Procesos Estocásticos y Series Temporales

Práctica 4: Modelo ARIMA y SARIMA

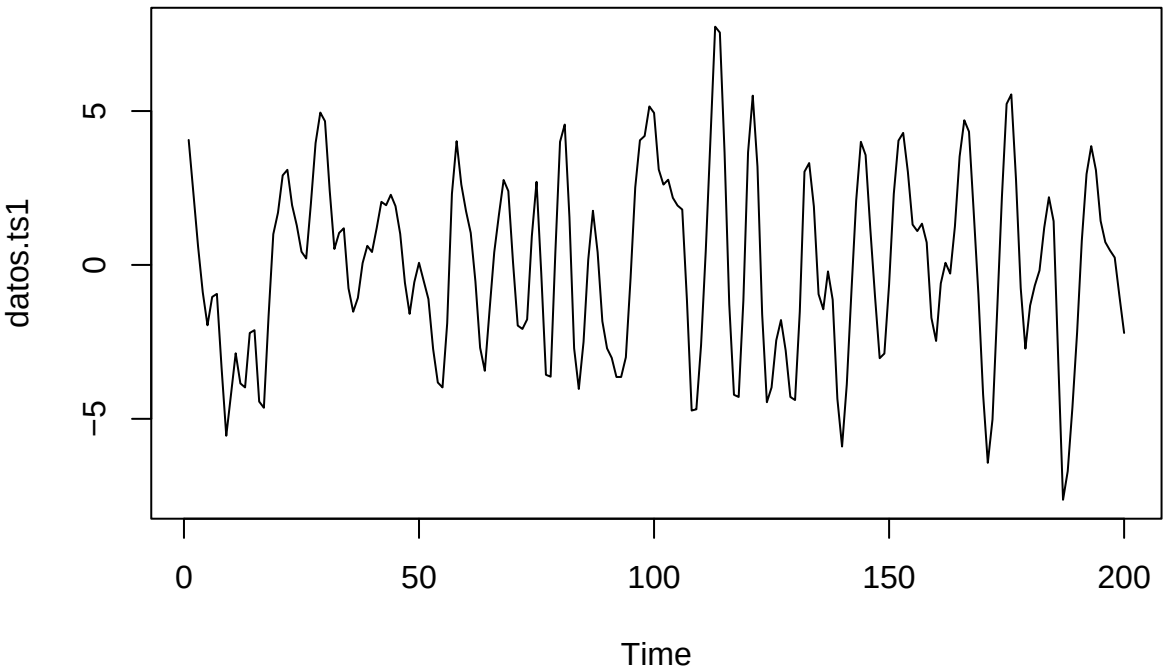
Francisco Javier Mercader Martínez

Problema 1

Usando los datos del fichero **ARIMA_P1.txt**, resuelve las siguientes cuestiones:

1) Representa la serie de datos en un gráfico temporal. ¿La serie presenta estacionalidad? ¿Crees que la serie proviene de un proceso estocástico estacionario? Justifica tu respuesta.

```
library(forecast)
library(tseries)
datos1 <- read.csv2("ARIMA_P1.txt", header = FALSE)
datos.ts1 <- ts(datos1$V1, frequency = 1)
ts.plot(datos.ts1)
```



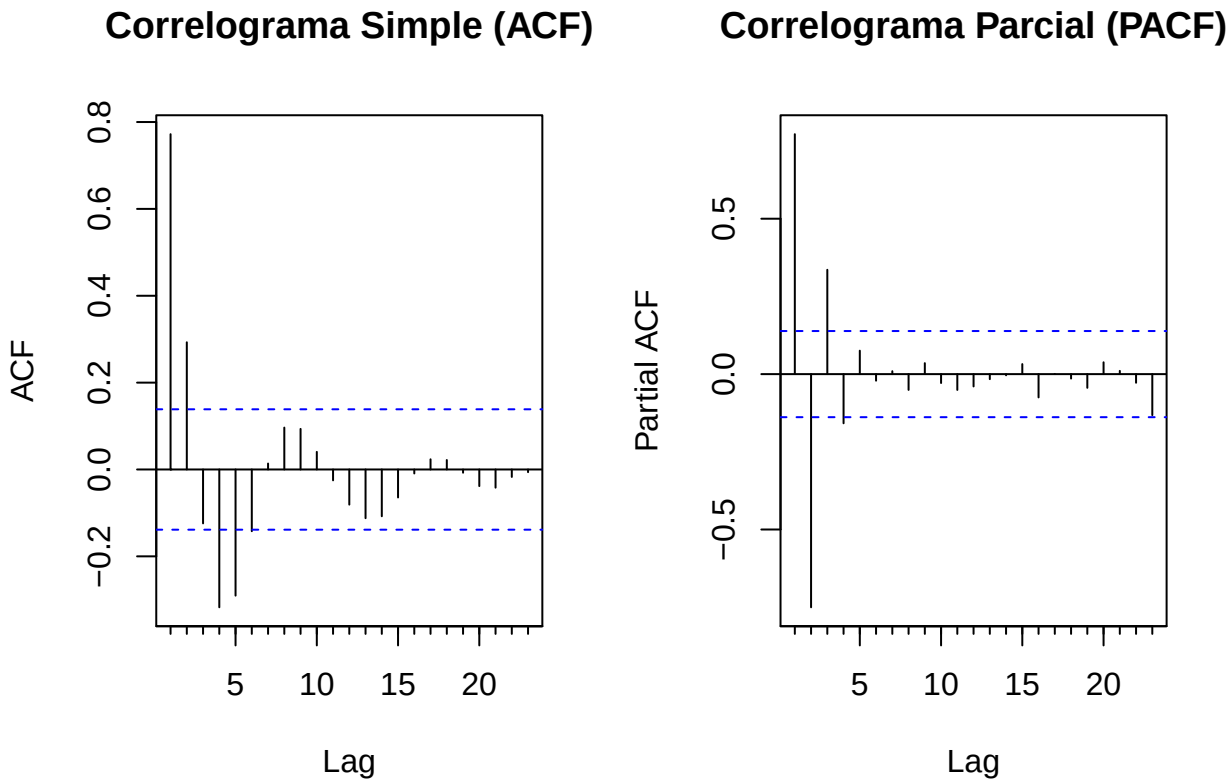
- La serie **no** presenta estacionalidad (no hay patrón periódico).
- La serie oscila alrededor de la media.
- La varianza parece constante

2) En el caso de que la serie no sea estacionaria, realiza las transformaciones que estimes oportunas para convertirla en estacionaria, comentando por qué las realizas.

Dado que la serie es estacionaria, **no** son necesarias transformaciones. Trabajaremos directamente con la serie original

3) Obtén el correlograma simple y parcial de la serie estacionaria. En función de los resultados, ¿qué modelo(s) ARIMA propondrías como generador de la serie en estudio? Justifica tu respuesta.

```
par(mfrow = c(1, 2))
Acf(datos.ts1, main = "Correlograma Simple (ACF)")
Pacf(datos.ts1, main = "Correlograma Parcial (PACF)")
```



Propondría un $AR(3)$ o un $ARMA(3,5)$.

```
arima.model300 <- Arima(datos.ts1, order = c(3, 0, 0))
arima.model305 <- Arima(datos.ts1, order = c(3, 0, 5))
```

4) Para cada uno de los modelos teóricos propuestos en el apartado anterior, reponde a las siguientes cuestiones:

```
pvalores <- function(modelo) {
  # Extract coefficients
  coefs <- coef(modelo)
  # Extract standard errores
  se <- sqrt(diag(vcov(modelo)))
  # Calculate the t-values
  t_values <- coefs / se
  # Calculate the p-values
  p_values <- 2 * (1 - pnorm(abs(t_values)))
  # Create a data frame with coefficients, standard errors, t-values, and p-values
  results_pvalues <- data.frame(
    Coefficients = coefs,
    Std_Errors = se,
    T_values = t_values,
    P_values = p_values
  )
  print(results_pvalues)
}
```

a) Determina si el modelo es reducible, es decir, si existen coeficientes del modelo no significativos. En caso afirmativo, simplifica el modelo.

```
pvalores(arima.model300)
```

```
##           Coefficients Std_Errors    T_values     P_values
## ar1        1.714592536 0.06365710  26.9348194 0.000000e+00
## ar2       -1.367976114 0.09632859 -14.2011429 0.000000e+00
## ar3        0.430072035 0.06349287   6.7735480 1.256617e-11
## intercept  0.005513975 0.33322111   0.0165475 9.867976e-01
```

