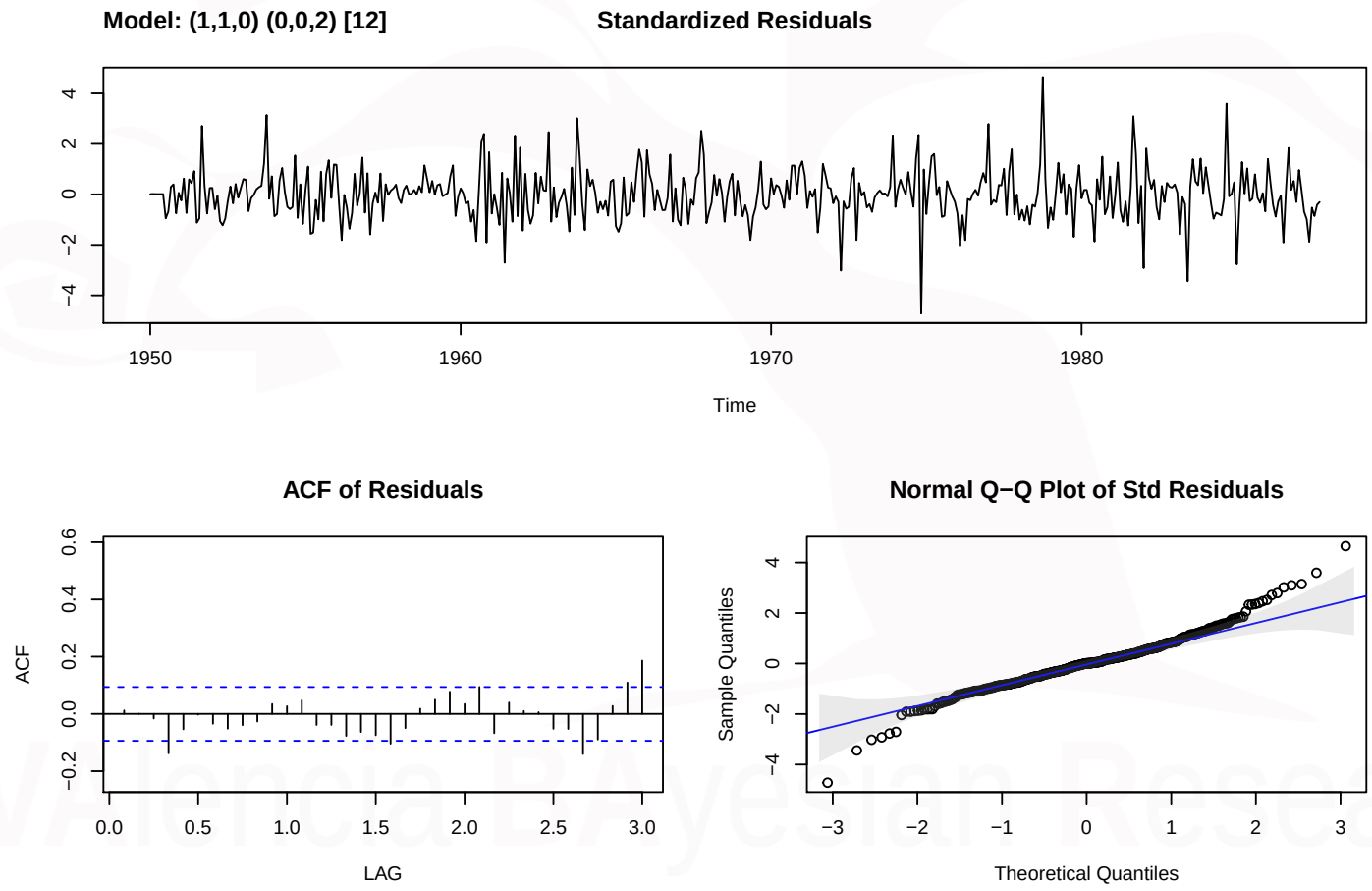
Ejemplo SARIMA

```
modelo <- sarima(rec, 2, 0, 0)
```



Ejemplo SARIMA

```
modelo <- sarima(rec, 1, 1, 0, 0, 0, 2, 12)
```



Test de Independencia de lso Residuos

La prueba de Ljung-Box se puede definir de la siguiente manera.

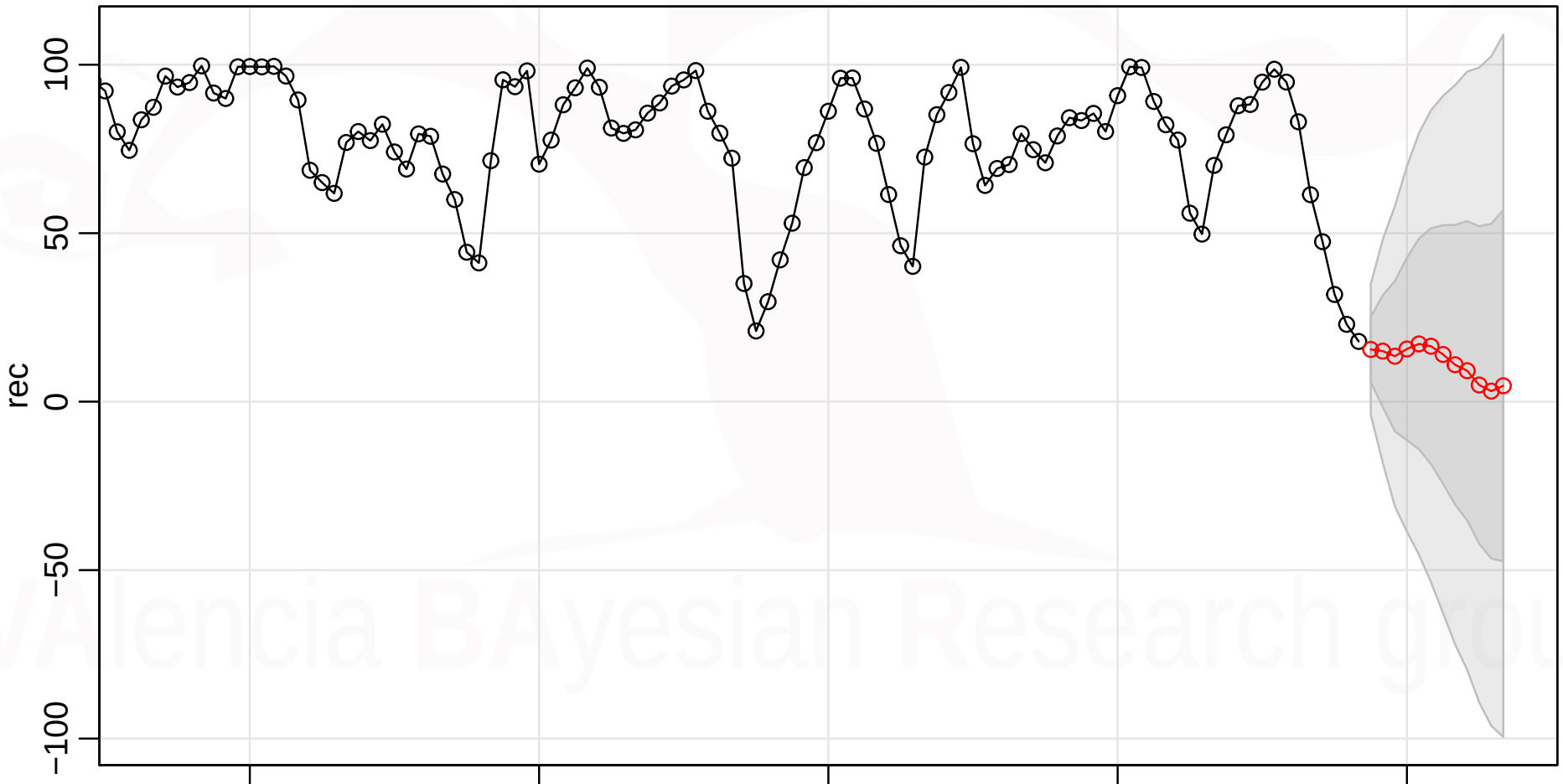
H_0: Los datos se distribuyen de forma independiente (es decir, las correlaciones en la población de la que se toma la muestra son 0, de modo que cualquier correlación observada en los datos es el resultado de la aleatoriedad del proceso de muestreo).

H_a: Los datos no se distribuyen de forma independiente.

```
    Box-Ljung test

data:  rec
X-squared = 1245.2, df = 25, p-value < 2.2e-16
```

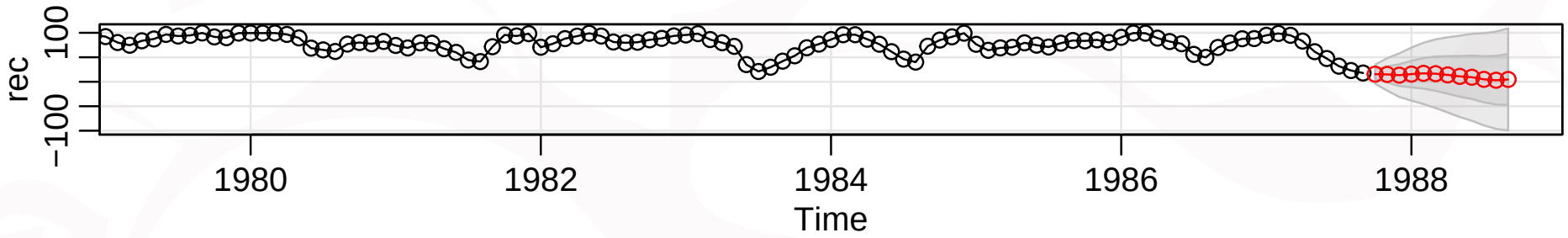
Predicción

```
# modelo2<-auto.arima(rec)
sarima.for(rec, 12, 1, 1, 0, 0, 0, 2, 12, plot.all = FALSE)
```



Predicción

```
# modelo2<-auto.arima(rec)
sarima.for(rec, 12, 1, 1, 0, 0, 0, 2, 12)
```




```
## $pred
##             Jan       Feb       Mar       Apr       May       Jun       Jul
## 1987
## 1988 15.629090 17.139372 16.447856 13.995275 10.953905  9.175640  4.938035
##             Aug       Sep       Oct       Nov       Dec
## 1987                      15.486908 14.996399 13.484424
## 1988  3.129710  4.709371
##
## $se
##             Jan       Feb       Mar       Apr       May       Jun       Jul
## 1987
## 1988 27.094743 31.283351 35.010316 38.389846 41.499602 44.393685 47.110906
##             Aug       Sep       Oct       Nov       Dec
## 1987                       9.709541 16.580948 22.275078
## 1988 49.679947 52.122597
```

