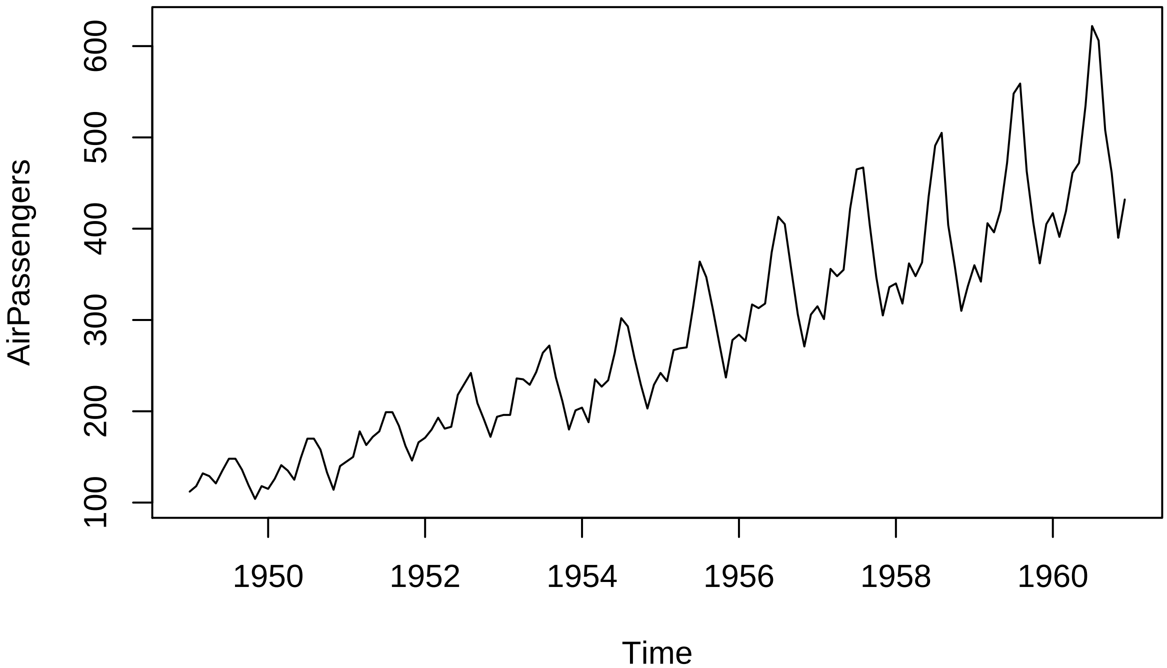
TimeSeries

Sergio Solano

9 de marzo de 2017

Pruebas series de tiempo de Time Series Analysis and its applications

```r
# Plot AirPassengers
plot(AirPassengers)
```




```r
# View the start and end dates of AirPassengers
start(AirPassengers)
```

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

```r
end(AirPassengers)
```

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

```r
# Use time(), deltat(), frequency(), and cycle() with AirPassengers
time(AirPassengers)
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 1949 1949.000 1949.083 1949.167 1949.250 1949.333 1949.417 1949.500
## 1950 1950.000 1950.083 1950.167 1950.250 1950.333 1950.417 1950.500
## 1951 1951.000 1951.083 1951.167 1951.250 1951.333 1951.417 1951.500
## 1952 1952.000 1952.083 1952.167 1952.250 1952.333 1952.417 1952.500
## 1953 1953.000 1953.083 1953.167 1953.250 1953.333 1953.417 1953.500
## 1954 1954.000 1954.083 1954.167 1954.250 1954.333 1954.417 1954.500
## 1955 1955.000 1955.083 1955.167 1955.250 1955.333 1955.417 1955.500
## 1956 1956.000 1956.083 1956.167 1956.250 1956.333 1956.417 1956.500
## 1957 1957.000 1957.083 1957.167 1957.250 1957.333 1957.417 1957.500
```

```
## 1958 1958.000 1958.083 1958.167 1958.250 1958.333 1958.417 1958.500
## 1959 1959.000 1959.083 1959.167 1959.250 1959.333 1959.417 1959.500
## 1960 1960.000 1960.083 1960.167 1960.250 1960.333 1960.417 1960.500
##           Aug      Sep      Oct      Nov      Dec
## 1949 1949.583 1949.667 1949.750 1949.833 1949.917
## 1950 1950.583 1950.667 1950.750 1950.833 1950.917
## 1951 1951.583 1951.667 1951.750 1951.833 1951.917
## 1952 1952.583 1952.667 1952.750 1952.833 1952.917
## 1953 1953.583 1953.667 1953.750 1953.833 1953.917
## 1954 1954.583 1954.667 1954.750 1954.833 1954.917
## 1955 1955.583 1955.667 1955.750 1955.833 1955.917
## 1956 1956.583 1956.667 1956.750 1956.833 1956.917
## 1957 1957.583 1957.667 1957.750 1957.833 1957.917
## 1958 1958.583 1958.667 1958.750 1958.833 1958.917
## 1959 1959.583 1959.667 1959.750 1959.833 1959.917
## 1960 1960.583 1960.667 1960.750 1960.833 1960.917
```

```r
deltat(AirPassengers)
```

```
## [1] 0.08333333
```