Irreducible

```
arima.model304 <- Arima(datos.ts1, order = c(3, 0, 4))
```

```r
pvalores(arima.model304)
```

```
##           Coefficients Std_Errors    T_values     P_values
## ar1        1.848432466 0.23714162  7.79463541 6.439294e-15
## ar2       -1.349223630 0.37059751 -3.64067107 2.719284e-04
## ar3        0.398462911 0.16242486  2.45321387 1.415861e-02
## ma1        0.064472666 0.24568799  0.26241684 7.930001e-01
## ma2       -0.542574925 0.07199458 -7.53633021 4.840572e-14
## ma3       -0.157293476 0.16151848 -0.97384195 3.301350e-01
## ma4       -0.023876179 0.12818115 -0.18626904 8.522338e-01
## intercept  0.008192219 0.23402202  0.03500619 9.720748e-01
```

```r
arima.model303 <- Arima(datos.ts1, order = c(3, 0, 3))
pvalores(arima.model303)
```

```
##           Coefficients Std_Errors     T_values  P_values
## ar1        1.284311332  1.7270260  0.743654871 0.4570853
## ar2       -0.736307341  1.9387434 -0.379785866 0.7041044
## ar3        0.074769639  1.0100632  0.074024712 0.9409907
## ma1        0.631690101  1.7250515  0.366186219 0.7142261
## ma2       -0.071054422  1.3783538 -0.051550207 0.9588871
## ma3       -0.059274493  0.2050931 -0.289012556 0.7725718
## intercept  0.002098279  0.2781542  0.007543583 0.9939811
```

```r
arima.model302 <- Arima(datos.ts1, order = c(3, 0, 2))
pvalores(arima.model302)
```

```
##           Coefficients Std_Errors     T_values  P_values
## ar1        0.886493733  0.7239150  1.224582688 0.2207325
## ar2       -0.304898290  0.8675540 -0.351445883 0.7252539
## ar3       -0.146333513  0.4664476 -0.313719106 0.7537344
## ma1        1.030677309  0.7157740  1.439947988 0.1498821
## ma2        0.265607782  0.5002828  0.530915269 0.5954775
## intercept  0.002504957  0.2840853  0.008817622 0.9929646
```

```r
arima.model202 <- Arima(datos.ts1, order = c(2, 0, 2))
pvalores(arima.model202)
```

```
##           Coefficients Std_Errors     T_values     P_values
## ar1        1.117787174 0.10258645 10.896050519 0.000000e+00
## ar2       -0.580154067 0.08389409 -6.915314728 4.668266e-12
## ma1        0.800210547 0.12456373  6.424105295 1.326474e-10
## ma2        0.102877348 0.12003355  0.857071604 3.914053e-01
## intercept  0.002825648 0.28757365  0.009825824 9.921603e-01
```

```r
arima.model201 <- Arima(datos.ts1, order = c(2, 0, 1))
pvalores(arima.model201)
```

```
##           Coefficients Std_Errors      T_values  P_values
## ar1        1.184512005 0.05933564  19.962908999 0.0000000
## ar2       -0.624314433 0.05879205 -10.619028851 0.0000000
## ma1        0.704949382 0.05239524  13.454454962 0.0000000
## intercept  0.001096124 0.27157071   0.004036239 0.9967796
```

b) Determina valores indicativos de la bondad del ajuste de los modelos que resultan del apartado anterior.

```r
arima.model201
```

```
## Series: datos.ts1
## ARIMA(2,0,1) with non-zero mean
##
## Coefficients:
##          ar1      ar2     ma1    mean
##       1.1845  -0.6243  0.7049  0.0011
## s.e.  0.0593   0.0588  0.0524  0.2716
```

```
##
## sigma^2 = 1.004:  log likelihood = -284.16
## AIC=578.32   AICc=578.63   BIC=594.82
```

```
paste("AIC =", arima.model201$aic)
```

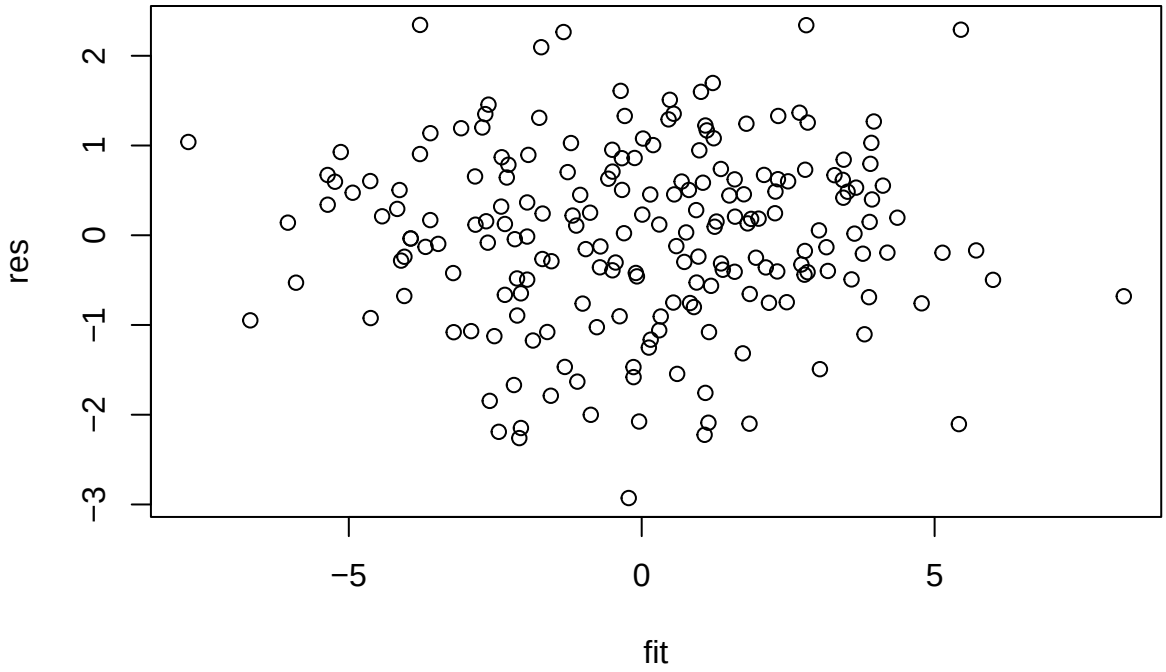
```
## [1] "AIC = 578.324782836619"
```

c) Analiza la validez de los modelos resultantes, es decir, comprueba que se verifican las hipótesis sobre los residuos.

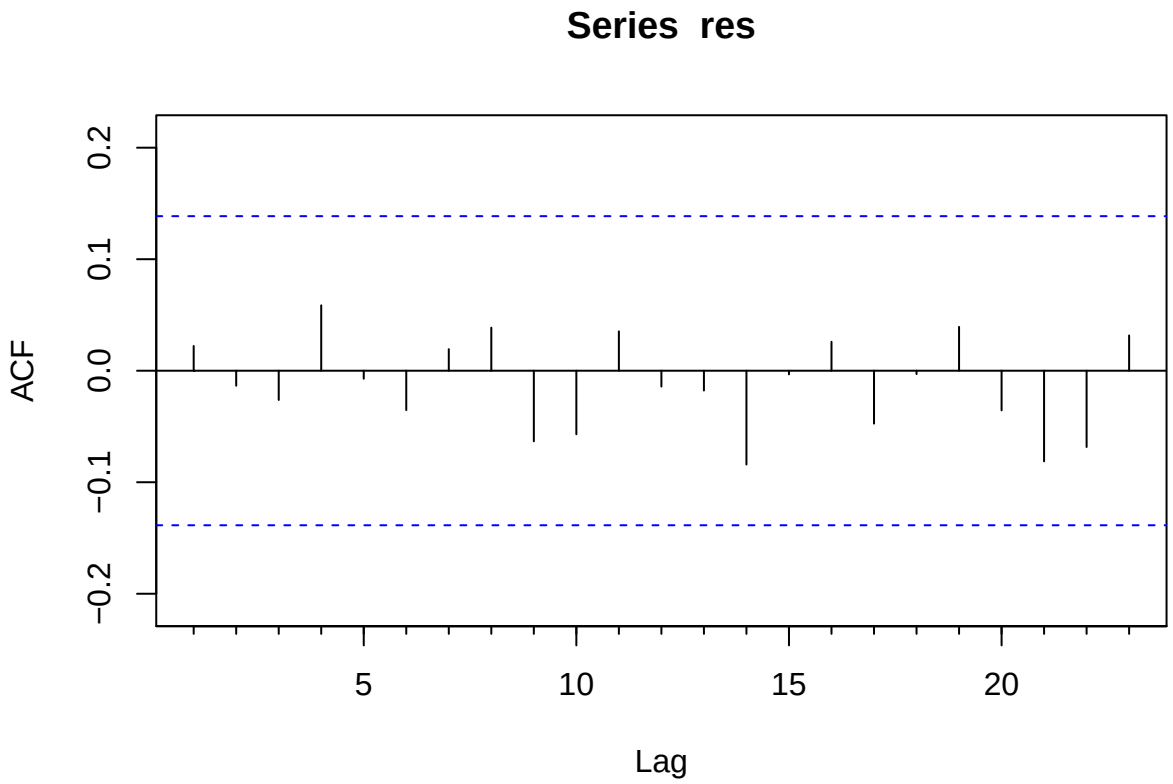
```
res <- arima.model201$residuals
fit <- arima.model201$fitted

# Normlidad de los residuos: Si p-values > 0.05 -> se acepta normalidad
shapiro.test(res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.98892, p-value = 0.1236
```

```
# Homecedasticidad: Si no hay ningún patrón evidente no diferencias de dispersión se acepta la
 ↪  homecedasticidad
plot(fit, res)
```




```
# Independencia: Si residuos normales y correlograma con valores no significativos, se acpeta
 ↪  independencia
Acf(res)
```



d) Tras el análisis realizado, indica cuál sería la expresión del modelo ARIMA estimado que propones como generador de la serie en estudio.