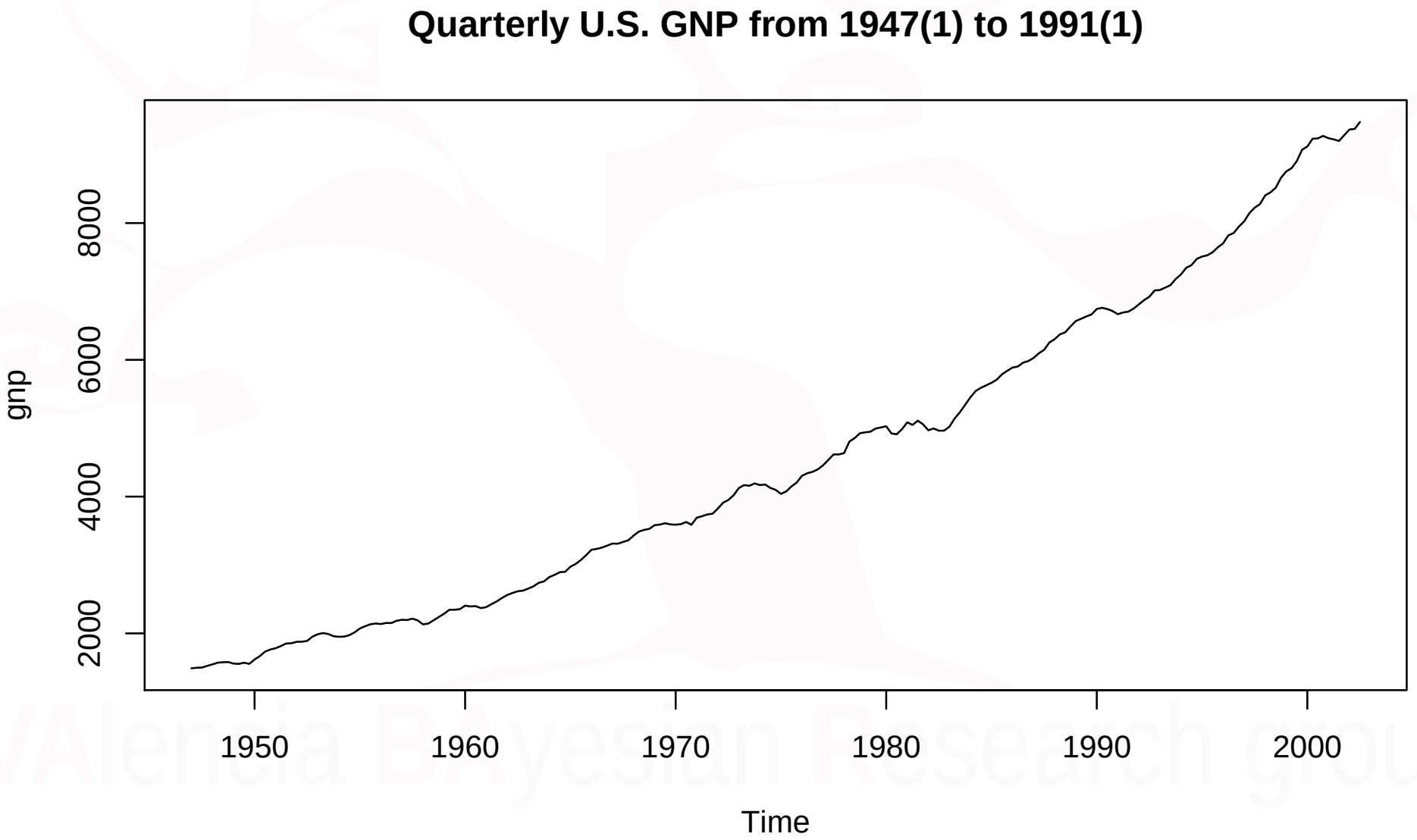
Ejemplo modelo **ARIMA**

US GNP Series:

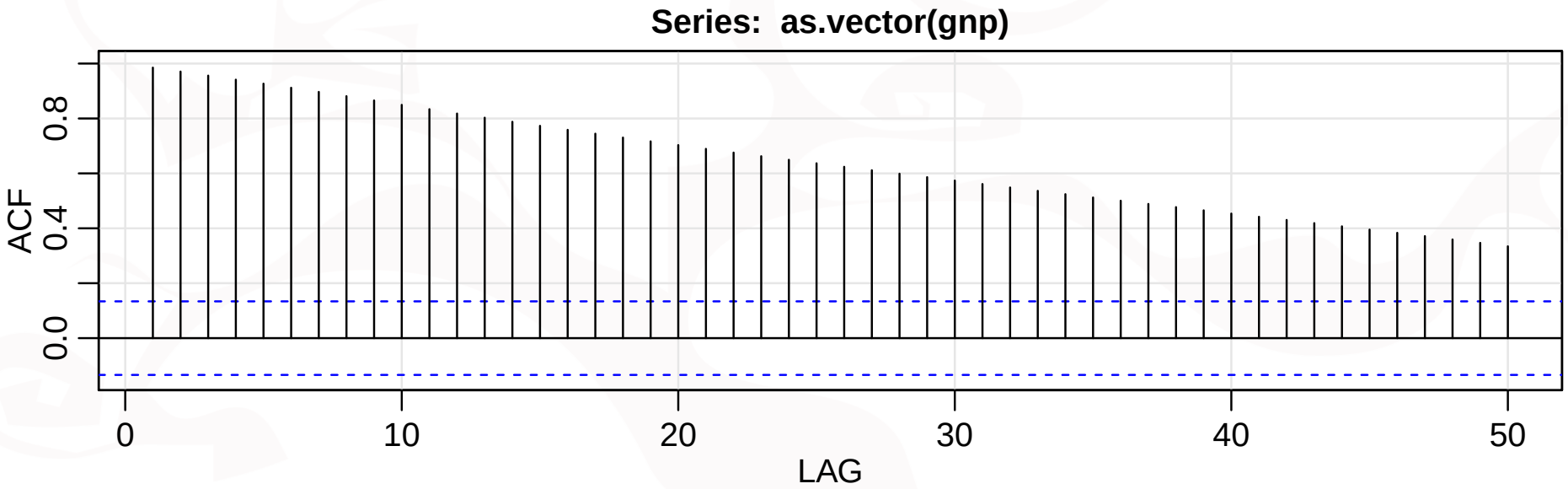
Análisis del producto interior bruto de Estados Unidos

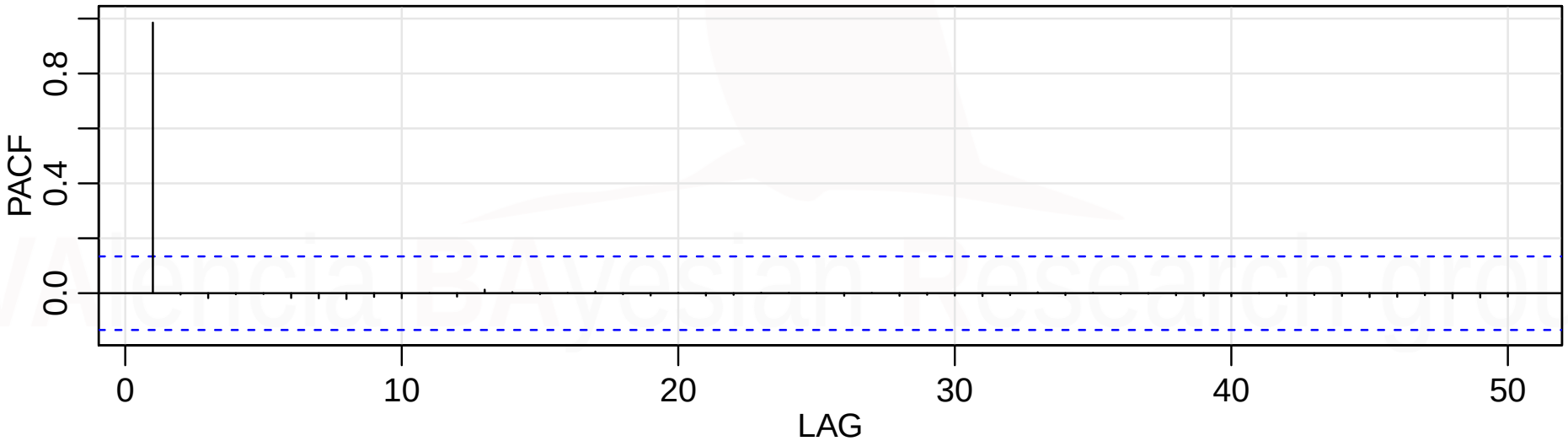
```
library(tseries)
library(astsa)
data(gnp)
plot(gnp)
title('Quarterly U.S. GNP from 1947(1) to 1991(1)')
acf2(as.vector(gnp), 50)
#estacionariedad
adf.test(rec, alternative="stationary", k=0)
# diferenciación de orden 1 (por no estacionaria)
plot(diff(gnp))
title('First Difference of U.S. GNP from 1947(1) to 1991(1)')
# diferenciación de orden 1 y logaritmo (por varianza no cons
gnpgr = diff(log(gnp)) # growth rate
plot(gnpgr)
title('First difference of the U.S. log(GNP) data')
```

US GNP Series:



US GNP: Exesivos retardos significativos





US GNP: test de estacionariedad

```
adf.test(gnp, alternative = "stationary", k = 0)
```

```
    Augmented Dickey-Fuller Test

data:  gnp
Dickey-Fuller = 0.069639, Lag order = 0, p-value = 0.99
alternative hypothesis: stationary
```

Claramente no es estacionaria.

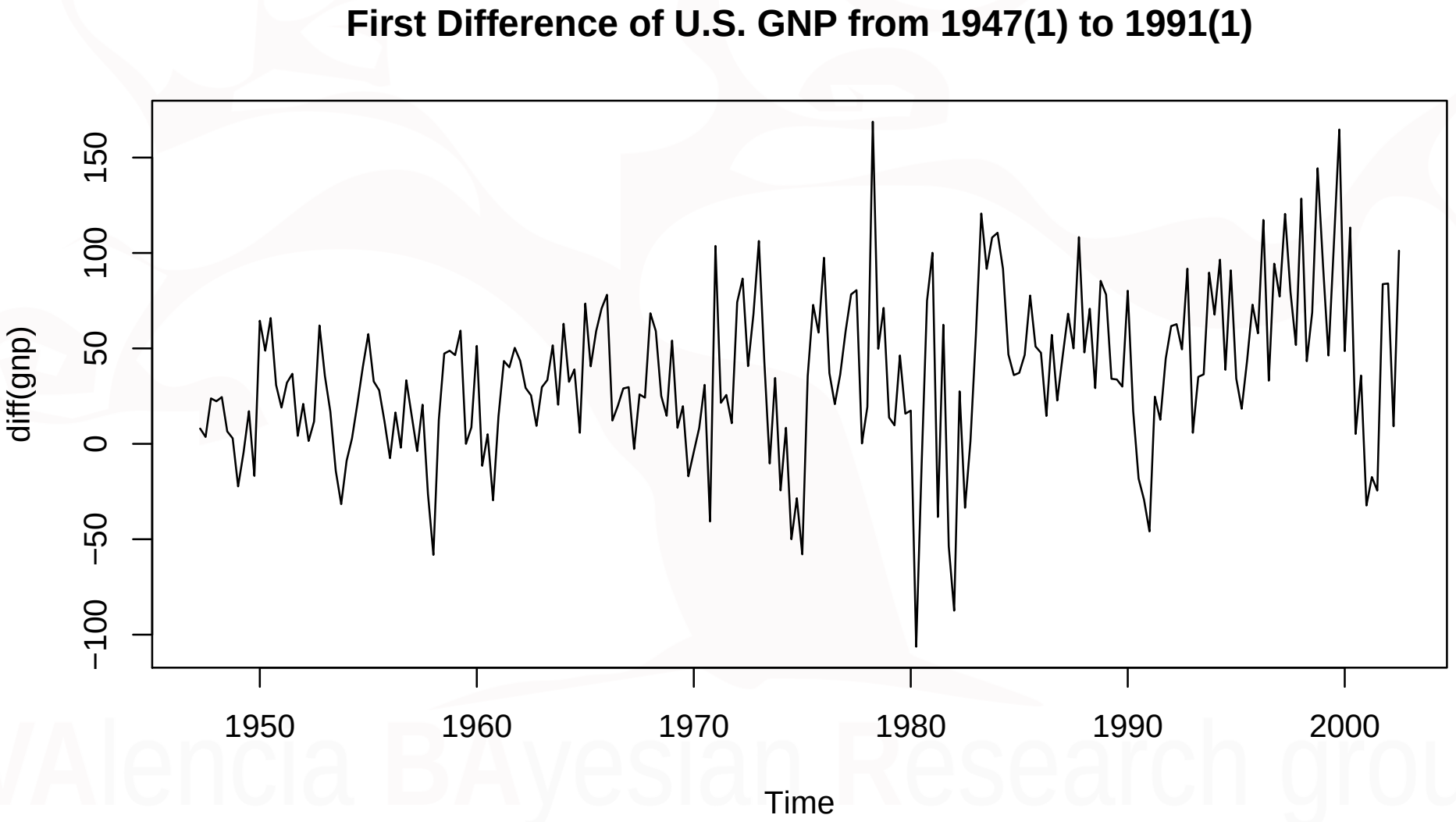
US GNP: test de estacionariedad-Tendencia

```
kpss.test(gnp, null = "Trend")
```

```
    KPSS Test for Trend Stationarity

data:  gnp
KPSS Trend = 0.94871, Truncation lag parameter = 4, p-value = 0.01
```

Este test responde a la pregunta de si la serie es estacionaria en torno a una tendencia.