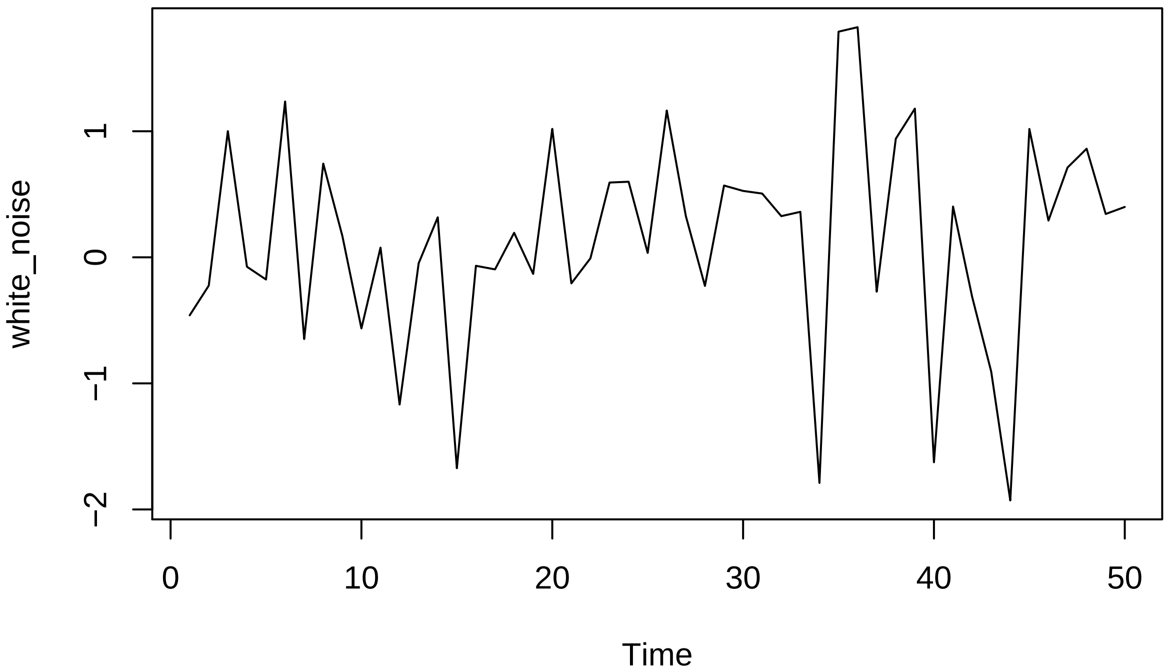
```r
frequency(AirPassengers)
```

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

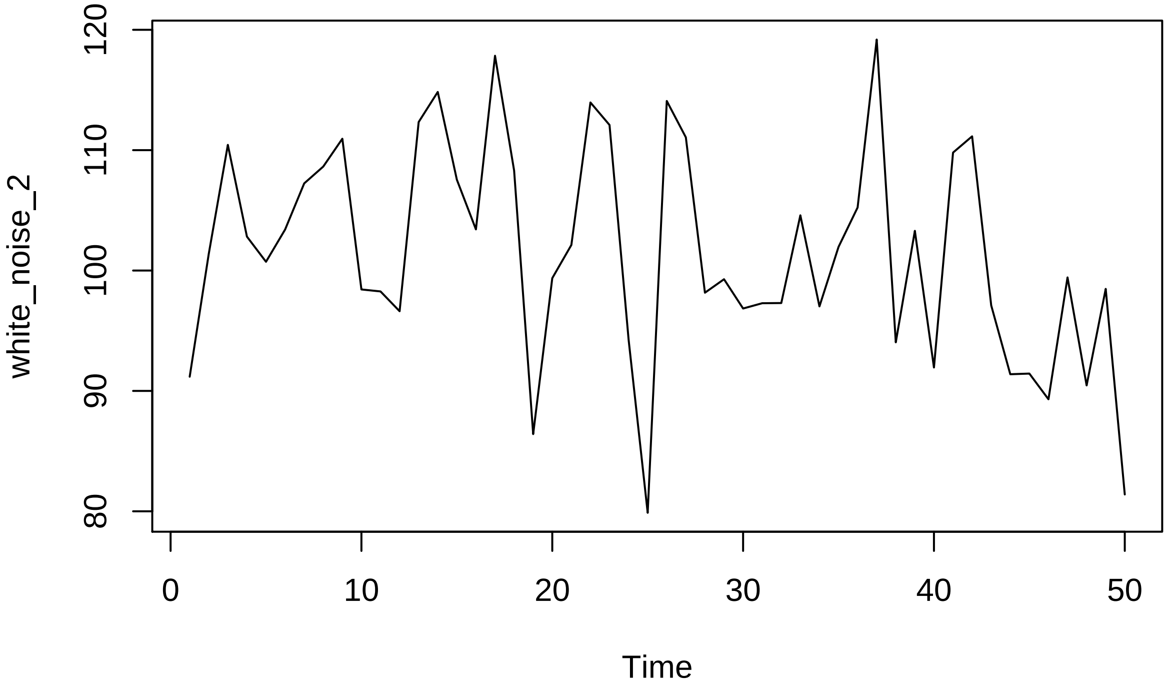
```r
cycle(AirPassengers)
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1949   1   2   3   4   5   6   7   8   9  10  11  12
## 1950   1   2   3   4   5   6   7   8   9  10  11  12
## 1951   1   2   3   4   5   6   7   8   9  10  11  12
## 1952   1   2   3   4   5   6   7   8   9  10  11  12
## 1953   1   2   3   4   5   6   7   8   9  10  11  12
## 1954   1   2   3   4   5   6   7   8   9  10  11  12
## 1955   1   2   3   4   5   6   7   8   9  10  11  12
## 1956   1   2   3   4   5   6   7   8   9  10  11  12
## 1957   1   2   3   4   5   6   7   8   9  10  11  12
## 1958   1   2   3   4   5   6   7   8   9  10  11  12
## 1959   1   2   3   4   5   6   7   8   9  10  11  12
## 1960   1   2   3   4   5   6   7   8   9  10  11  12
```

```r
# Simulate a WN model with list(order = c(0, 0, 0))
white_noise <- arima.sim(model = list(order = c(0,0,0)), n = 50)

# Plot your white_noise data

ts.plot(white_noise)
```