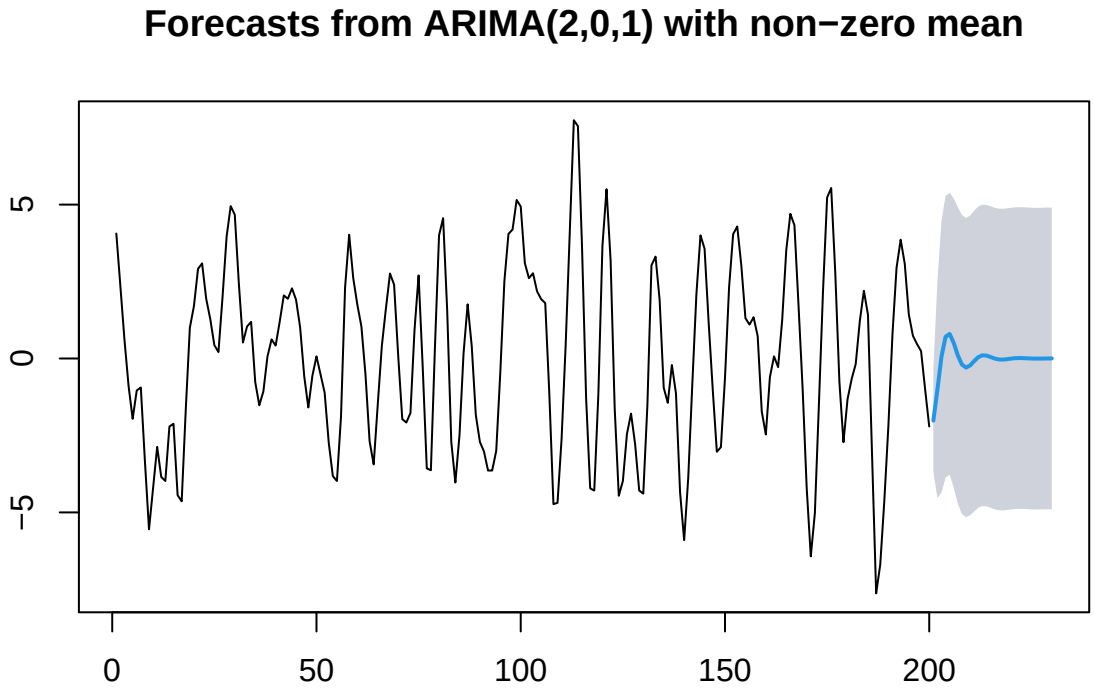
```
arima.model201$coef
```

```
##          ar1          ar2          ma1    intercept
##  1.184512005 -0.624314433  0.704949382  0.001096124
```

$$(1 - 1.184512005B + 0.624314433B^2)X_t = (1 + 0.704949382B)\varepsilon_t$$

5) Representa, en un mismo gráfico, los valores observados de la serie (en negro) y las predicciones para los 30 instantes de tiempo siguientes (incluyendo las bandas de confianza). ¿Las 30 predicciones contienen el mismo error?

```
plot(forecast(arima.model201, h = 30, level = 0.9))
```



6) Repite el análisis anterior pero usando una función de R para obtener el modelo ARIMA de manera automática. Comenta los resultados.

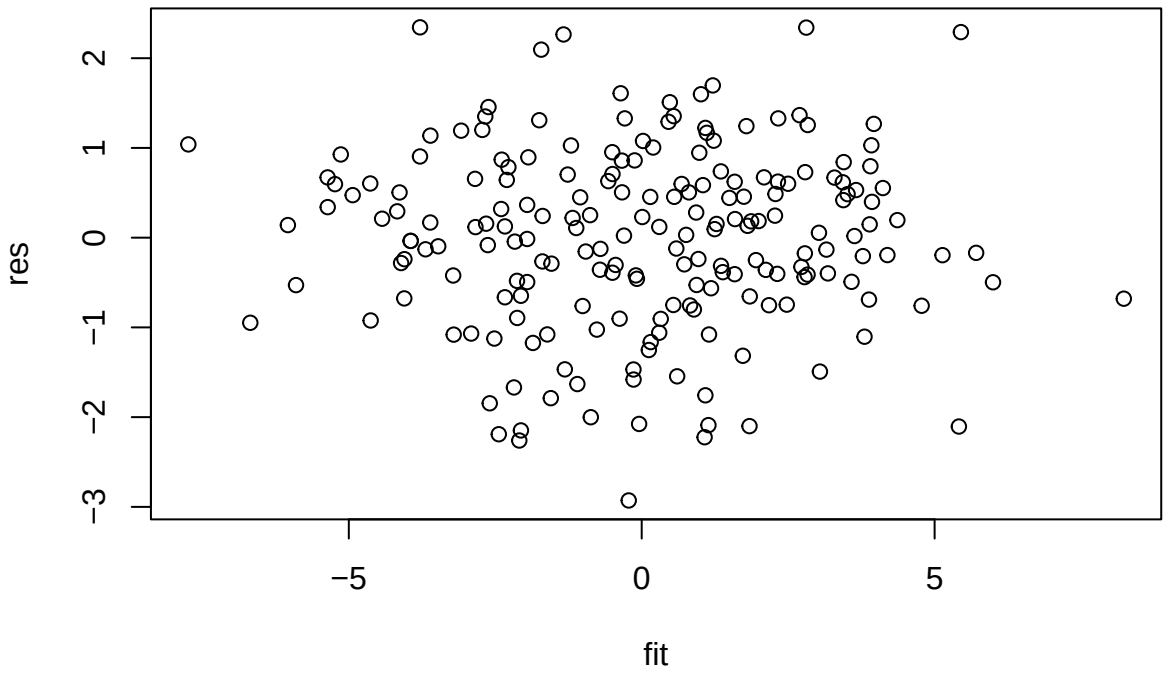
```r
modelo_auto <- auto.arima(datos.ts1)
modelo_auto
```

```
## Series: datos.ts1
## ARIMA(2,0,1) with zero mean
##
## Coefficients:
##          ar1      ar2     ma1
##       1.1845  -0.6243  0.7049
## s.e.  0.0593   0.0588  0.0524
##
## sigma^2 = 0.9991:  log likelihood = -284.16
## AIC=576.32   AICc=576.53   BIC=589.52
```

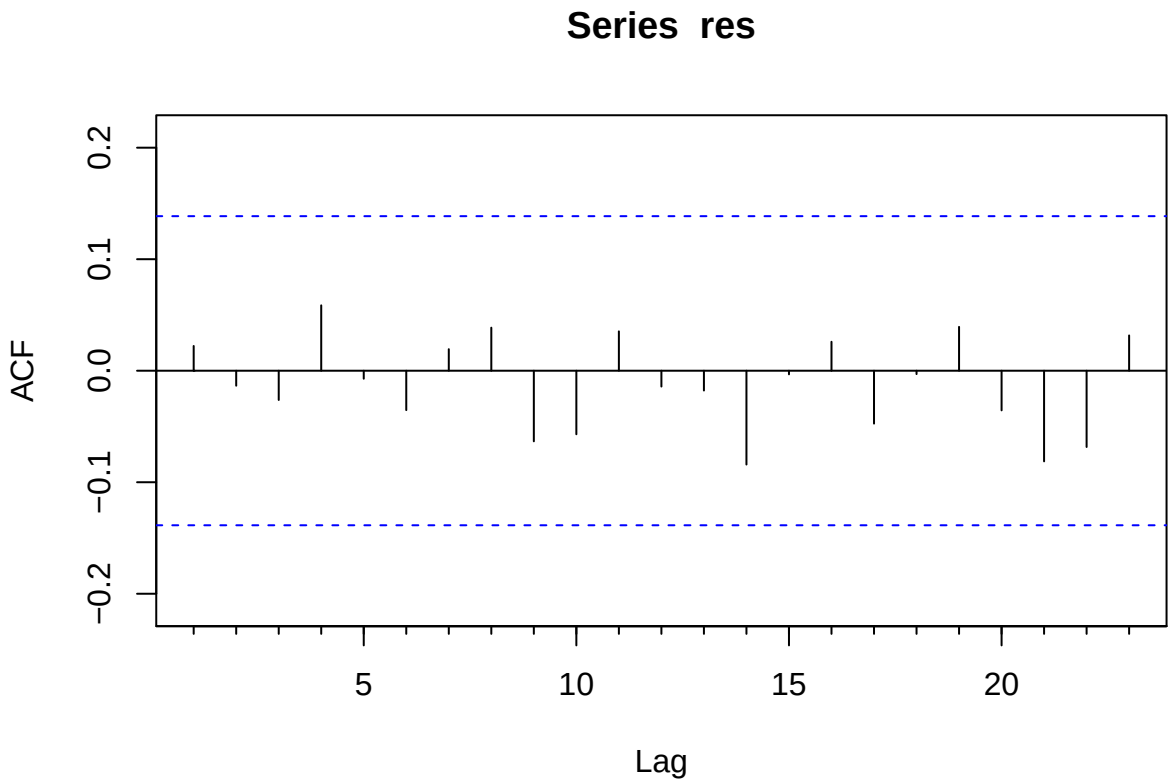
```r
res <- arima.model201$residuals
fit <- arima.model201$fitted

# Normlidad de los residuos: Si p-values > 0.05 -> se acepta normalidad
shapiro.test(res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.98892, p-value = 0.1236
```

```r
# Homecedasticidad: Si no hay ningún patrón evidente no diferencias de dispersión se acepta la
 ↪  homecedasticidad
plot(fit, res)
```




```r
# Independencia: Si residuos normales y correlograma con valores no significativos, se acpeta
 ↪  independencia
Acf(res)
```