US GNP: Dif. de orden 1

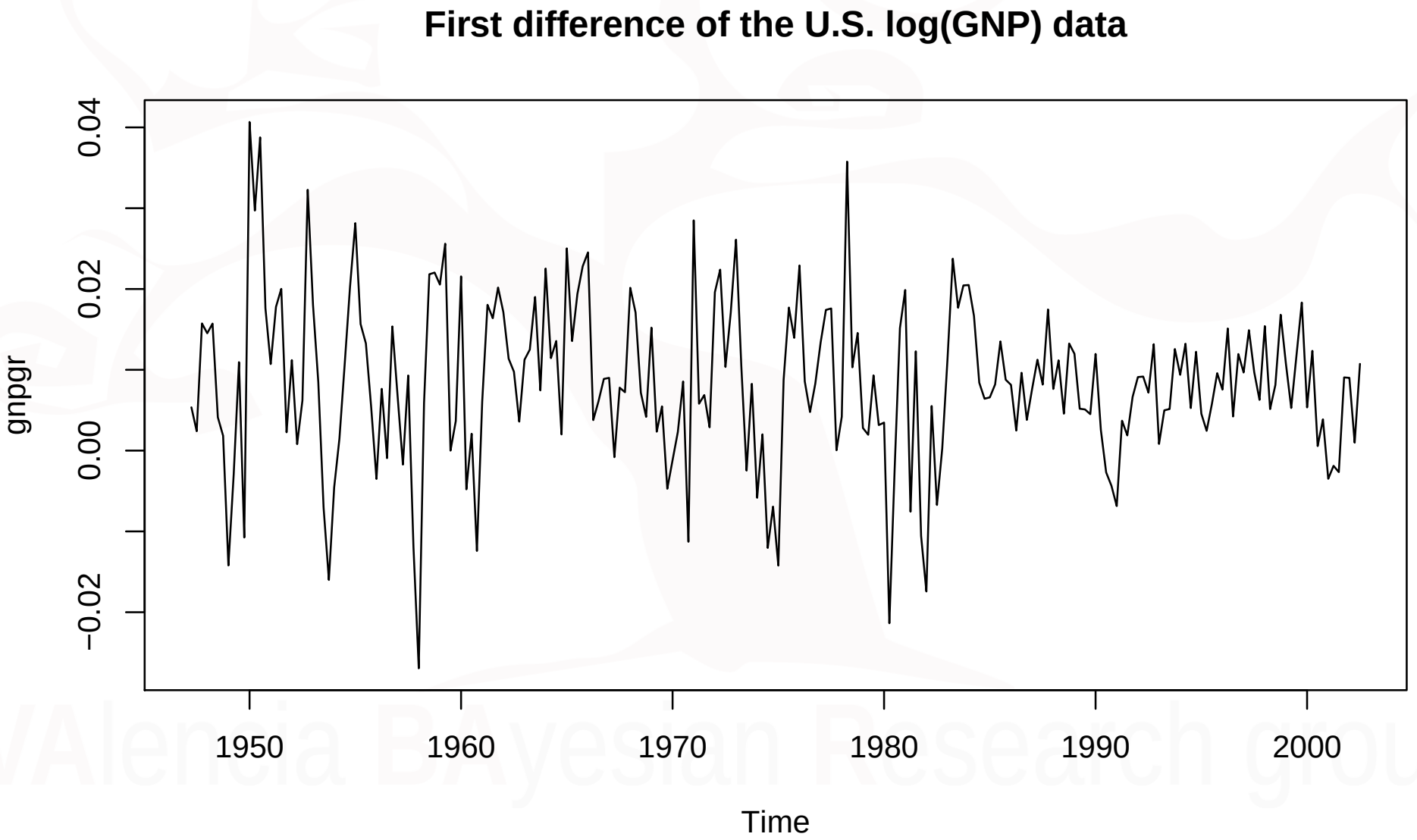


US GNP: test de estacionariedad

```
adf.test(diff(gnp), alternative = "stationary", k = 0)
```

```
    Augmented Dickey-Fuller Test

data:  diff(gnp)
Dickey-Fuller = -10.64, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary
```

log(GNP): Dif. orden 1



US GNP: estacionariedad tras dif. y log

```r
adf.test(gnpgr, alternative = "stationary", k = 0)
```

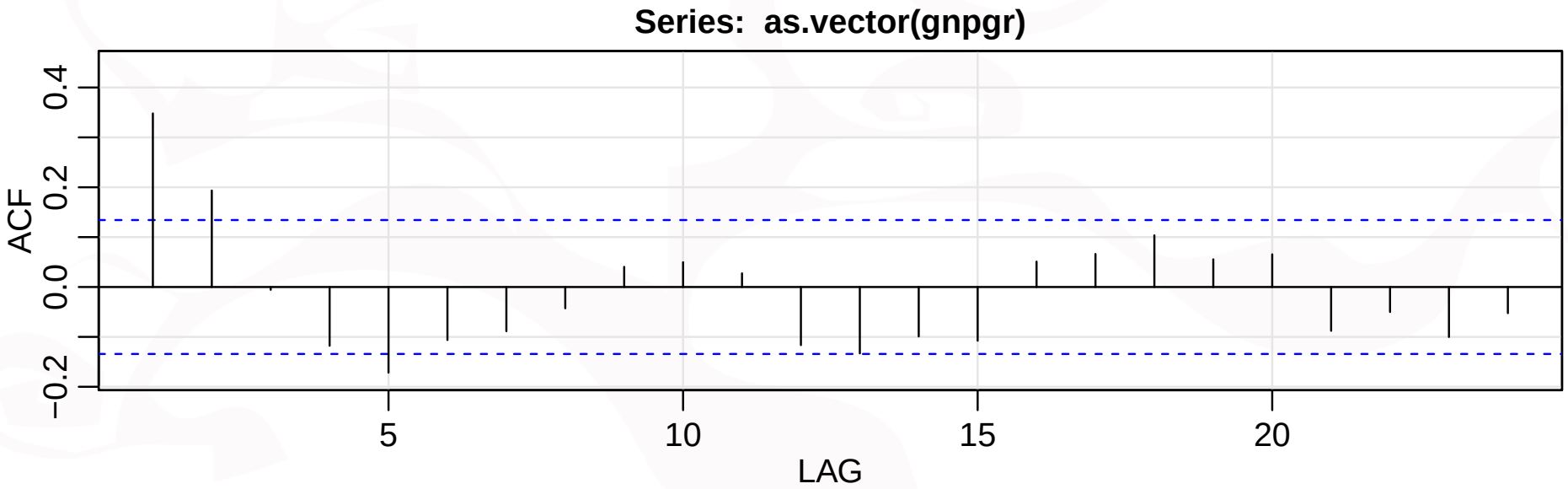
```
    Augmented Dickey-Fuller Test

data:  gnpgr
Dickey-Fuller = -10.341, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary
```

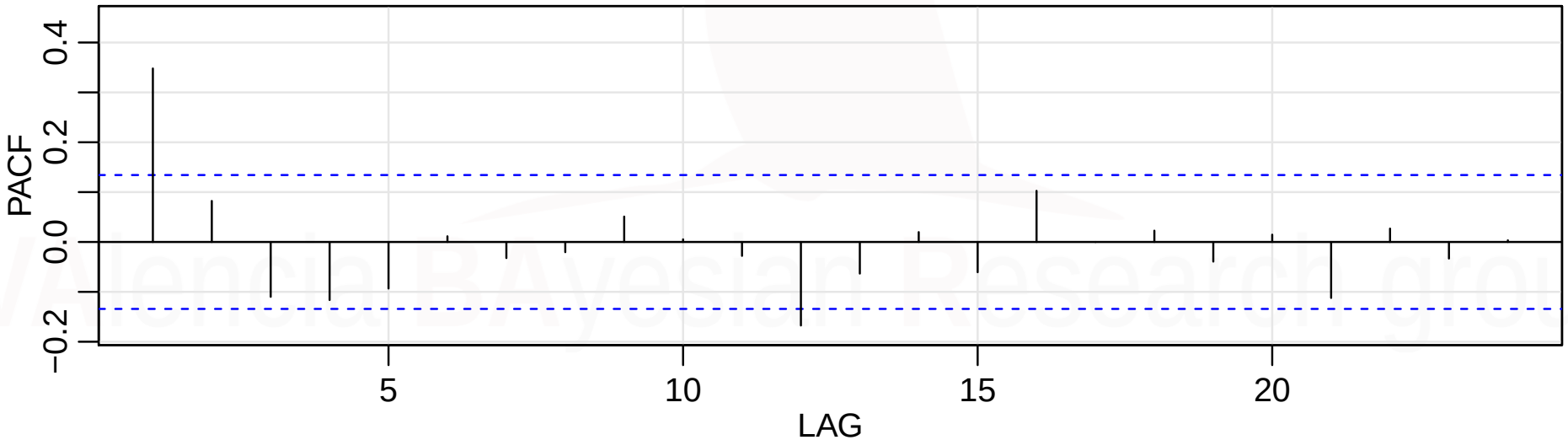
US GNP: Tendencia tras log y $d = 1$

```
# Test de estacionariadad con tendencia
kpss.test(gnpgr, null = "Trend")
```

```
    KPSS Test for Trend Stationarity

data:  gnpgr
KPSS Trend = 0.024163, Truncation lag parameter = 4, p-value = 0.1
```

ACF log(GNP) con d=1





AUTO.ARIMA

Este comando encuentra el "mejor" modelo que se adapta a los datos, sea o no estacional.

```
install.packages("forecast")
library(forecast)
auto.arima(x, d=NA, D=NA, max.p=5, max.q=5,
    max.P=2, max.Q=2, max.order=5, start.p=2,
    start.q=2, start.P=1, start.Q=1,
    stationary=FALSE,
    seasonal=TRUE,ic=c("aicc","aic", "bic"),
    stepwise=TRUE, trace=FALSE,
    approximation=(length(x)>100 | frequency(x)>12),
    xreg=NULL,test=c("kpss","adf","pp"),
    seasonal.test=c("ocsb","ch"),allowdrift=TRUE,
    lambda=NULL, parallel=FALSE, num.cores=NULL)
```

```
# library(forecast) # libro: https://otexts.com/fpp2/
ar.mod = Arima(log(gnp), order = c(1, 1, 0), sea = c(0, 0, 0))
auto.AR.mod = auto.arima(log(gnp), d = 1, D = 0, sea = FALSE)
```