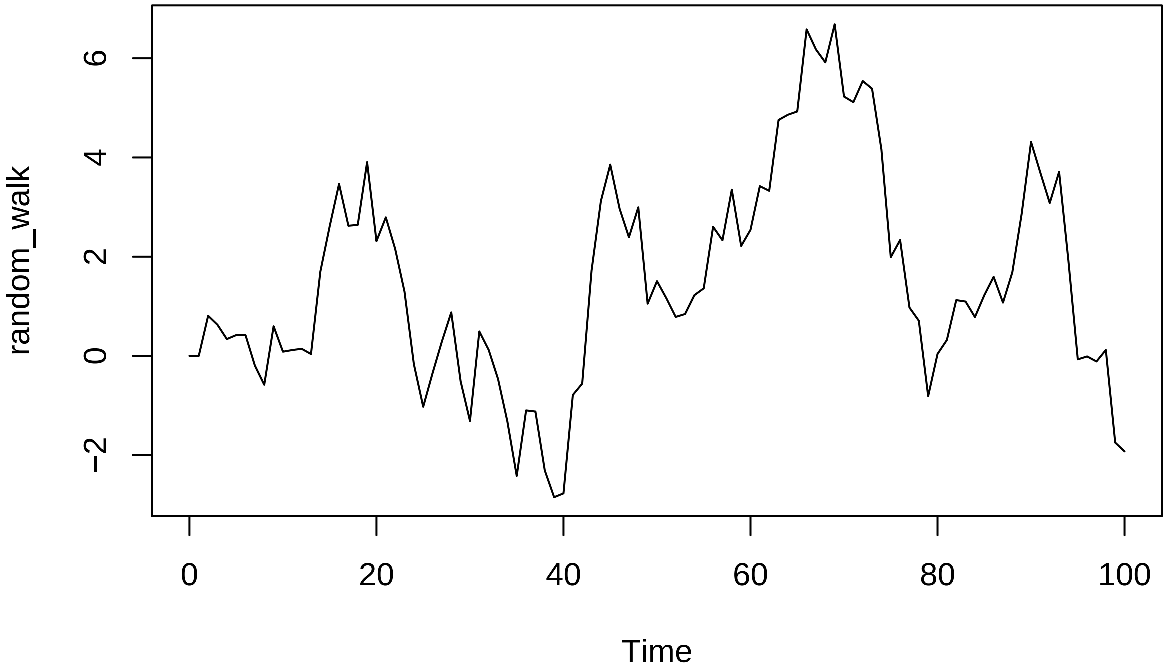

```
# Simulate from the WN model with: mean = 100, sd = 10
white_noise_2 <- arima.sim(model = list(order = c(0,0,0)), n = 50, mean = 100, sd = 10)

# Plot your white_noise_2 data
ts.plot(white_noise_2)
```



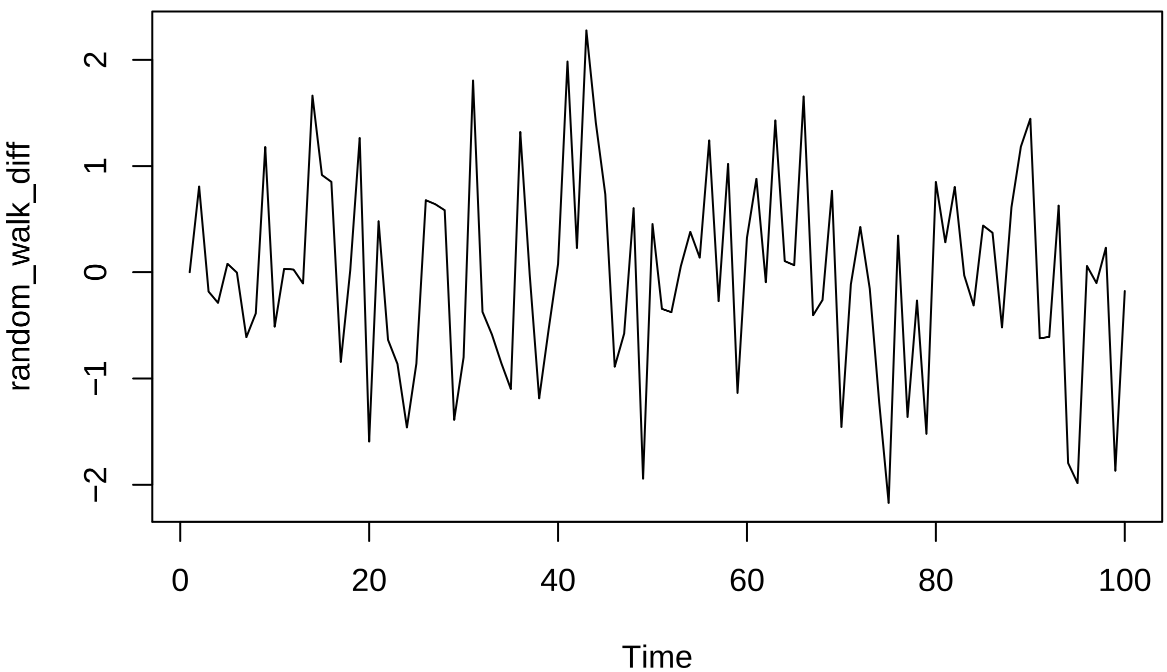
Los cambios en una serie de tiempo de Random Walk siguen un comportamiento de White noise.

```
# Generate a RW model using arima.sim
random_walk <- arima.sim(model = list(order = c(0, 1, 0)) , n = 100)

# Plot random_walk
```

```r
ts.plot (random_walk)
```