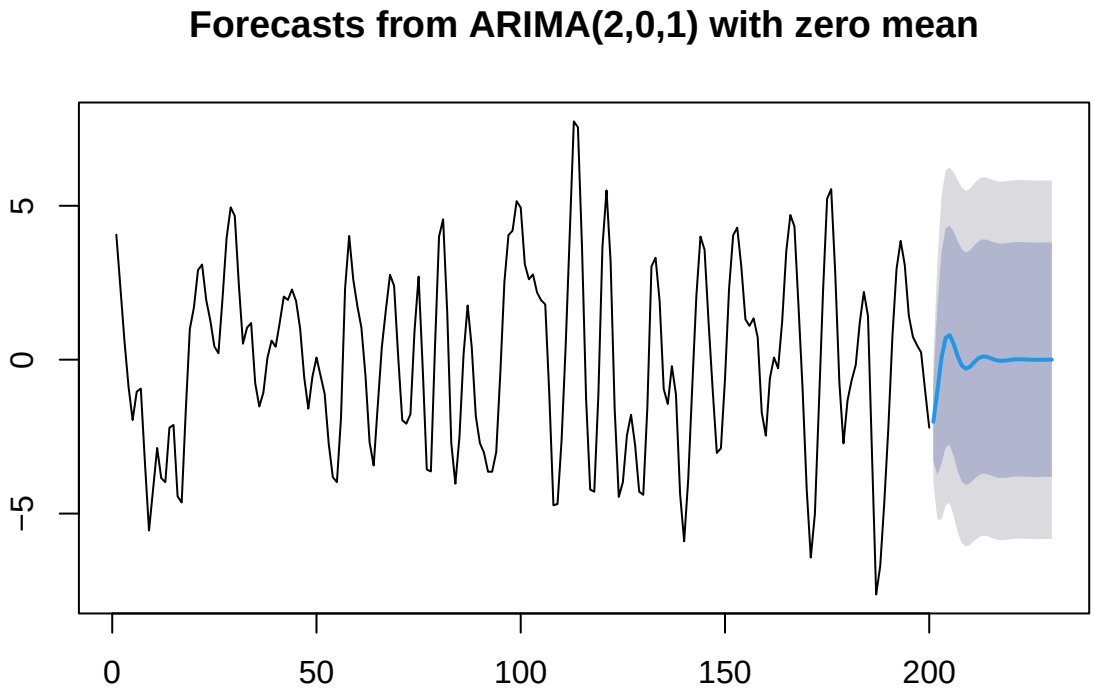



```
modelo_auto$coef
```

```
##        ar1        ar2        ma1
##  1.1845141 -0.6243176  0.7049484
```

$$(1 - 1.184512005B + 0.624314433B^2)X_t = (1 + 0.704949382B)\varepsilon_t$$

```
plot(forecast(modelo_auto, h = 30))
```

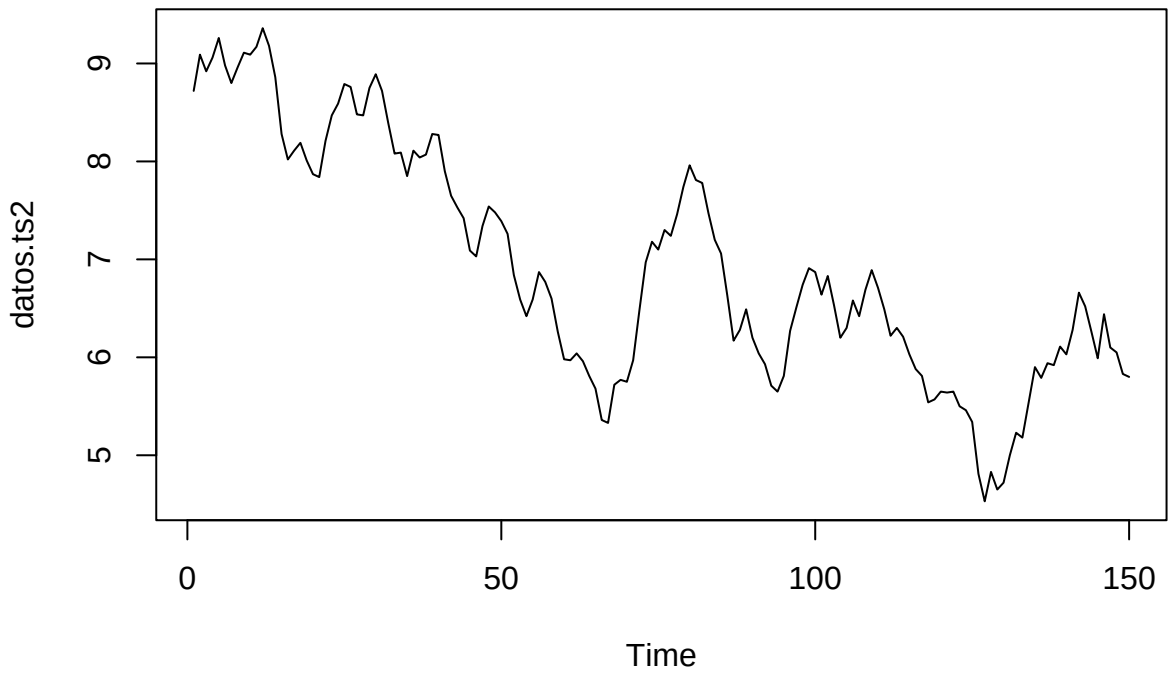


Problema 2

Usando los datos del fichero **ARIMA_P2.txt**, resuelve las mismas cuestiones planteadas en el Problema 1.

```
datos2 <- read.csv2("ARIMA_P2.txt", header = TRUE, dec = ".")
datos.ts2 <- ts(datos2$x, frequency = 1)
```

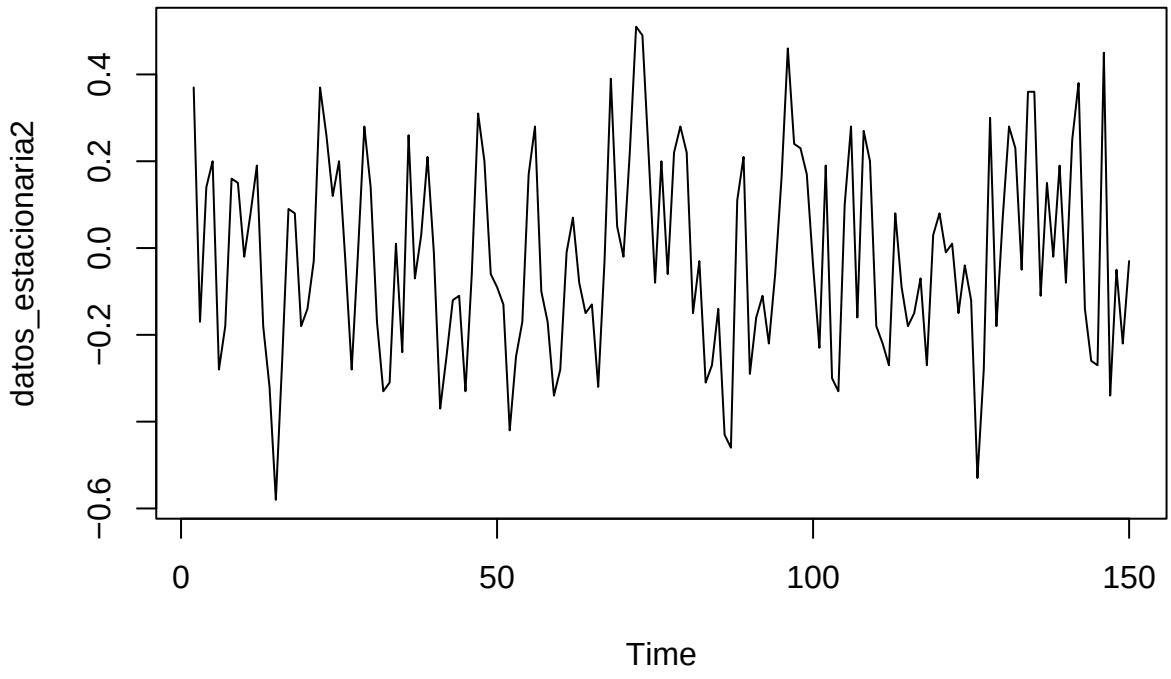
```r
ts.plot(datos.ts2)
```



- La serie presenta estacionalidad (hay un patrón periódico descendente).