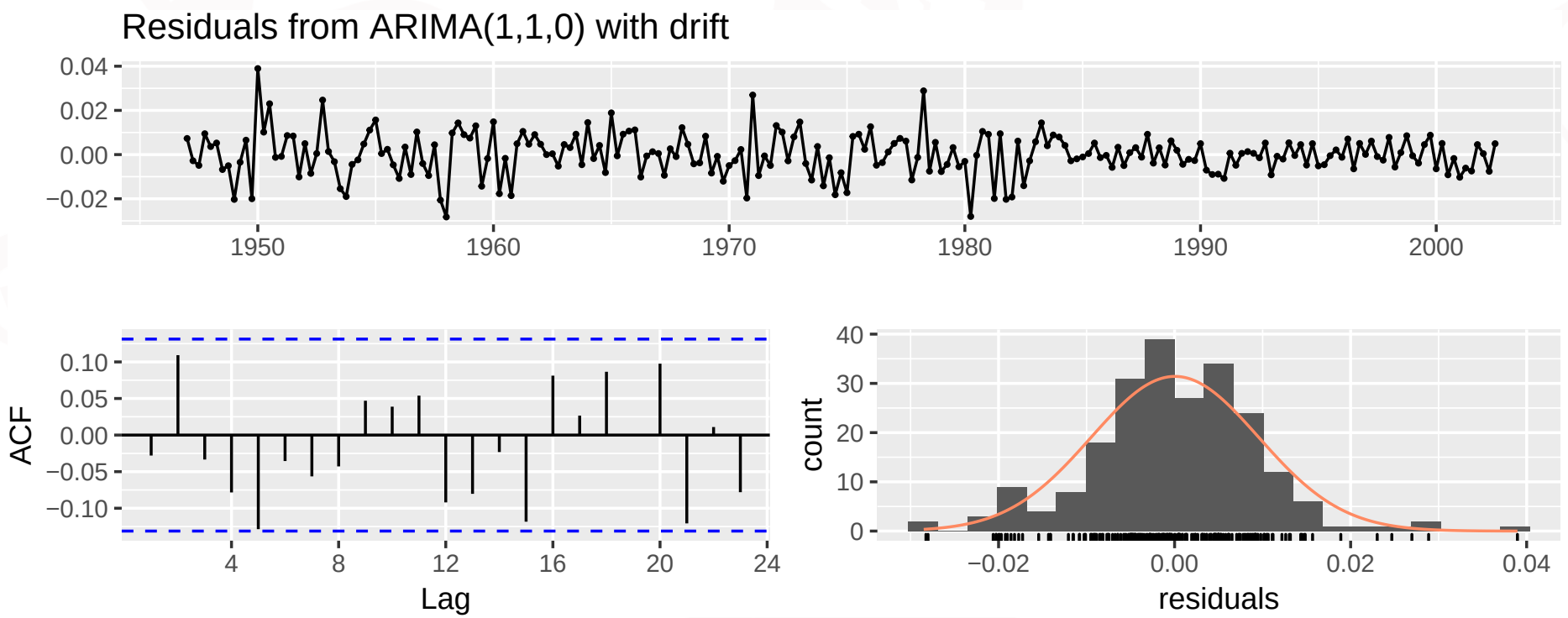
US GNP Series: ARIMA(1,1,0)

```
ar.mod = Arima(log(gnp), order=c(1, 1, 0),
               seasonal = c(0, 0, 0),
               include.drift = TRUE)
```

```
Series: log(gnp)
ARIMA(1,1,0) with drift

Coefficients:
         ar1   drift
      0.3467  0.0083
s.e.  0.0627  0.0010

sigma^2 estimated as 9.136e-05:  log likelihood=718.61
AIC=-1431.22   AICc=-1431.11   BIC=-1421.01
```

US GNP Series: ARIMA(1,1,0)

```
checkresiduals(ar.mod)
```



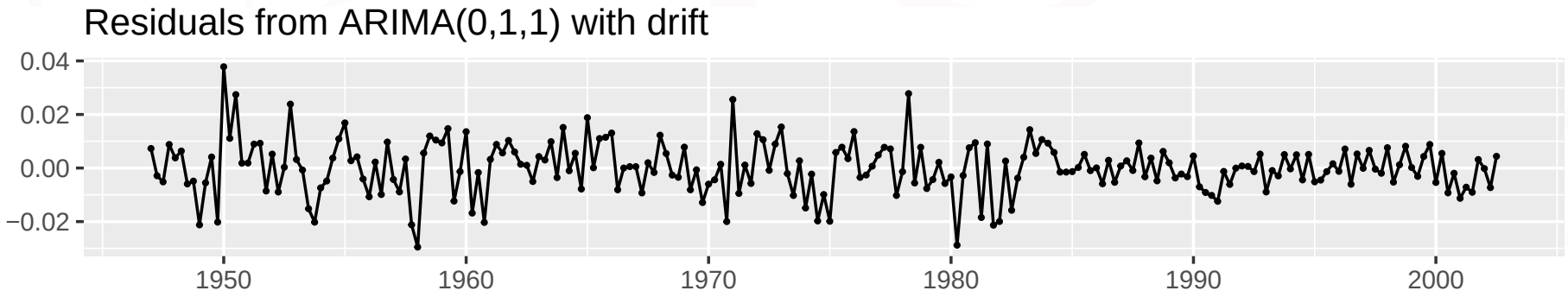

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,0) with drift
## Q* = 9.8183, df = 6, p-value = 0.1325
##
## Model df: 2.   Total lags used: 8
```

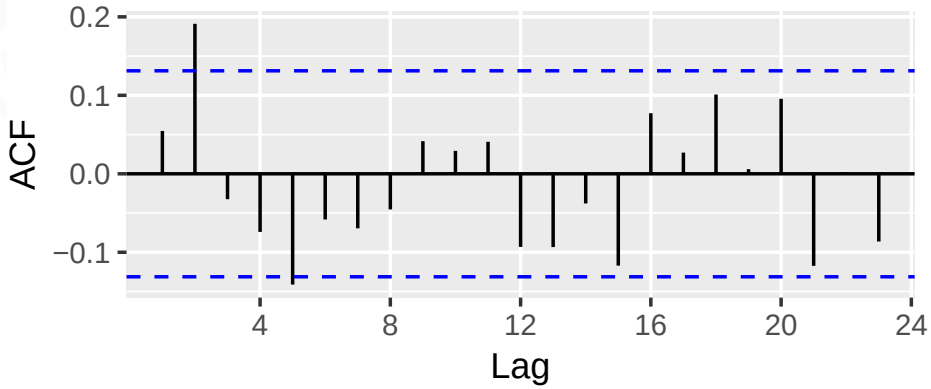
US GNP Series: ARIMA(0,1,1)

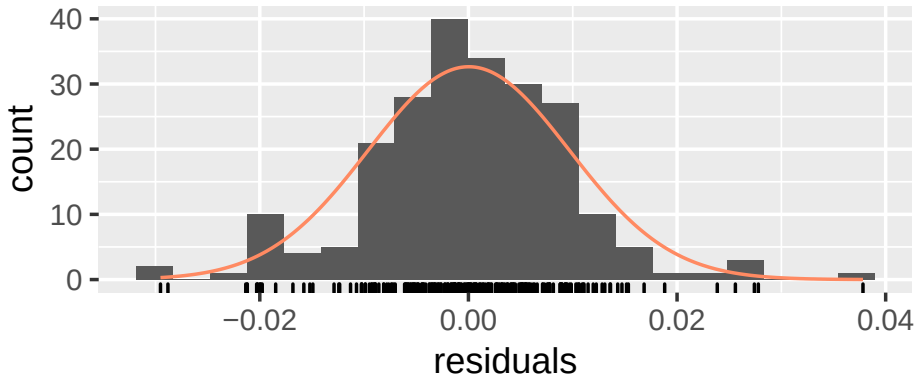
```
ma.mod = Arima(log(gnp), order=c(0, 1, 1),
               seasonal = c(0, 0, 0),
               include.drift = TRUE)
```

```
Series: log(gnp)
ARIMA(0,1,1) with drift

Coefficients:
         ma1   drift
      0.2719  0.0083
s.e.  0.0549  0.0008

sigma^2 estimated as 9.414e-05:  log likelihood=715.3
AIC=-1424.6   AICc=-1424.49   BIC=-1414.39
```

US GNP Series: ARIMA(0,1,1)







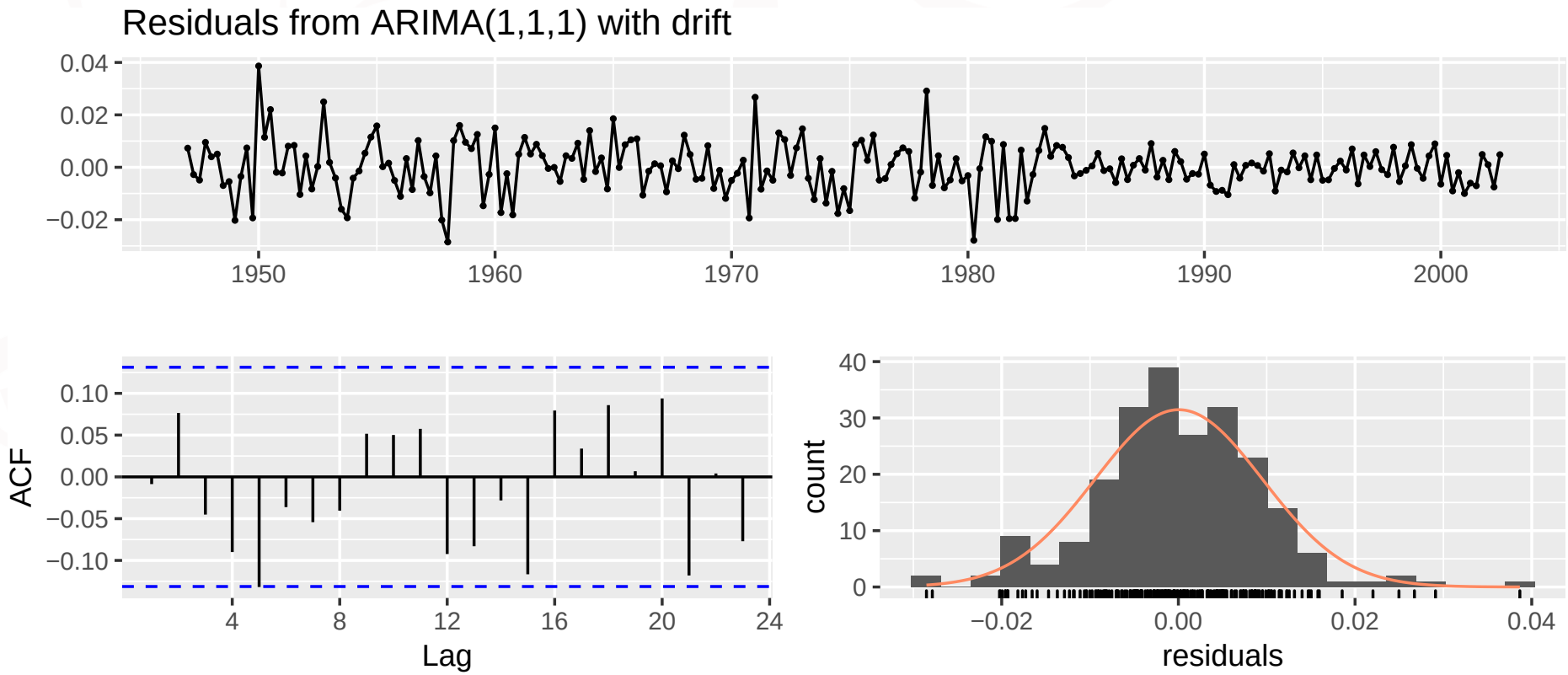

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1) with drift
## Q* = 17.439, df = 6, p-value = 0.007797
##
```

US GNP Series: ARIMA(1,1,1)

```
arma11 = Arima(log(gnp), order=c(1, 1, 1),
               seasonal = c(0, 0, 0),
               include.drift = TRUE)
```

```
Series: log(gnp)
ARIMA(1,1,1) with drift

Coefficients:
         ar1      ma1   drift
      0.4632  -0.1306  0.0083
s.e.  0.1272   0.1350  0.0010

sigma^2 estimated as 9.142e-05:  log likelihood=719.04
AIC=-1430.08   AICc=-1429.9   BIC=-1416.47
```

US GNP Series: ARIMA(0,1,1)