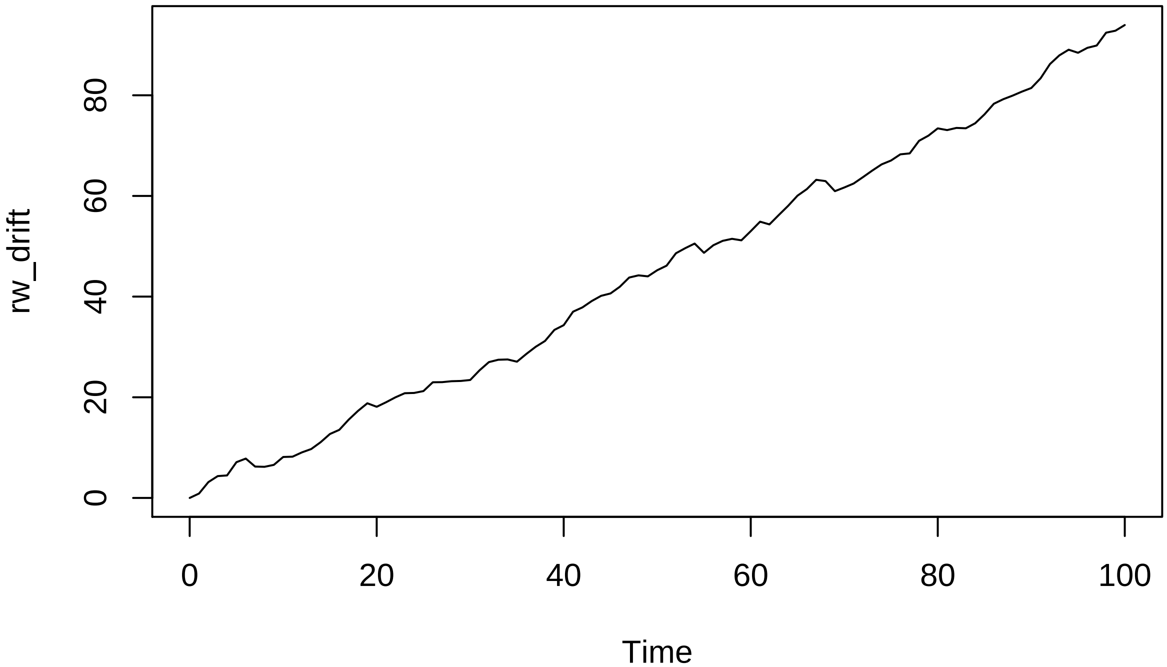



```r
# Calculate the first difference series
random_walk_diff <- diff(random_walk)

# Plot random_walk_diff

ts.plot(random_walk_diff)
```