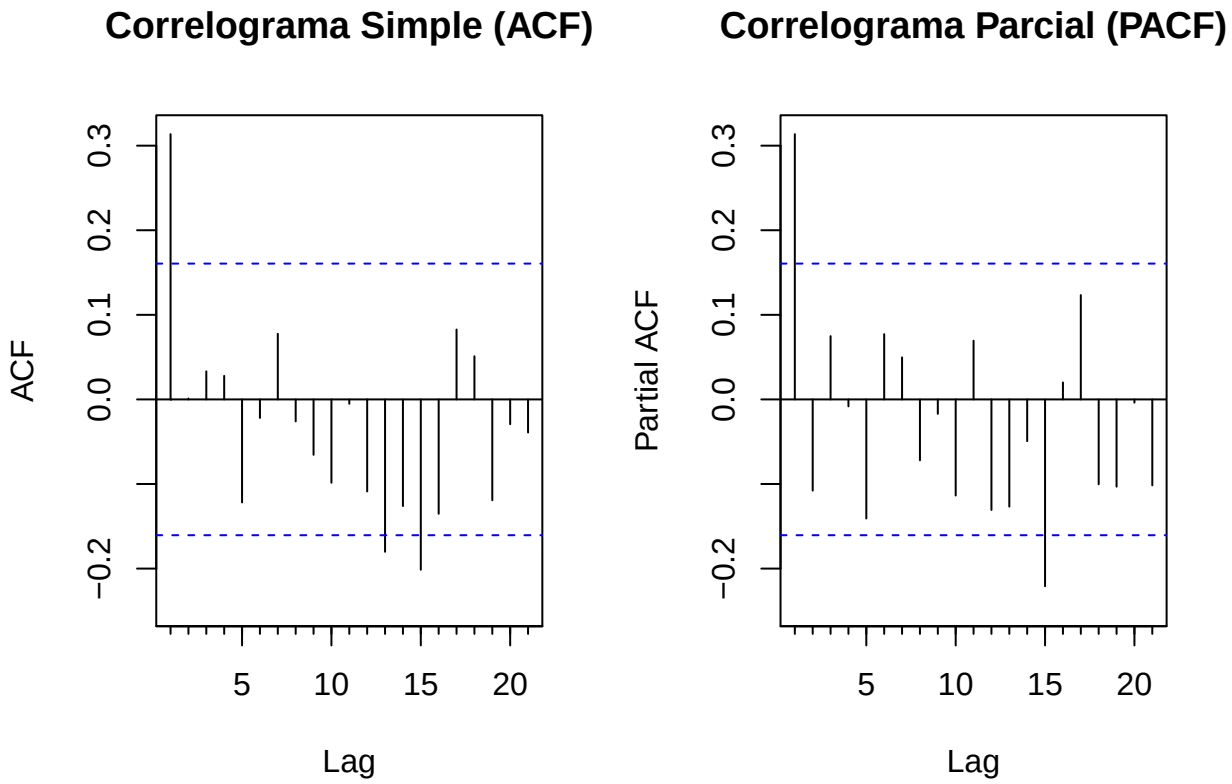
2) En el caso de que la serie no sea estacionaria, realiza las transformaciones que estimes oportunas para convertirla en estacionaria, comentando por qué las realizas.

```r
datos_estacionaria2 <- diff(datos.ts2, differences = 1)
ts.plot(datos_estacionaria2)
```



3) Obtén el correlograma simple y parcial de la serie estacionaria. En función de los resultados, ¿qué modelo(s) ARIMA propondrías como generador de la serie en estudio? Justifica tu respuesta.

```r
par(mfrow = c(1, 2))
Acf(datos_estacionaria2, main = "Correlograma Simple (ACF)")
Pacf(datos_estacionaria2, main = "Correlograma Parcial (PACF)")
```



Propondría un $ARMA(1,1)$.

```
arima.model101 <- Arima(datos_estacionaria2, order = c(1, 0, 1))
```

4) Para cada uno de los modelos teóricos propuestos en el apartado anterior, reponde a las siguientes cuestiones:

```
pvalores <- function(modelo) {
  # Extract coefficients
  coefs <- coef(modelo)
  # Extract standard errores
  se <- sqrt(diag(vcov(modelo)))
  # Calculate the t-values
  t_values <- coefs / se
  # Calculate the p-values
  p_values <- 2 * (1 - pnorm(abs(t_values)))
  # Create a data frame with coefficients, standard errors, t-values, and p-values
  results_pvalues <- data.frame(
    Coefficients = coefs,
    Std_Errors = se,
    T_values = t_values,
    P_values = p_values
  )
  print(results_pvalues)
}
```

a) Determina si el modelo es reducible, es decir, si existen coeficientes del modelo no significativos. En caso afirmativo, simplifica el modelo.

```
pvalores(arima.model101)
```

```
##           Coefficients Std_Errors    T_values   P_values
## ar1        0.004095238 0.19952518  0.02052492 0.98362463
## ma1        0.359945585 0.18298010  1.96712969 0.04916826
## intercept -0.018277724 0.02408499 -0.75888456 0.44792162
```

```
arima.model001 <- Arima(datos_estacionaria2, order = c(0, 0, 1))
pvalores(arima.model001)
```

```
##           Coefficients Std_Errors   T_values     P_values
## ma1         0.36342812 0.07579981  4.7945782 1.630175e-06
```

```
## intercept  -0.01829773 0.02404764 -0.7608951 4.467197e-01
```

b) Determina valores indicativos de la bondad del ajuste de los modelos que resultan del apartado anterior.

```
arima.model001
```

```
## Series: datos_estacionaria2
## ARIMA(0,0,1) with non-zero mean
##
## Coefficients:
##          ma1     mean
##       0.3634  -0.0183
## s.e.  0.0758   0.0240
##
## sigma^2 = 0.04714:  log likelihood = 17.09
## AIC=-28.17   AICc=-28.01   BIC=-19.16
```

```
paste("AIC =", arima.model001$aic, ", BIC =", arima.model001$bic)
```

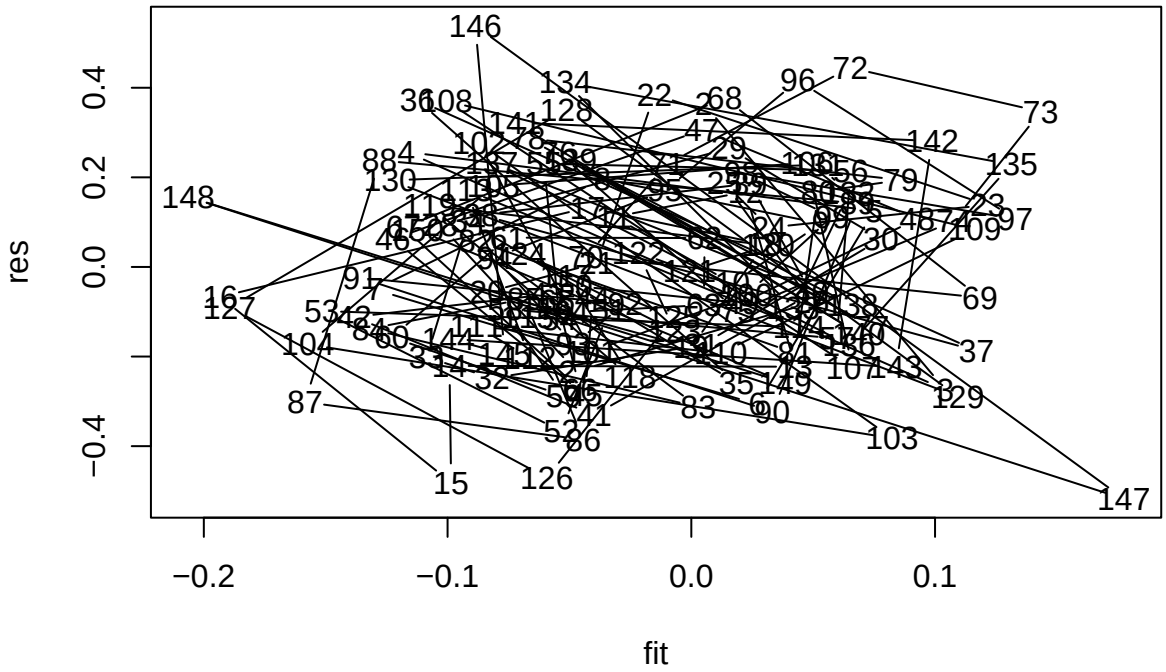
```
## [1] "AIC = -28.1730112044854 , BIC = -19.161172286649"
```

c) Analiza la validez de los modelos resultantes, es decir, comprueba que se verifican las hipótesis sobre los residuos.