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1) with drift
## Q* = 9.0298, df = 5, p-value = 0.1079
##
```

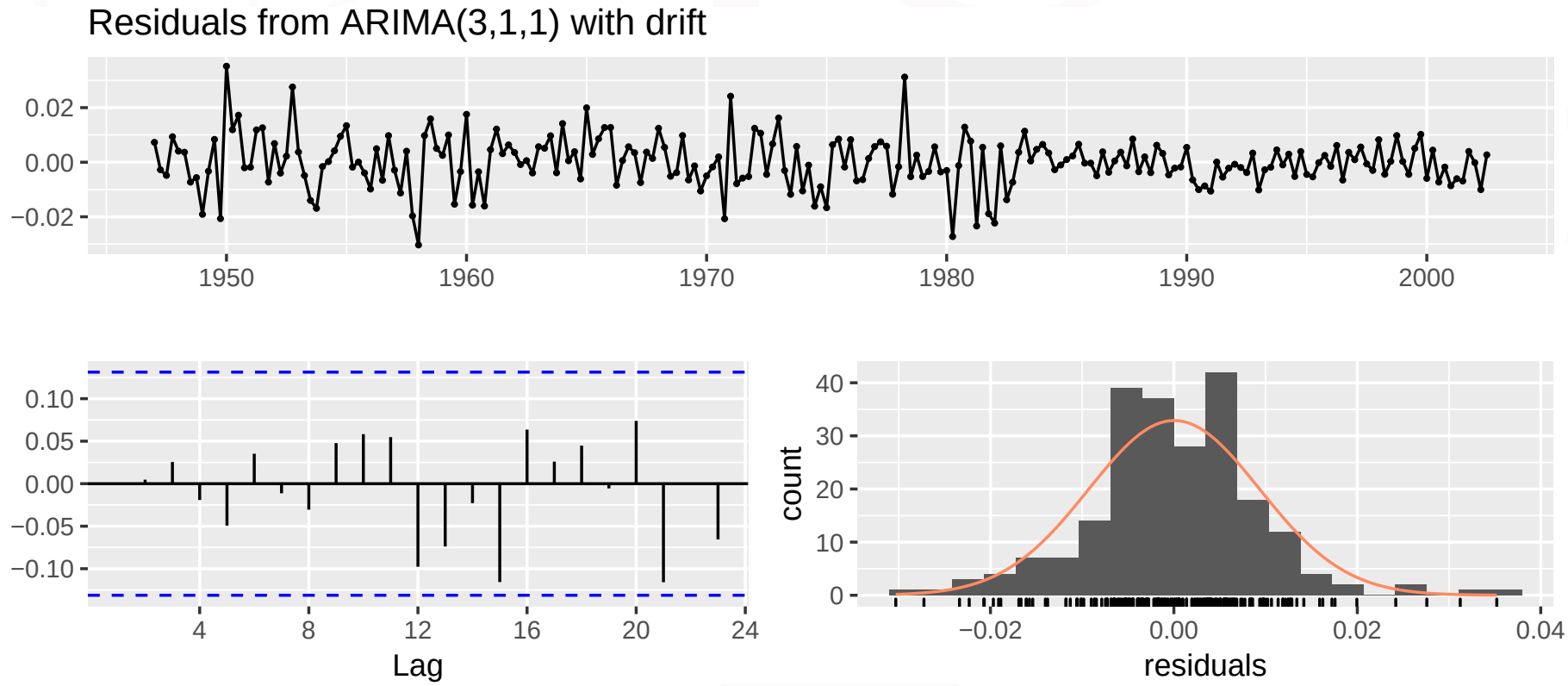
US GNP Series: auto.arima

```
auto.arma11 = auto.arima(log(gnp),d=1,D=0,
                         seasonal=FALSE)
```

```
Series: log(gnp)
ARIMA(3,1,1) with drift

Coefficients:
         ar1      ar2      ar3      ma1   drift
      1.0905  -0.1251  -0.1739  -0.7881  0.0084
s.e.  0.3027   0.1529   0.0777   0.3112  0.0006

sigma^2 estimated as 8.906e-05:  log likelihood=722.89
AIC=-1433.78   AICc=-1433.39   BIC=-1413.36
```

*The parameter μ is called the **"drift"** in the R output when $d = 1$*

US GNP Series: `auto.arima`




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,1,1) with drift
## Q* = 1.3307, df = 3, p-value = 0.7218
##
```

US GNP Series: Selección de modelos

	AIC	AICc	BIC
ARIMA(1,1,0)	-1431.221	-1431.110	-1421.012
ARIMA(0,1,1)	-1424.600	-1424.490	-1414.392
ARIMA(1,1,1)	-1430.080	-1429.896	-1416.469
auto.arima	-1433.781	-1433.390	-1413.365

Predicción

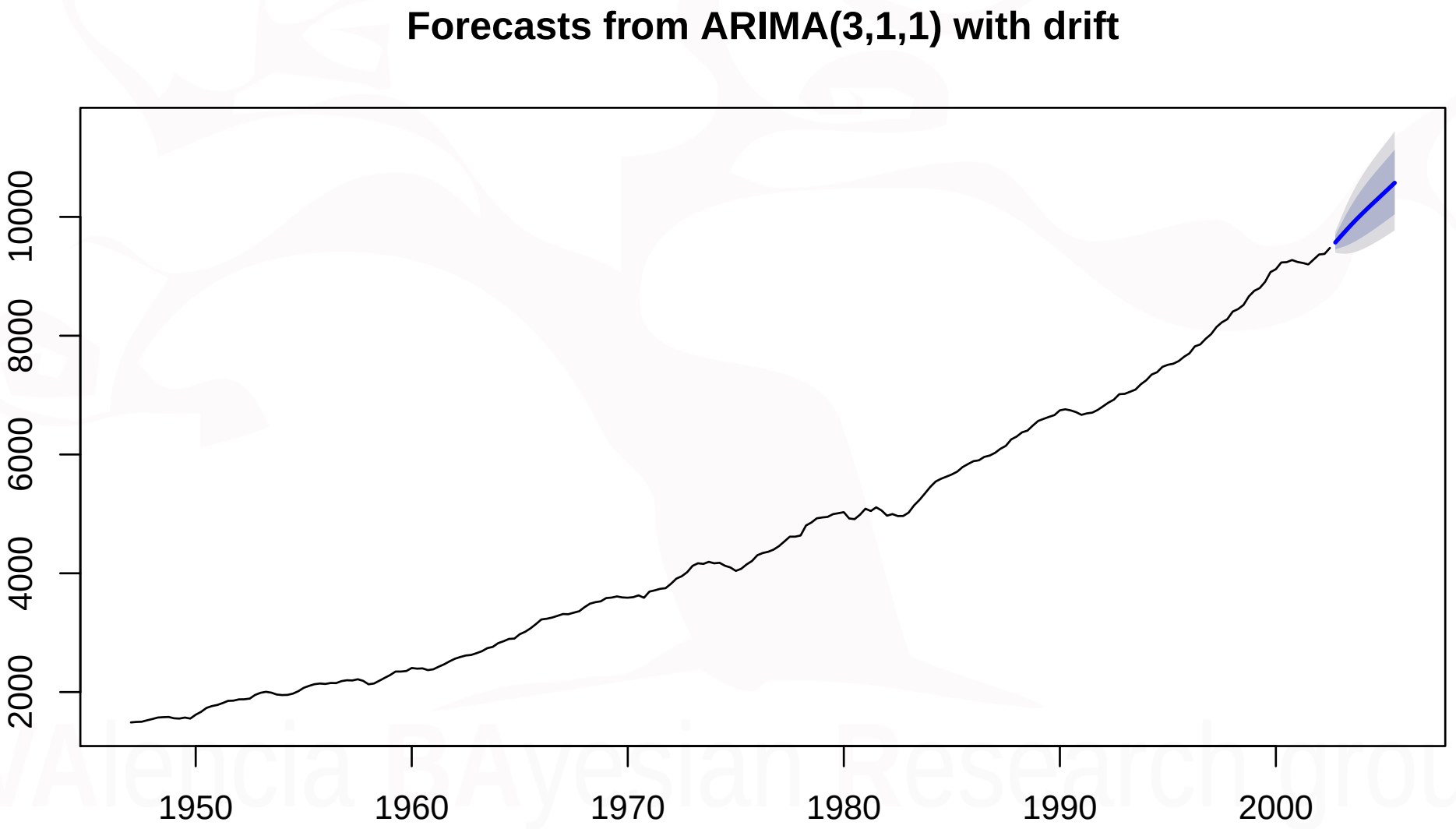
```
autoPred <- forecast(auto.arma11, h = 12)
```

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
2002 Q4	9.166345	9.154251	9.178440	9.147848	9.184842
2003 Q1	9.177074	9.157215	9.196933	9.146703	9.207446
2003 Q2	9.187448	9.160494	9.214403	9.146225	9.228671
2003 Q3	9.197489	9.164874	9.230104	9.147609	9.247369
2003 Q4	9.207020	9.169975	9.244065	9.150364	9.263675
2004 Q1	9.216098	9.175644	9.256551	9.154229	9.277966
2004 Q2	9.224803	9.181695	9.267912	9.158875	9.290732
2004 Q3	9.233249	9.188011	9.278487	9.164063	9.302434
2004 Q4	9.241535	9.194513	9.288557	9.169621	9.313449
2005 Q1	9.249746	9.201156	9.298337	9.175433	9.324059
2005 Q2	9.257940	9.207906	9.307973	9.181420	9.334459
2005 Q3	9.266151	9.214744	9.317558	9.187531	9.344772

Predicción deshaciendo **log(datos)**

```
autoPred$mean = exp(autoPred$mean)
autoPred$lower = exp(autoPred$lower)
autoPred$upper = exp(autoPred$upper)
autoPred$x = exp(autoPred$x)
autoPred$fitted = exp(autoPred$fitted)
autoPred$residuals = exp(autoPred$residuals)
plot(autoPred)
```

Predicción deshaciendo **log(datos)**



Ejemplo serie con estacionalidad

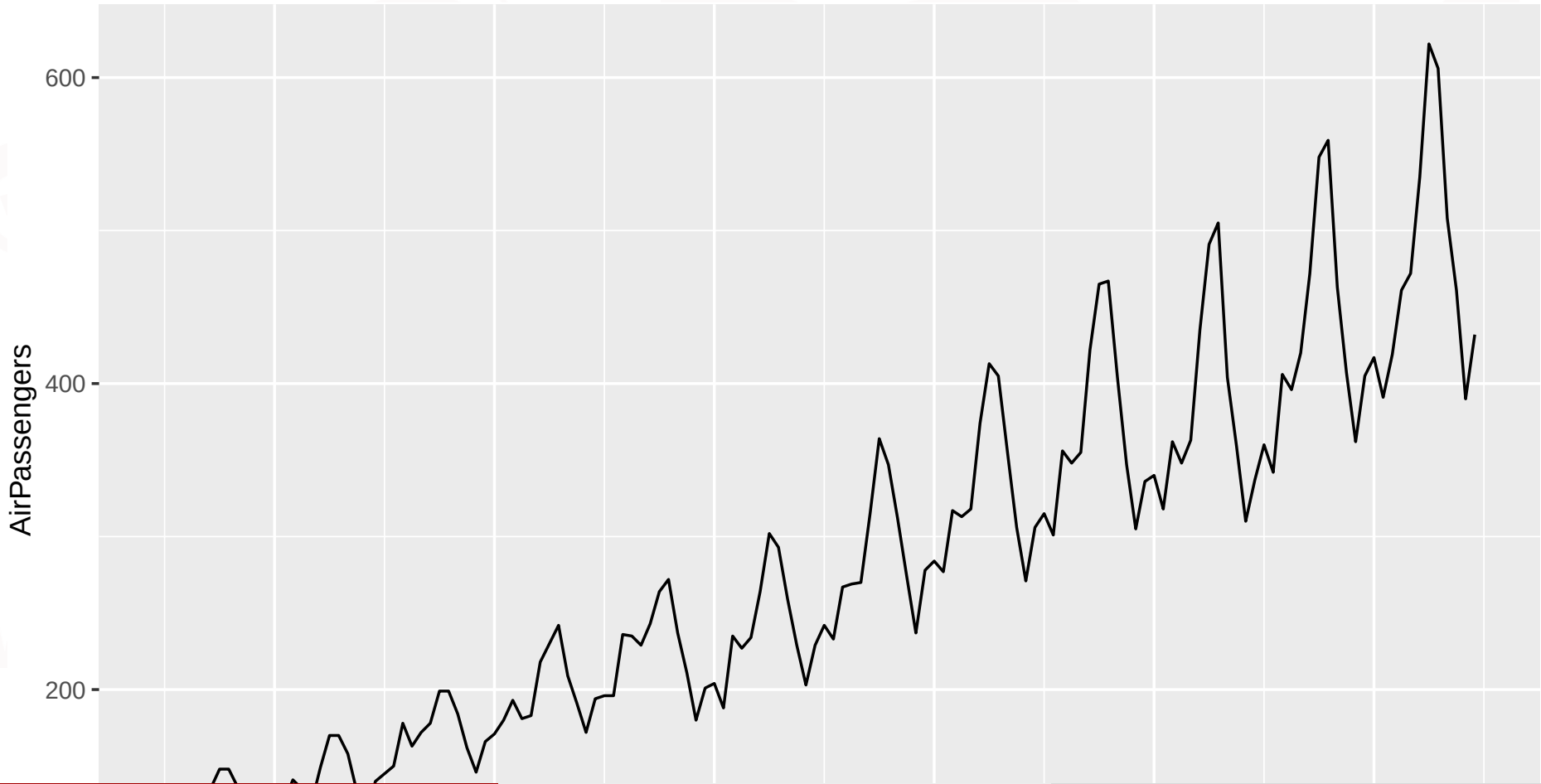
AirPassengers

```
library(tseries)  # for ADF test
library(forecast)  # auto.arima
data(AirPassengers)  # load
class(AirPassengers)
```