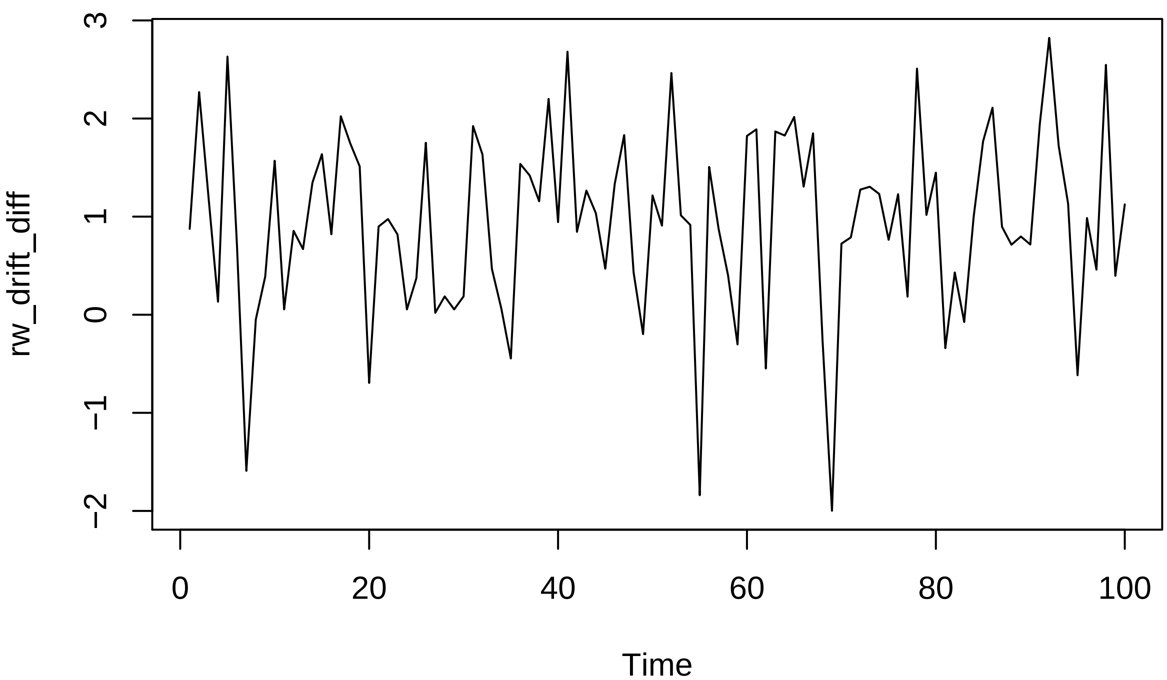



```r
# RANDOM WALK WITH DRIFT
```

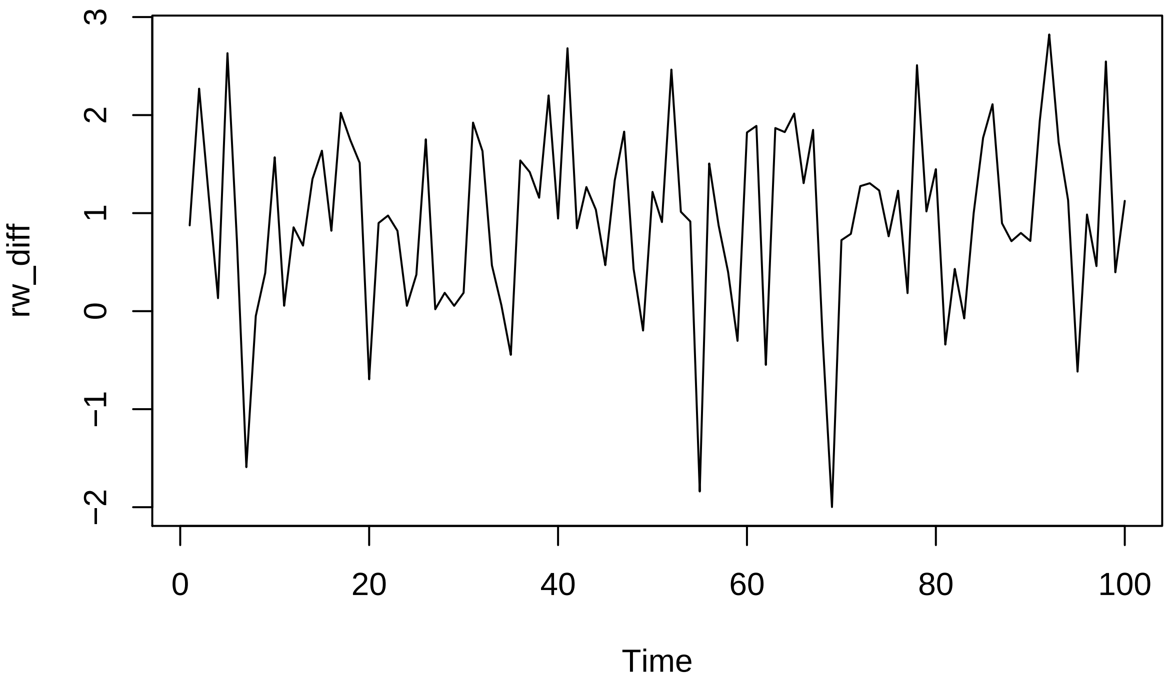
```r
# Generate a RW model with a drift uing arima.sim
rw_drift <- arima.sim(model = list(order = c(0, 1, 0)), n = 100, mean = 1)

# Plot rw_drift
ts.plot(rw_drift)
```