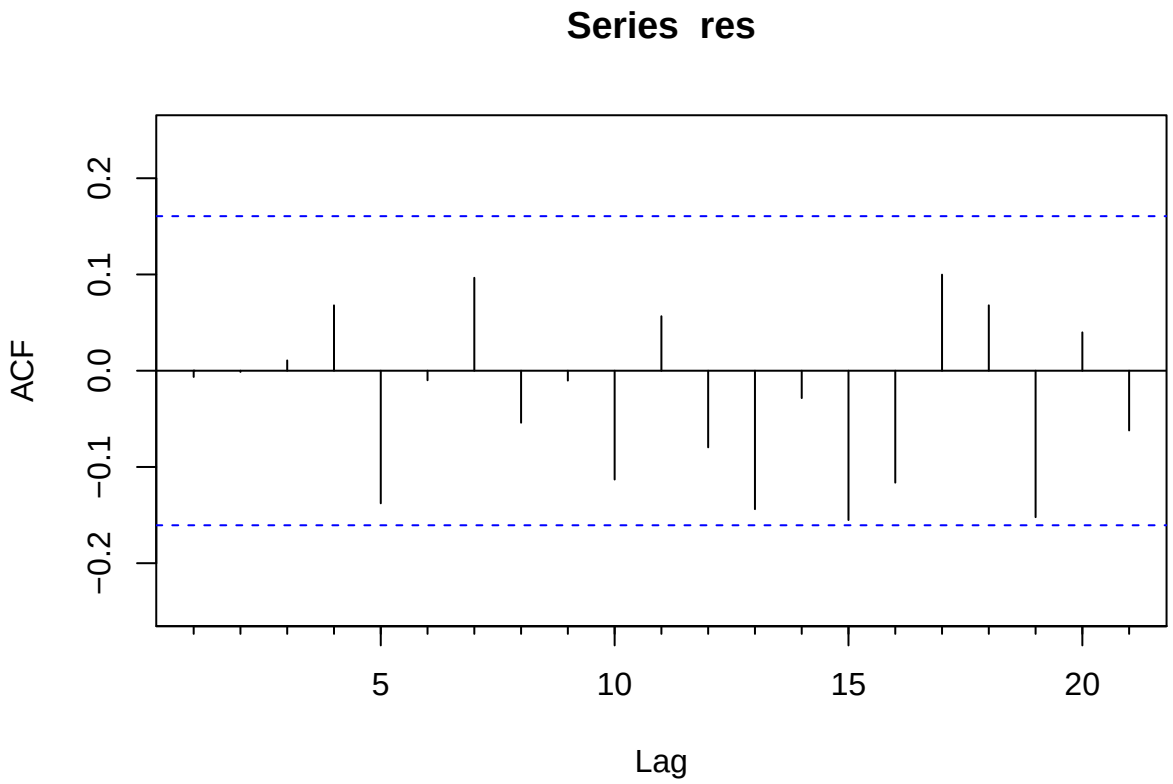
```
res <- arima.model001$residuals
fit <- arima.model001$fitted

# Normlidad de los residuos: Si p-values > 0.05 -> se acepta normalidad
shapiro.test(res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.98806, p-value = 0.2318
```

```
# Homecedasticidad: Si no hay ningún patrón evidente no diferencias de dispersión se acepta la
↪  homecedasticidad
plot(fit, res)
```




```
# Independencia: Si residuos normales y correlograma con valores no significativos, se acpeta
↪  independencia
Acf(res)
```



d) Tras el análisis realizado, indica cuál sería la expresión del modelo ARIMA estimado que propones como generador de la serie en estudio.

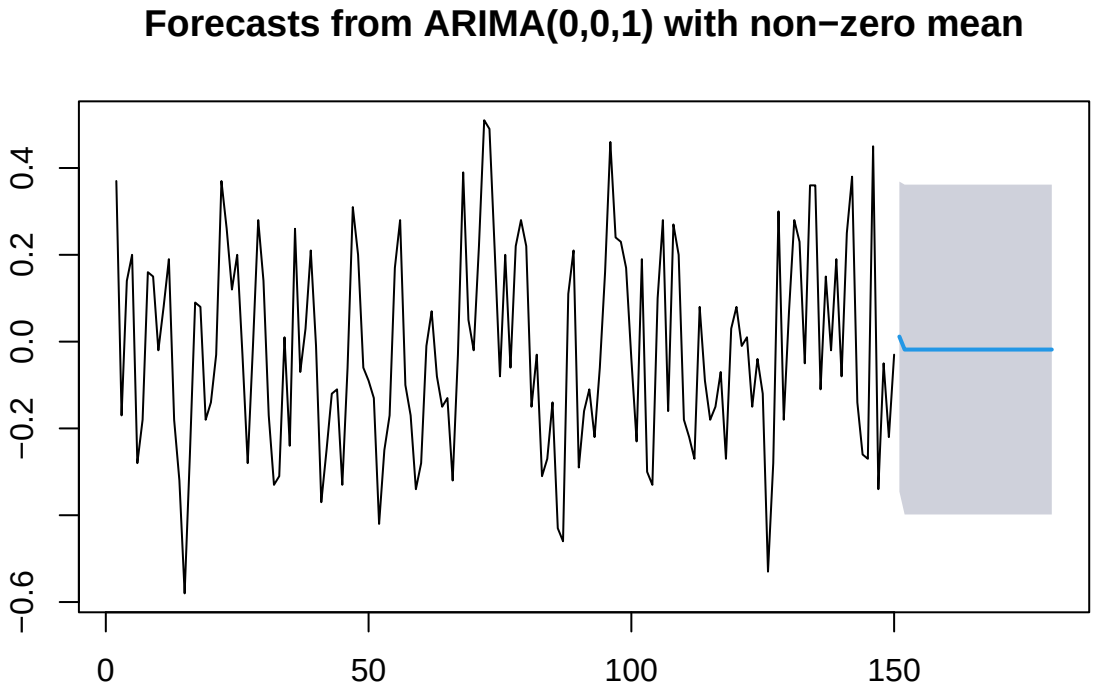
```
arima.model001$coef
```

```
##         ma1   intercept
##  0.36342812 -0.01829773
```

$$X_t = (1 + 0.36342812B)\varepsilon_t$$

5) Representa, en un mismo gráfico, los valores observados de la serie (en negro) y las predicciones para los 30 instantes de tiempo siguientes (incluyendo las bandas de confianza). ¿Las 30 predicciones contienen el mismo error?

```
plot(forecast(arima.model001, h = 30, level = 0.9))
```



6) Repite el análisis anterior pero usando una función de `R` para obtener el modelo ARIMA de manera automática. Comenta los resultados.

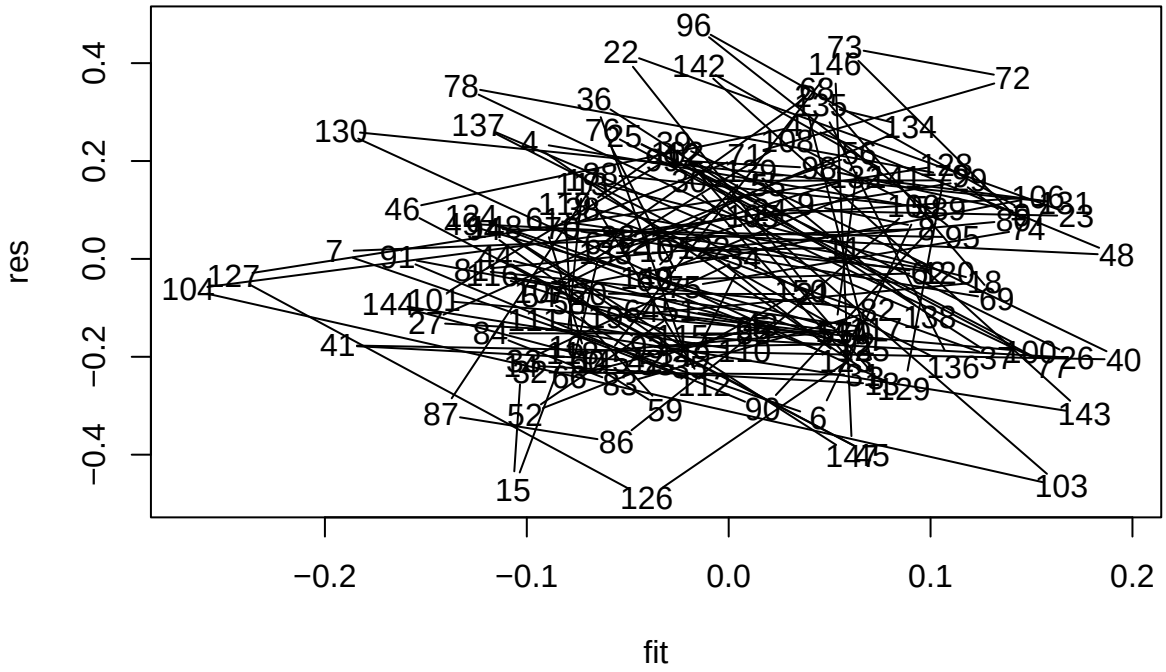
```r
modelo_auto2 <- auto.arima(datos_estacionaria2)
modelo_auto2
```

```
## Series: datos_estacionaria2
## ARIMA(2,0,2) with zero mean
##
## Coefficients:
##           ar1      ar2     ma1     ma2
##       -1.0730  -0.5727  1.4562  0.9107
## s.e.   0.0942   0.0948  0.0486  0.0513
##
## sigma^2 = 0.04447:  log likelihood = 21.77
## AIC=-33.54   AICc=-33.13   BIC=-18.53
```