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

AirPassengers

```
autoplot(AirPassengers)
```



AirPassengers

```
summary(AirPassengers)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   104.0   180.0   265.5   280.3   360.5   622.0
```

AirPassengers

```
start(AirPassengers)
```

```
## [1] 1949    1
```

```
end(AirPassengers)
```

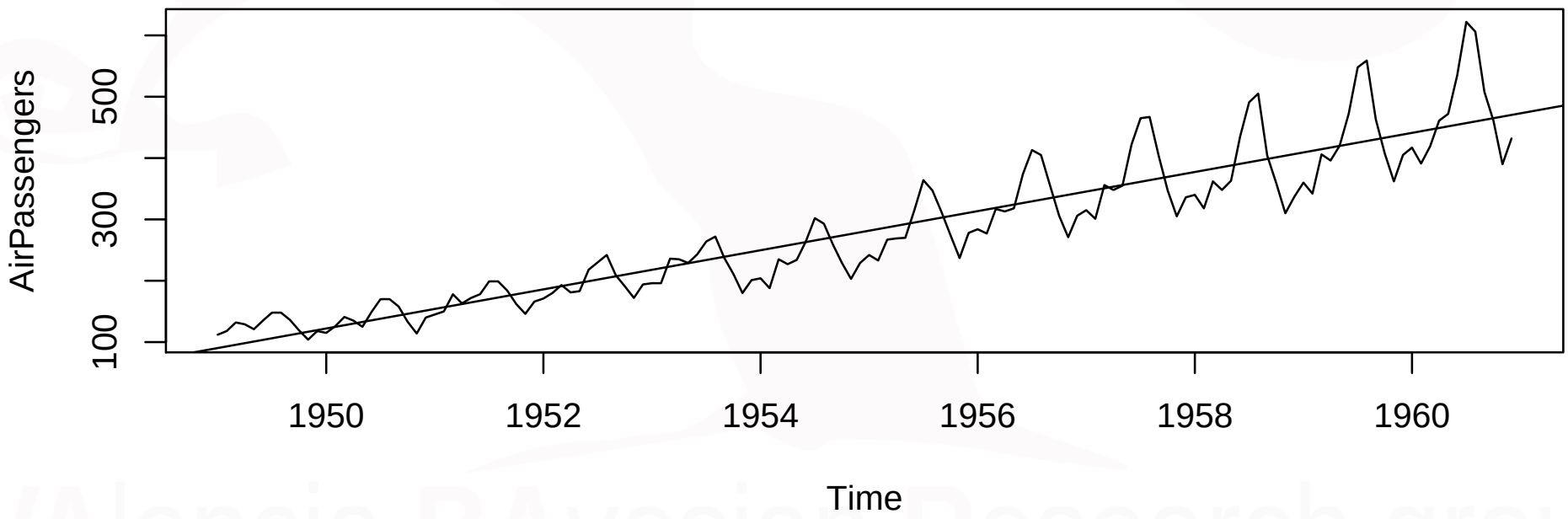
```
## [1] 1960   12
```

```
frequency(AirPassengers)  # The cycle of this time series: 12
```

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

AirPassengers

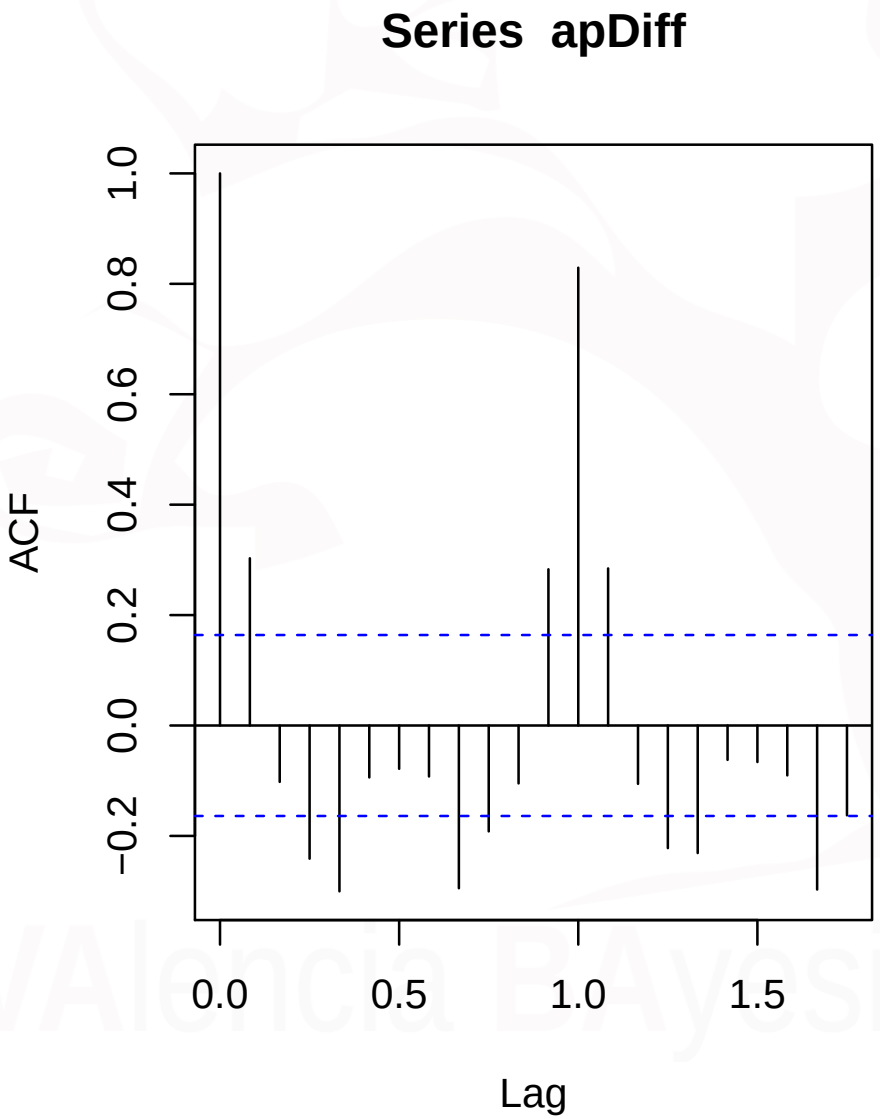
```
plot(AirPassengers)
linearModel = lm(AirPassengers ~ time(AirPassengers))
abline(reg = linearModel)  # Probando un modelo lineal
```

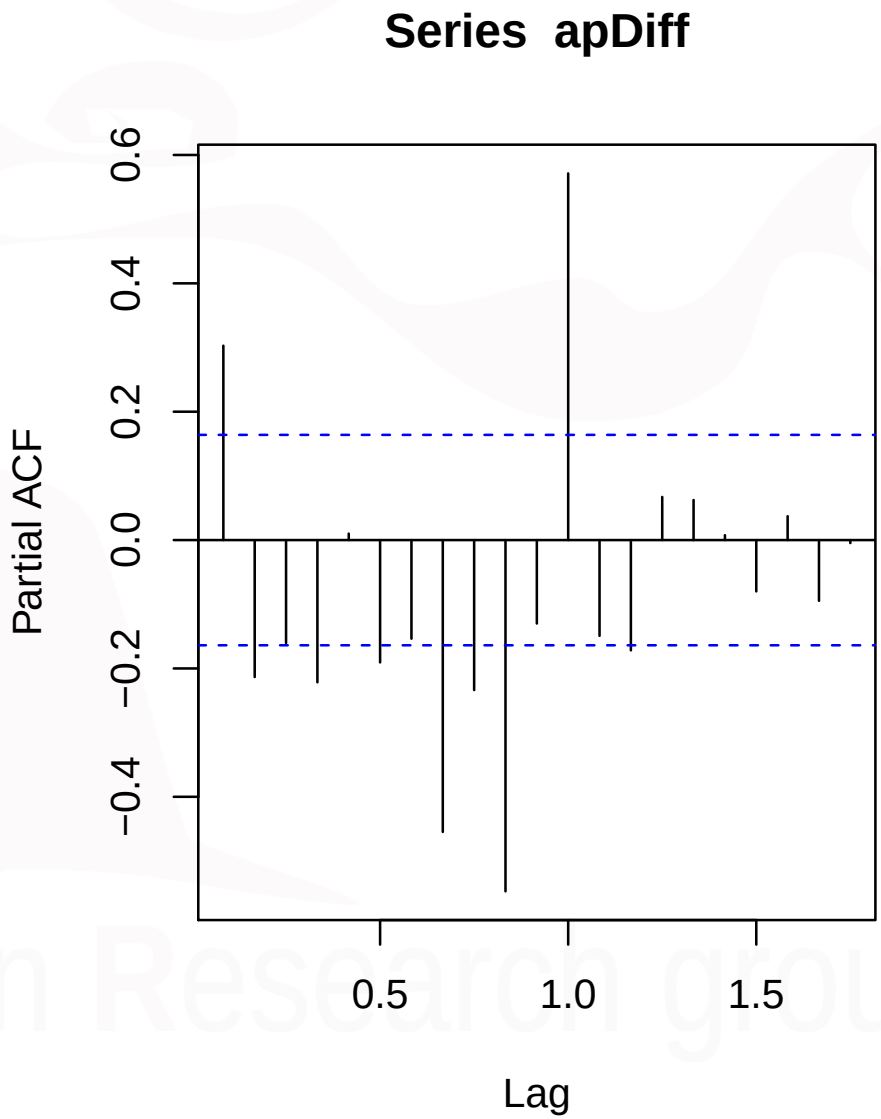


AirPassengers: ACF

```
apNum = as.numeric(AirPassengers)
apDiff = diff(AirPassengers, differences = 1)
op = par(mfrow = c(1, 2))  # start multi plot
acf(apDiff, plot = T)
pacf(apDiff, plot = T)
```