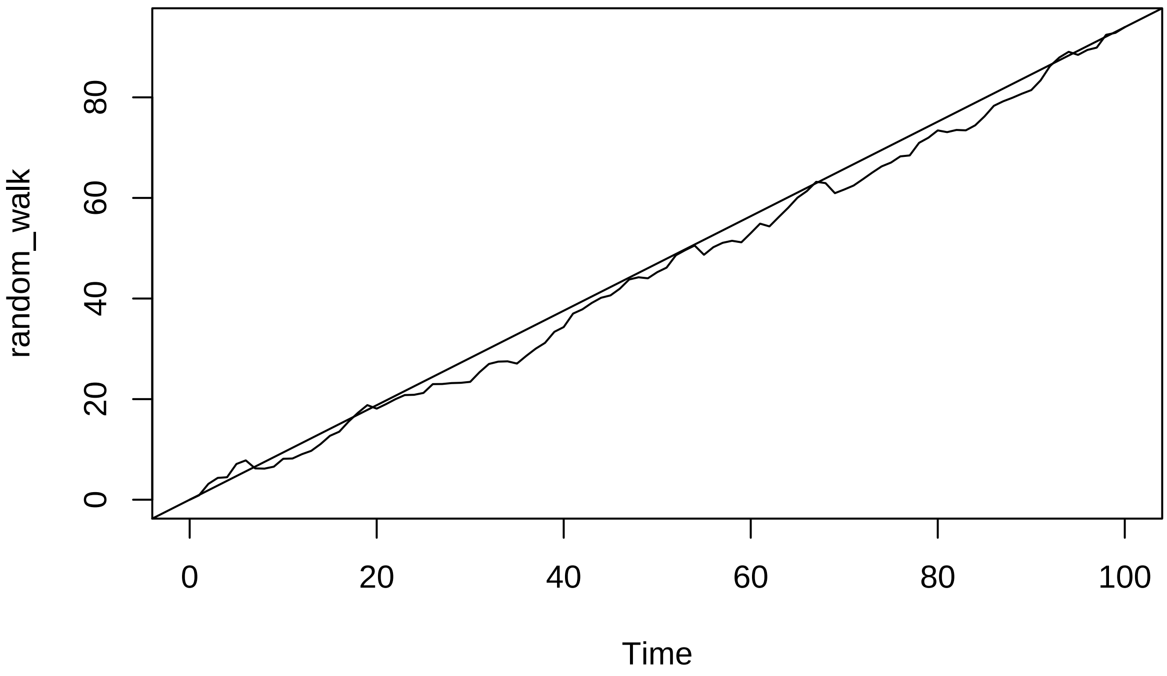

```r
# Calculate the first difference series
rw_drift_diff <- diff(rw_drift)

# Plot rw_drift_diff
ts.plot(rw_drift_diff)
```



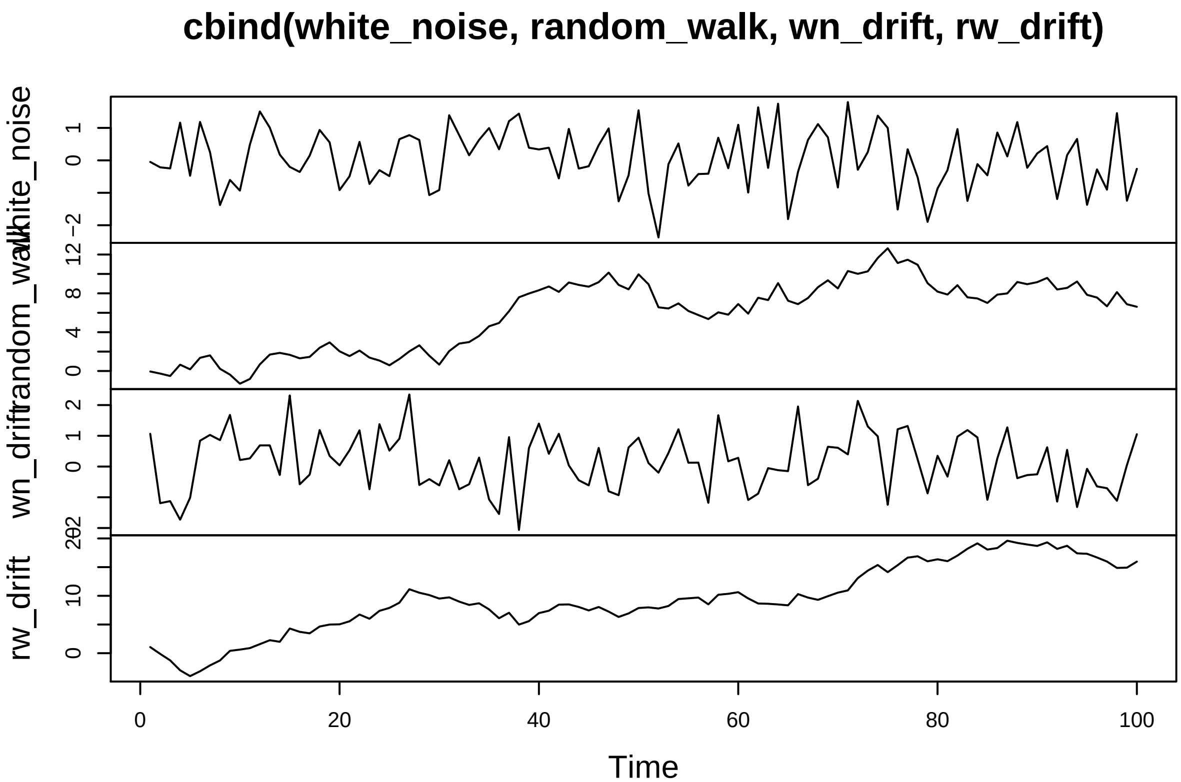
```r
random_walk <- rw_drift

# Difference your random_walk data
rw_diff <- diff(random_walk)


# Plot rw_diff
plot.ts(rw_diff)
```




```r
# Now fit the WN model to the differenced data
model_wn <- arima(rw_diff,order = c(0, 0, 0))

# Store the value of the estimated time trend (intercept)
int_wn <- model_wn$coef

# Plot the original random_walk data
ts.plot(random_walk)

# Use abline(0, ...) to add time trend to the figure
abline(0,int_wn)
```