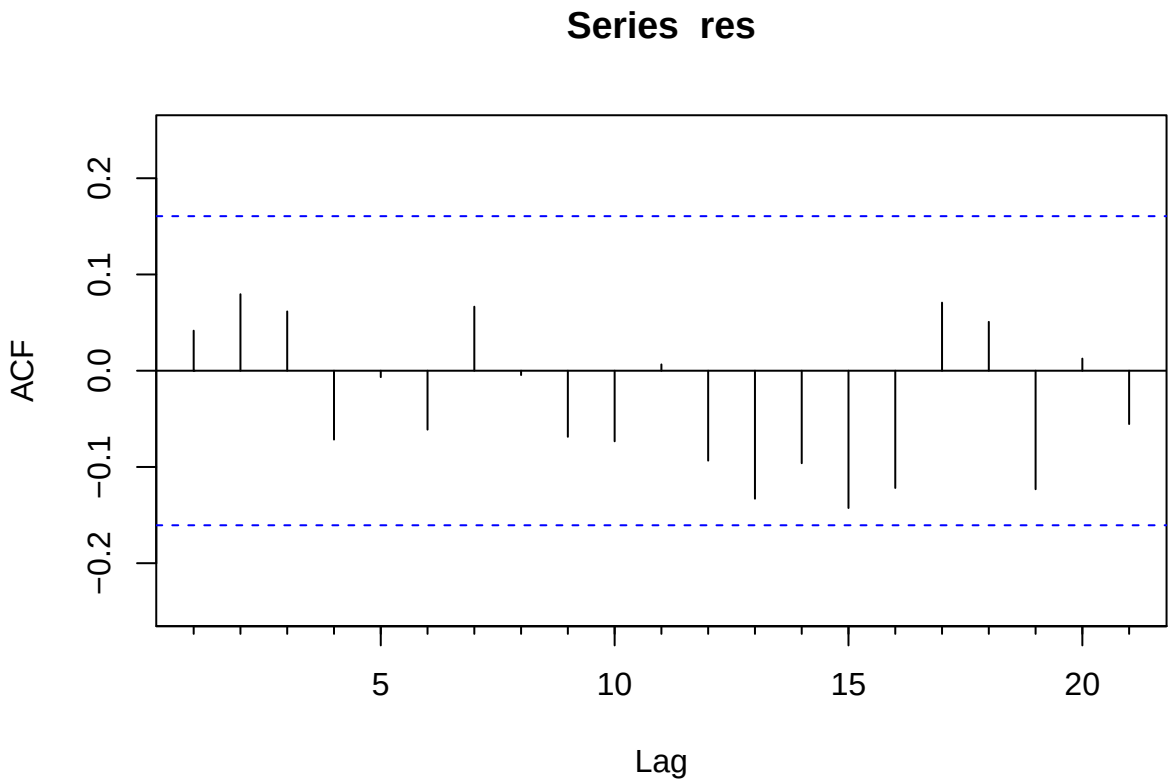
```r
res <- modelo_auto2$residuals
fit <- modelo_auto2$fitted

# Normlidad de los residuos: Si p-values > 0.05 -> se acepta normalidad
shapiro.test(res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.98747, p-value = 0.2002
```

```r
# Homecedasticidad: Si no hay ningún patrón evidente no diferencias de dispersión se acepta la homecedasticidad
plot(fit, res)
```




```r
# Independencia: Si residuos normales y correlograma con valores no significativos, se acpeta independencia
Acf(res)
```

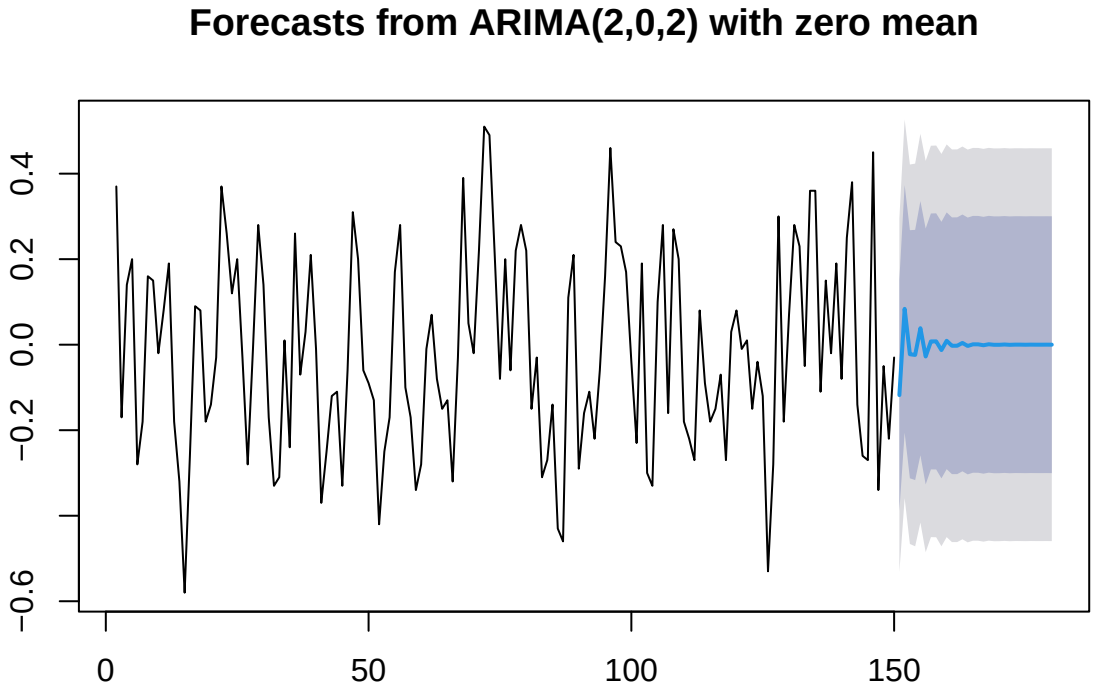



```
modelo_auto2$coef
```

```
##        ar1        ar2        ma1        ma2
## -1.0730041 -0.5727073  1.4561844  0.9106881
```

$$(1 + 1.0730041B + 0.5727073B^2)X_t = (1 + 1.4561844B + 0.9106881B^2)\varepsilon_t$$

```
plot(forecast(modelo_auto2, h = 30))
```

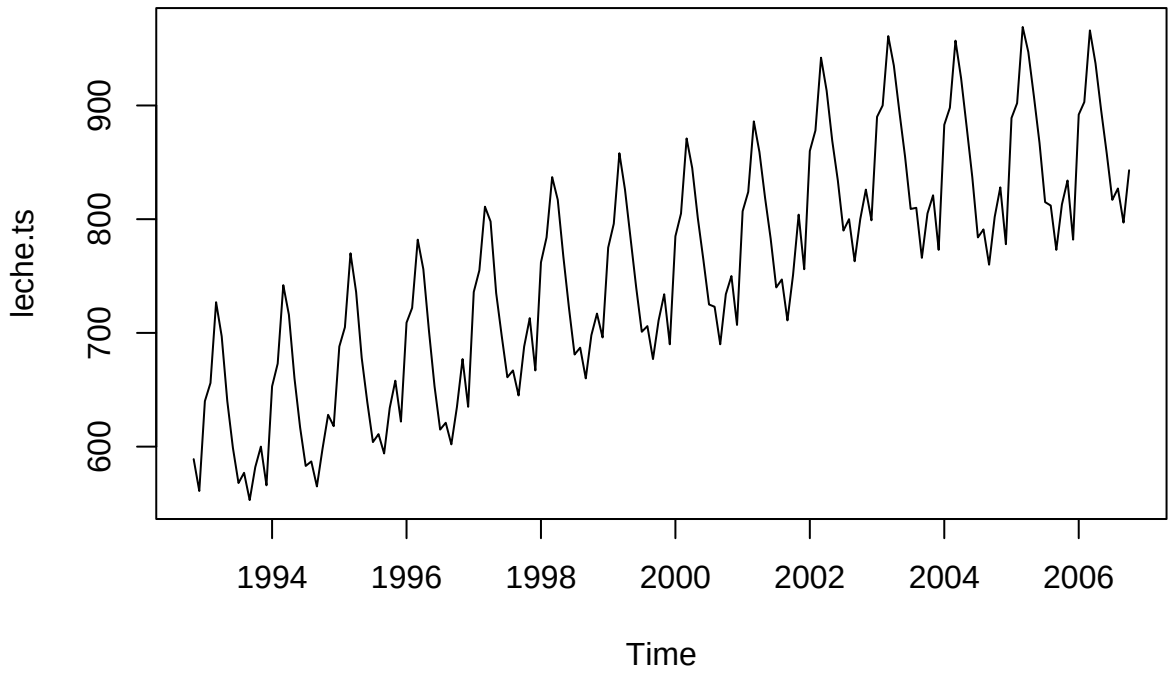


Problema 3

En el fichero **consumo_leche.txt** se encuentran los datos correspondientes al consumo de leche mensual (en miles de litros) por los habitantes de una determinada ciudad. Se dispone de datos sobre el consumo desde noviembre de 1992 hasta octubre de 2006.