AirPassengers: ACF





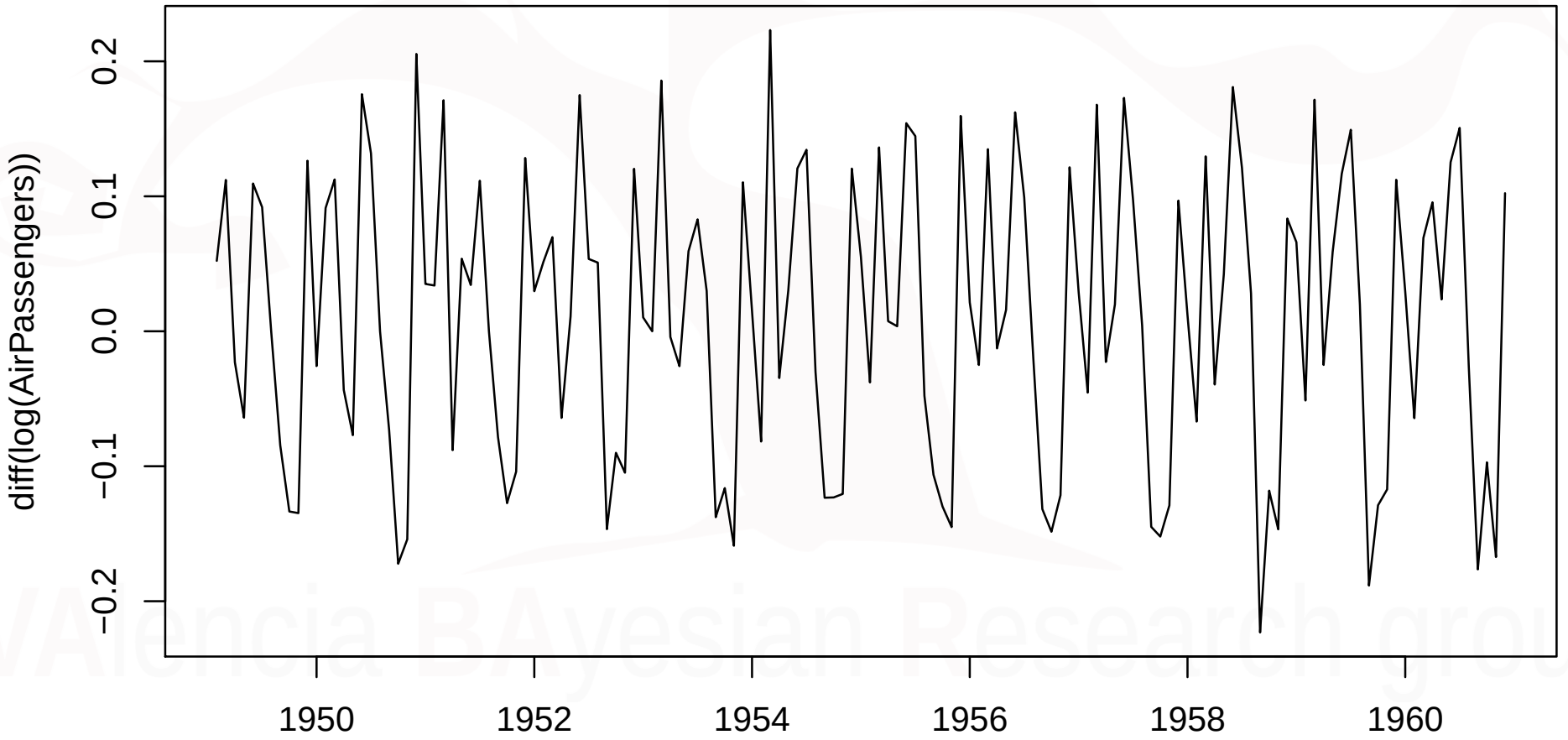
AirPassengers: ¿Es estacionaria?

```
adf.test(diff(log(AirPassengers)), alternative = "statio
    k = 0)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(log(AirPassengers))
## Dickey-Fuller = -9.6003, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

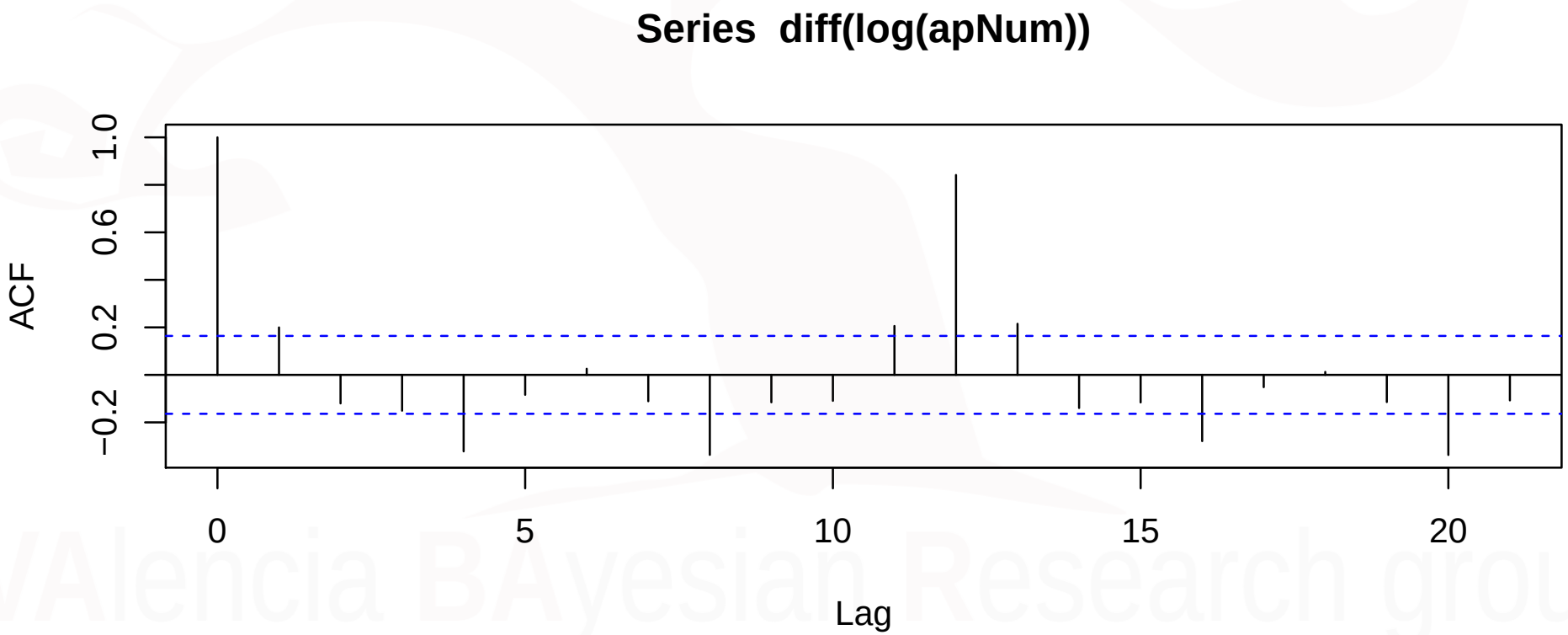
AirPassengers: Diferenciando la serie

```
plot(diff(log(AirPassengers)))
```



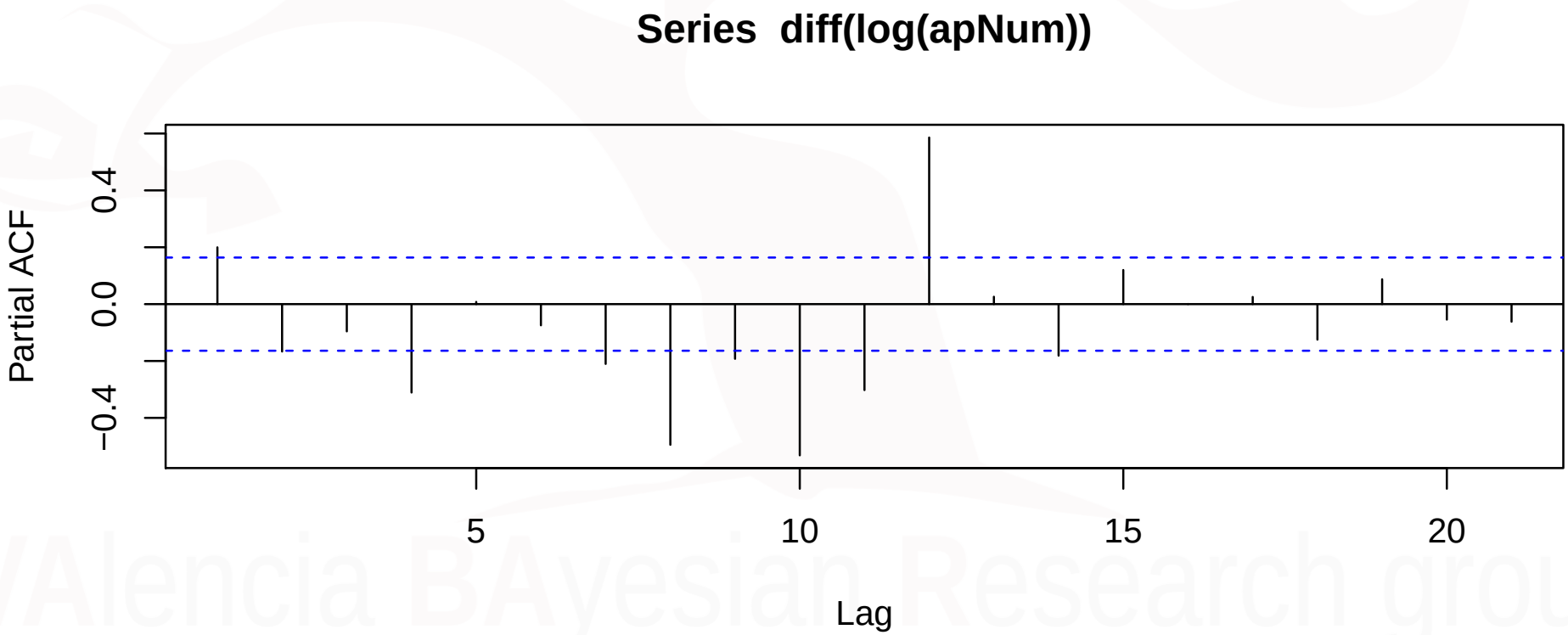
AirPassengers: Diferenciando y logaritmo

```
acf(diff(log(apNum)))
```



AirPassengers: Diferenciando y logaritmo

```
pacf(diff(log(apNum)))
```



AirPassengers: construyendo el modelo (I)

```
autoArimaModel = auto.arima(AirPassengers, d = 1)
autoArimaModel
```

```
## Series: AirPassengers
## ARIMA(2,1,1)(0,1,0)[12]
##
## Coefficients:
##          ar1     ar2      ma1
##       0.5960  0.2143  -0.9819
## s.e.  0.0888  0.0880   0.0292
##
## sigma^2 estimated as 132.3:  log likelihood=-504.92
## AIC=1017.85   AICc=1018.17   BIC=1029.35
```

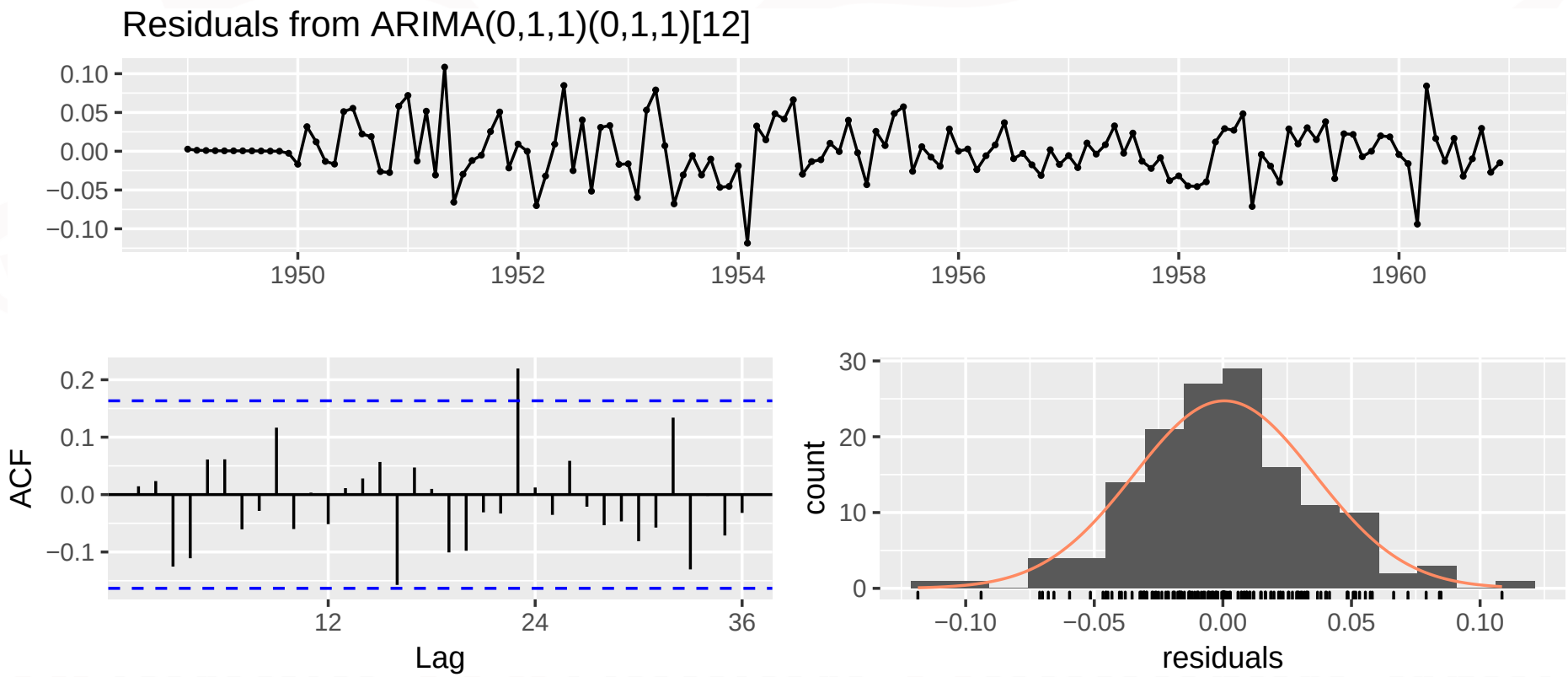
AirPassengers: construyendo el modelo (II)