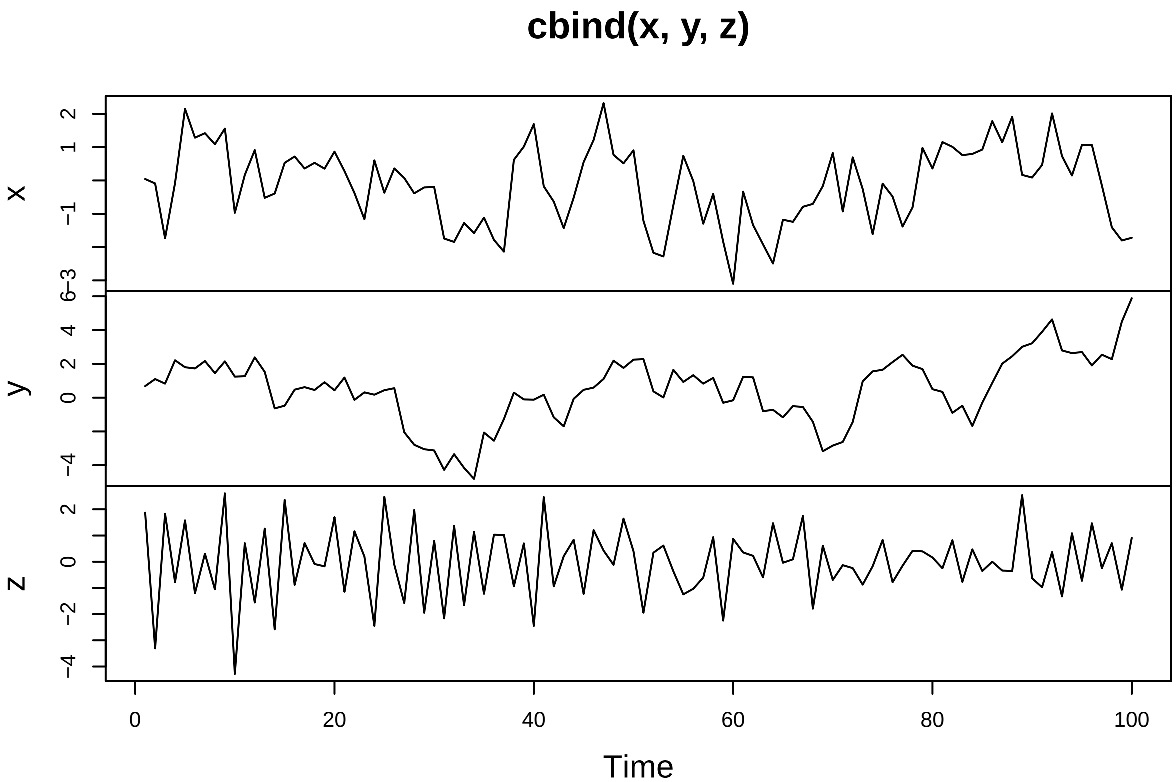



```
#  RANDOM WALK CON Y SIN DRIFT (Estationarity)
#
# The white noise (WN) and random walk (RW) models are very closely related. However, only the RW is al
#
# Recall that if we start with a mean zero WN process and compute its running or cumulative sum, the re

# Use arima.sim() to generate WN data
white_noise <- arima.sim(model = list(order = c(0, 0, 0)), n = 100)

# Use cumsum() to convert your WN data to RW
random_walk <- cumsum(white_noise)

# Use arima.sim() to generate WN drift data
wn_drift <- arima.sim(model = list(order = c(0, 0, 0)),mean=0.4, n = 100)

# Use cumsum() to convert your WN drift data to RW
rw_drift <- cumsum(wn_drift)

# Plot all four data objects
plot.ts(cbind(white_noise, random_walk, wn_drift, rw_drift))
```