1) Representa los datos del consumo de leche en un gráfico temproal y comenta los aspectos más relevantes.

```r
datos_leche <- read.table("consumo_leche.txt", header = TRUE, dec = ",")
leche.ts <- ts(datos_leche[, 1], start = c(1992, 11), frequency = 12)
ts.plot(leche.ts)
```



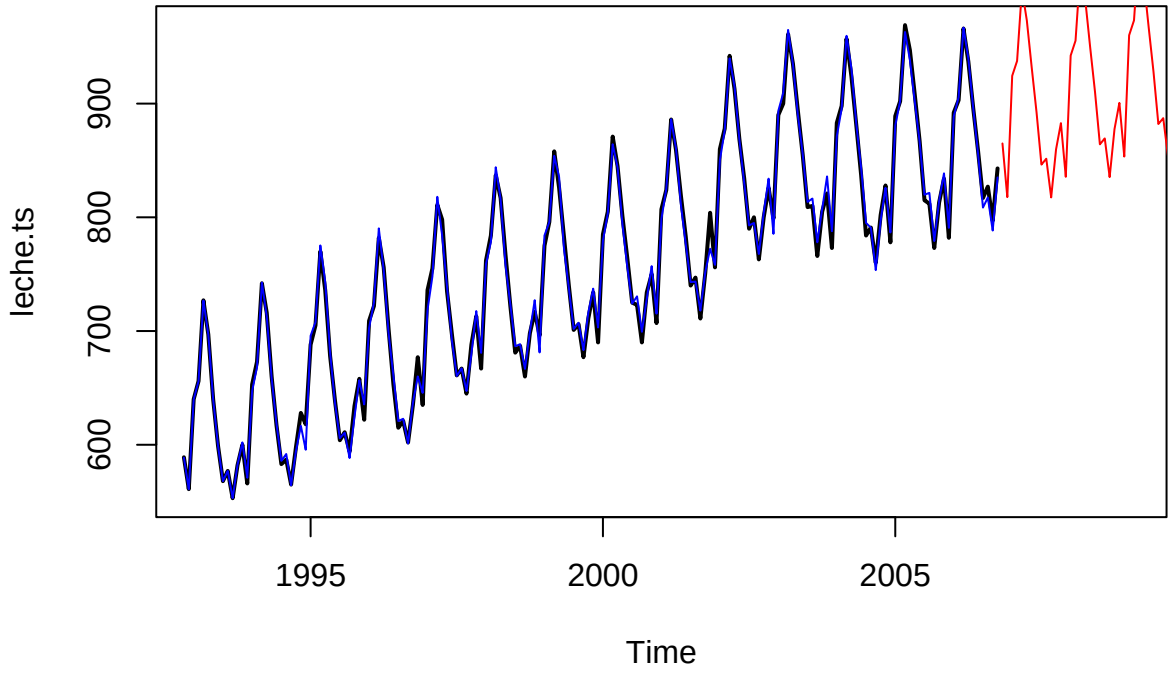
2) Proporciona un modelo ARIMA estacional (SARIMA) como generador de la serie en estudio. Utiliza una función de `R` para obtener el modelo de forma sencilla.

```r
model_leche <- auto.arima(leche.ts)
model_leche
```

```
## Series: leche.ts
## ARIMA(0,1,1)(0,1,1)[12]
##
## Coefficients:
##           ma1     sma1
##       -0.2204  -0.6214
## s.e.   0.0748   0.0627
##
## sigma^2 = 53.42:  log likelihood = -530.15
## AIC=1066.3   AICc=1066.46   BIC=1075.43
```

3) Representa en un mismo gráfico la secuencia de la serie observada (en negro), la serie ajustada (en azul) y de la serie predicha para los próximos 3 años (en rojo). A priori, ¿te parece que se trata de "buenas" predicciones? ¿Qué sucedería si se analizaran sólo los 5 últimos años de la serie?

```r
pred_leche <- forecast(model_leche, h = 36)
ts.plot(leche.ts, xlim = c(1993, 2009), lwd = 2) # Línea más gruesa para mejor visión

tiempo_leche <- 1:(length(datos_leche) + 36)
lines(model_leche$fitted, col = "blue")
lines(pred_leche$mean, col = "red")
```



4) Separemos ahora los datos del fichero completo en entrenamiento y test, dejando para test sólo los últimos 12 meses observados. En esta situación, compara las medidas de error (MAE y RMSE) del conjunto test que se obtendrían con los diferentes métodos (descomposición clásica, STL, alisado exponencial y ARIMA).

```
n_total <- length(leche.ts)
n_train <- n_total - 12
n_test <- 12

# Crear series de entrenamiento y test
leche_train <- window(leche.ts, end = c(2005, 10))
leche_test <- window(leche.ts, start = c(2005, 11))

cat("Tamaño conjunto entrenamiento:", length(leche_train))
```

```
## Tamaño conjunto entrenamiento: 156
```

```
cat("Tamaño conjunto test:", length(leche_test))
```

```
## Tamaño conjunto test: 12
```

```
# Método 1: Descomposición Clásica
# Descomponer la serie de entrenamiento
decomp_clasica <- decompose(leche_train, type = "multiplicative")

# Obtener último valor de tendencia no-NA
tendencia <- decomp_clasica$trend
tendencia_final <- tail(na.omit(tendencia), 1)

# Obtener índices estacionales (un ciclo completo)
estacional <- decomp_clasica$seasonal
indices_estacionales <- as.numeric(window(estacional, start = c(1993, 1), end = c(1993, 12)))

# Crear predicción para los meses de test (nov 2005 - oct 2006)
# Noviembre = mes 11, diciembre = mes 12, enero = mes 1, ..., octubre = mes 10
meses_test <- c(11, 12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
pred_clasica <- as.numeric(tendencia_final) * indices_estacionales[meses_test]

# Calcular errores
mae_clasica <- mean(abs(as.numeric(leche_test) - pred_clasica))
rmse_clasica <- sqrt(mean((as.numeric(leche_test) - pred_clasica)^2))

cat("MAE  =", round(mae_clasica, 4))
```

```
## MAE  = 16.2977
```

```
cat("RMSE =", round(rmse_clasica, 4))
```

```
## RMSE = 19.0656
```

```
# Método 2: STL
stl_decomp <- stl(leche_train, s.window = "periodic")

# Extraer componentes
tendencia_stl <- stl_decomp$time.series[, "trend"]
estacional_stl <- stl_decomp$time.series[, "seasonal"]

# Último valor de tendencia
tendencia_final_stl <- as.numeric(tail(tendencia_stl, 1))

# Índices estacionales (primer año completo)
indices_stl <- as.numeric(window(estacional_stl, start = c(1993, 1), end = c(1993, 12)))

# Predicción
pred_stl <- tendencia_final_stl + indices_stl[meses_test]

mae_stl <- mean(abs(as.numeric(leche_test) - pred_stl))
rmse_stl <- sqrt(mean((as.numeric(leche_test) - pred_stl)^2))

cat("MAE  =", round(mae_stl, 4))
```

```
## MAE  = 10.744
```

```
cat("RMSE =", round(rmse_stl, 4))
```

```
## RMSE = 12.8579
```

```
# Método 3: Alisado Exponencial (ETS)
modelo_ets <- ets(leche_train)
cat("Modelo ETS ajustado:", modelo_ets$method)
```

```
## Modelo ETS ajustado: ETS(A,A,A)
```

```
pred_ets <- forecast(modelo_ets, h = 12)

mae_ets <- mean(abs(as.numeric(leche_test) - as.numeric(pred_ets$mean)))
rmse_ets <- sqrt(mean((as.numeric(leche_test) - as.numeric(pred_ets$mean))^2))

cat("MAE  =", round(mae_ets, 4))
```

```
## MAE  = 6.6948
```

```
cat("RMSE =", round(rmse_ets, 4))
```

```
## RMSE = 8.6343
```

```
# Método 4: ARIMA/SARIMA
modelo_arima <- auto.arima(leche_train, seasonal = TRUE)
paste("Modelo:", modelo_arima)
```

```
## [1] "Modelo: ARIMA(2,0,0)(2,1,1)[12] with drift"
```

```
pred_arima <- forecast(modelo_arima, h = 12)

mae_arima <- mean(abs(as.numeric(leche_test) - as.numeric(pred_arima$mean)))
rmse_arima <- sqrt(mean((as.numeric(leche_test) - as.numeric(pred_arima$mean))^2))

cat("MAE  =", round(mae_arima, 4))
```

```
## MAE  = 14.9331
```

```r
cat("RMSE =", round(rmse_arima, 4))
```

```
## RMSE = 16.6822
```

```r
# Comparación de Métodos
resultados <- data.frame(
  Método = c("Decomp. Clásica", "STL", "ETS", "SARIMA"),
  MAE = round(c(mae_clasica, mae_stl, mae_ets, mae_arima), 4),
  RMSE = round(c(rmse_clasica, rmse_stl, rmse_ets, rmse_arima), 4)
)
resultados <- resultados[order(resultados$MAE), ]
resultados$Ranking <- 1:4

print(resultados, row.names = FALSE)
```

```
##           Método     MAE    RMSE Ranking
##              ETS  6.6948  8.6343       1
##              STL 10.7440 12.8579       2
##           SARIMA 14.9331 16.6822       3
##  Decomp. Clásica 16.2977 19.0656       4
```

```r
cat("Mejor método:", resultados$Método[1])
```

```
## Mejor método: ETS
```