```
autoArimaModelLog = auto.arima(log(AirPassengers), d = 1)
autoArimaModelLog
```

```
## Series: log(AirPassengers)
## ARIMA(0,1,1)(0,1,1)[12]
##
## Coefficients:
##           ma1     sma1
##       -0.4018  -0.5569
## s.e.   0.0896   0.0731
##
## sigma^2 estimated as 0.001371:  log likelihood=244.7
## AIC=-483.4   AICc=-483.21   BIC=-474.77
```

AirPassengers: checkresiduals

```
checkresiduals(autoArimaModelLog)
```




```
##
##  Ljung-Box test
##
```

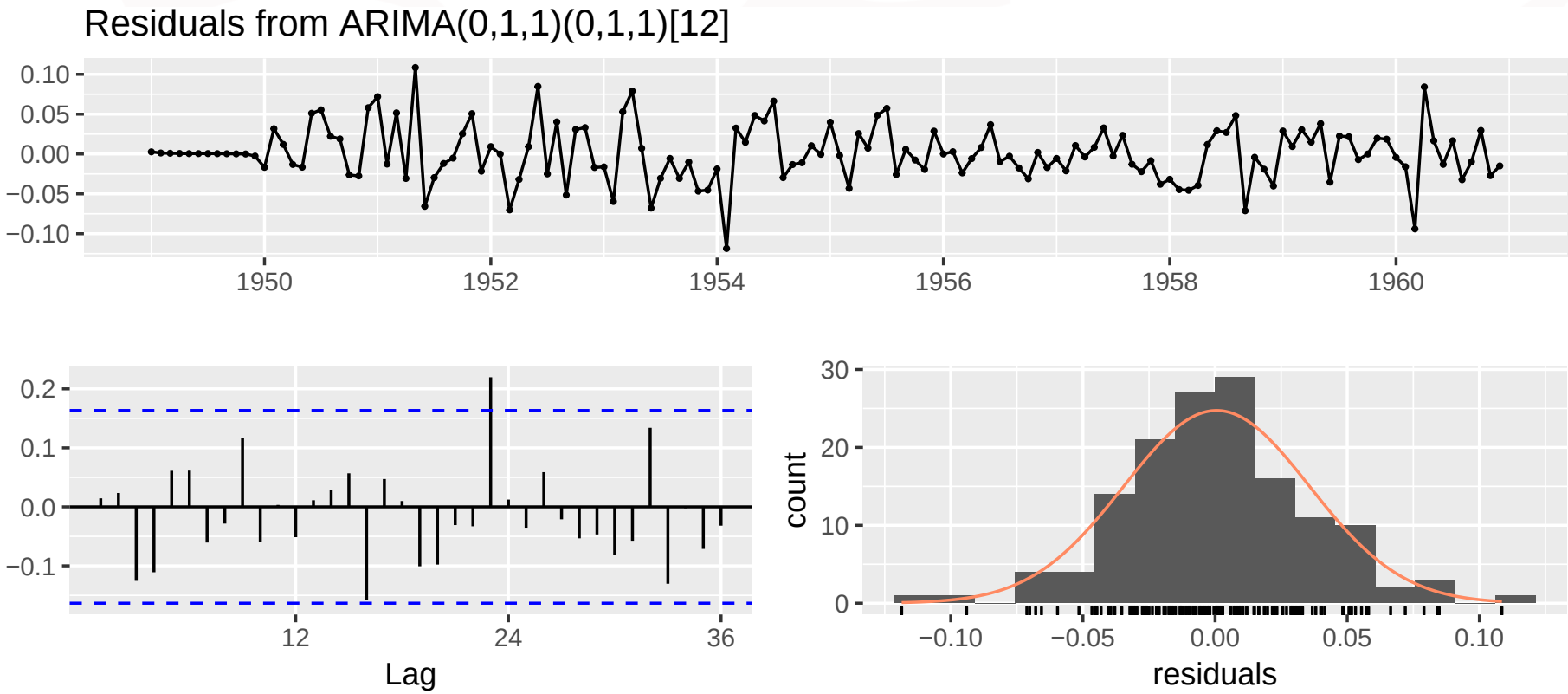
AirPassengers: construyendo el modelo (III)

```
#Manualmente
pdqParam = c(0, 1, 1)
manualFit <- arima(log(AirPassengers), pdqParam,
                   seasonal = list(order = pdqParam,
                                   period = 12))
manualFit
```

```
##
## Call:
## arima(x = log(AirPassengers), order = pdqParam, seasonal = list(
##     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 = -
```

AirPassengers: checkresiduals

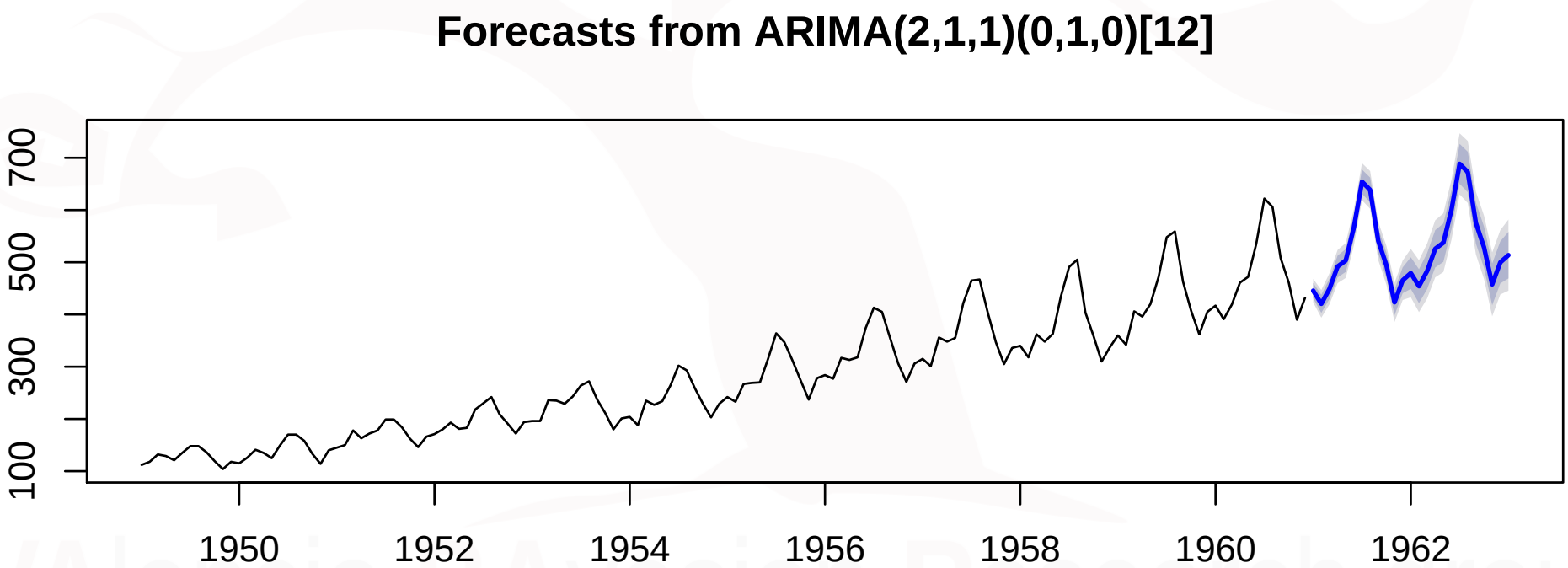
```
checkresiduals(manualFit)
```




```
##
##  Ljung-Box test
##
```

AirPassengers: Prediciendo

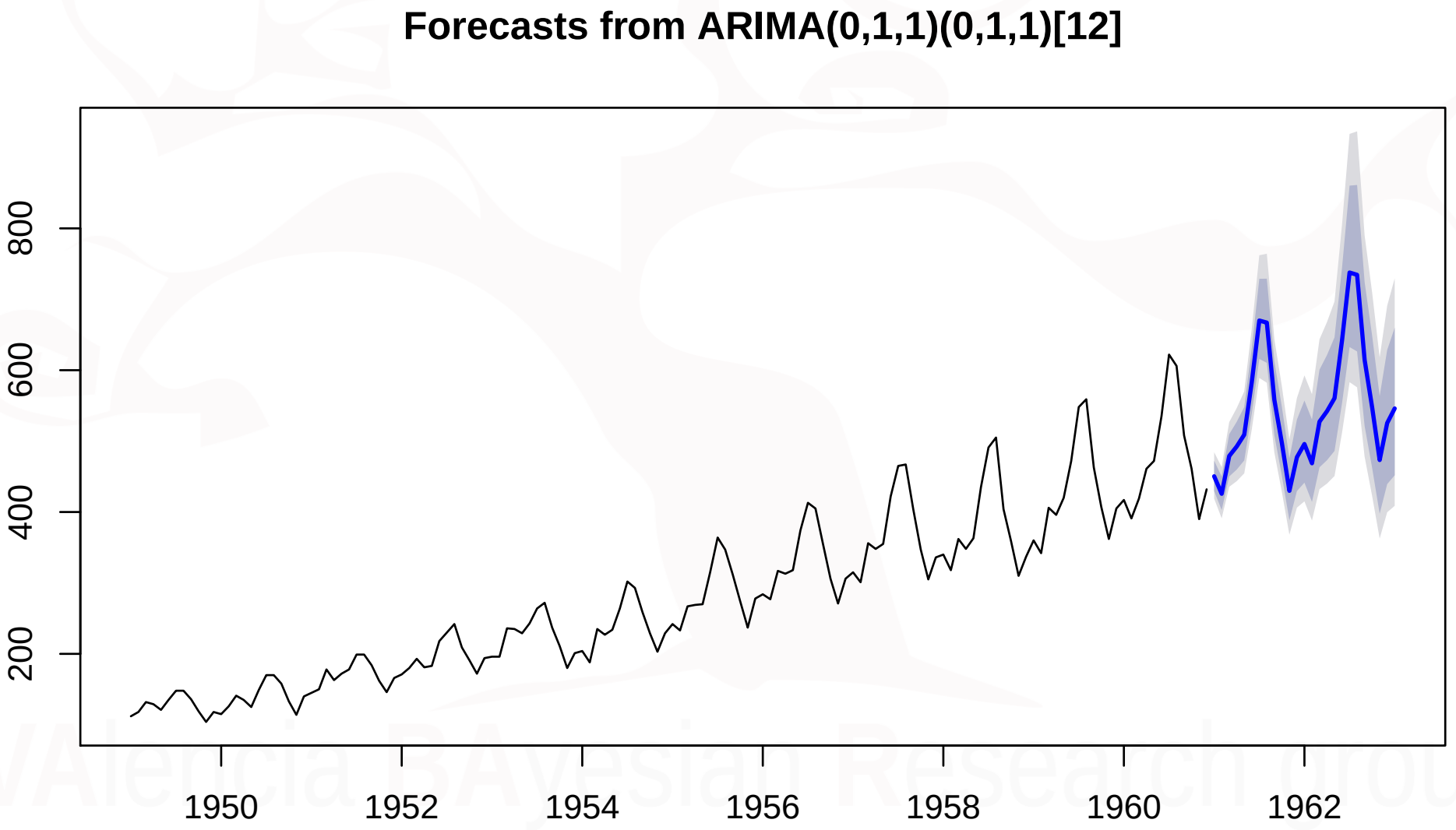
```
autoPred = forecast(autoArimaModel, h = 25)
plot(autoPred)
```



AirPassengers: Deshaciendo **log**

```
autoPred = forecast(autoArimaModelLog, h = 25)
autoPred$mean = exp(autoPred$mean)
autoPred$lower = exp(autoPred$lower)
autoPred$upper = exp(autoPred$upper)
autoPred$x = exp(autoPred$x)
autoPred$fitted = exp(autoPred$fitted)
autoPred$residuals = exp(autoPred$residuals)
plot(autoPred)
```

AirPassengers: Deshaciendo **log**



AirPassengers: Predecir manualmente

```
manualPred <- predict(manualFit, n.ahead = 25)
ts.plot(AirPassengers, exp(manualPred$pred), log = "y", lty = c(1,
    3))
```