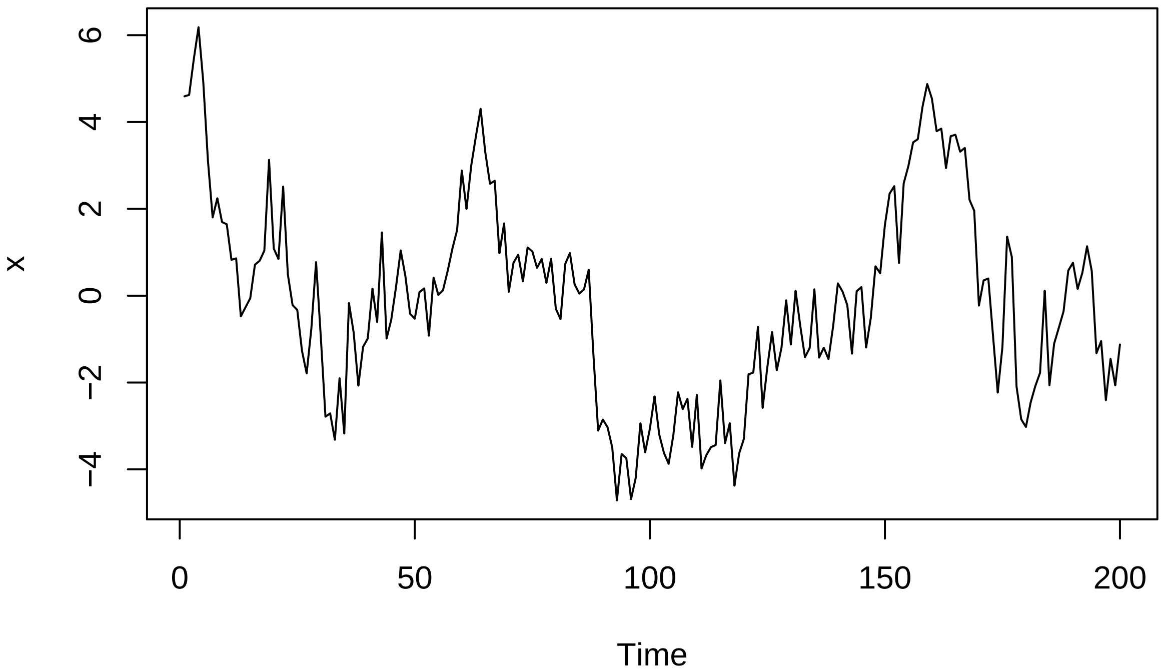



```
#
# # Generate means from eu_percentreturns
# colMeans(eu_percentreturns)
#
# # Use apply to calculate sample variance from eu_percentreturns
# apply(eu_percentreturns, MARGIN = 2, FUN = var)
#
# # Use apply to calculate standard deviation from eu_percentreturns
# apply(eu_percentreturns, MARGIN = 2, FUN = sd)
#
# # Display a histogram of percent returns for each index
# par(mfrow = c(2,2))
# apply(eu_percentreturns, MARGIN = 2, FUN = hist, main = "", xlab = "Percentage Return")
#
# # Display normal quantile plots of percent returns for each index
# par(mfrow = c(2,2))
# apply(eu_percentreturns, MARGIN = 2, FUN = qqnorm, main = "")
# qqline(eu_percentreturns)

# pairs(eu_stocks)


# MODELOS AUTOREGRESIVOS:

# Simulate an AR model with 0.5 slope
x <- arima.sim(model = list(ar = 0.5), n = 100)

# Simulate an AR model with 0.9 slope
y <- arima.sim(model = list(ar = 0.9), n = 100)
```

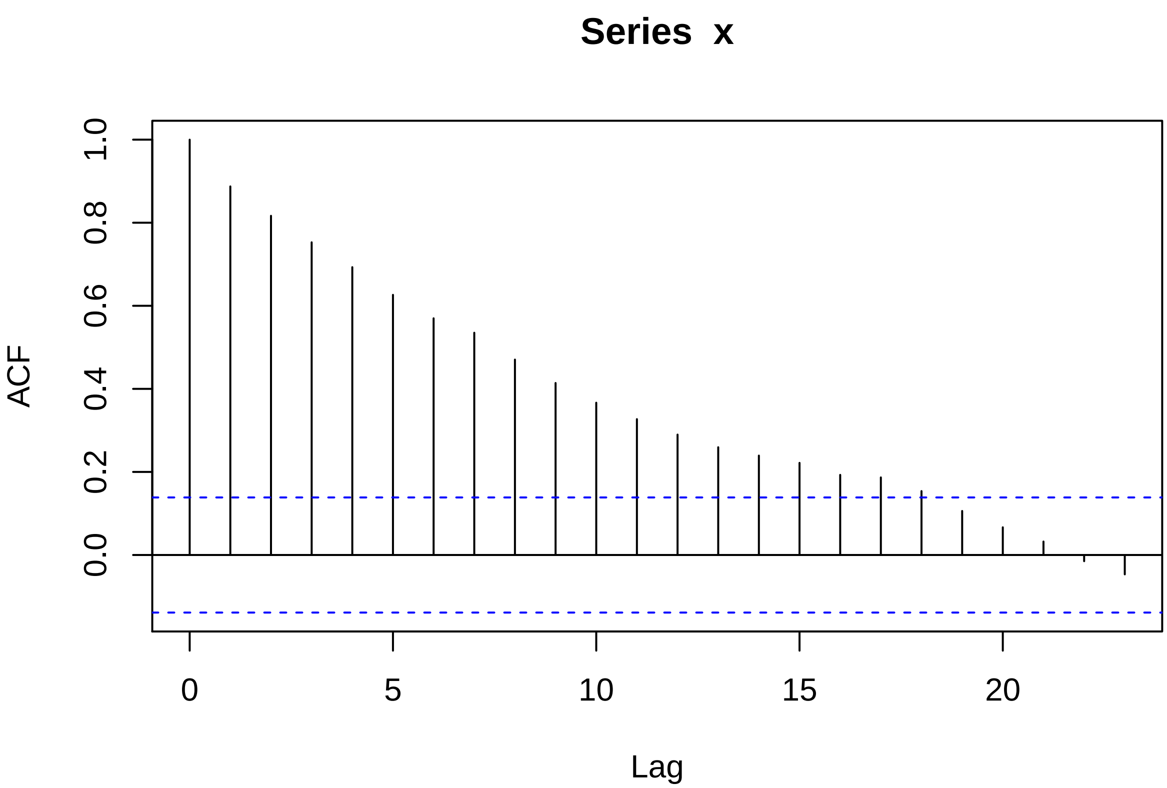
```r
# Simulate an AR model with -0.75 slope
z <- arima.sim(model = list(ar = -0.75), n = 100)

# Plot your simulated data
plot.ts(cbind(x, y, z))
```



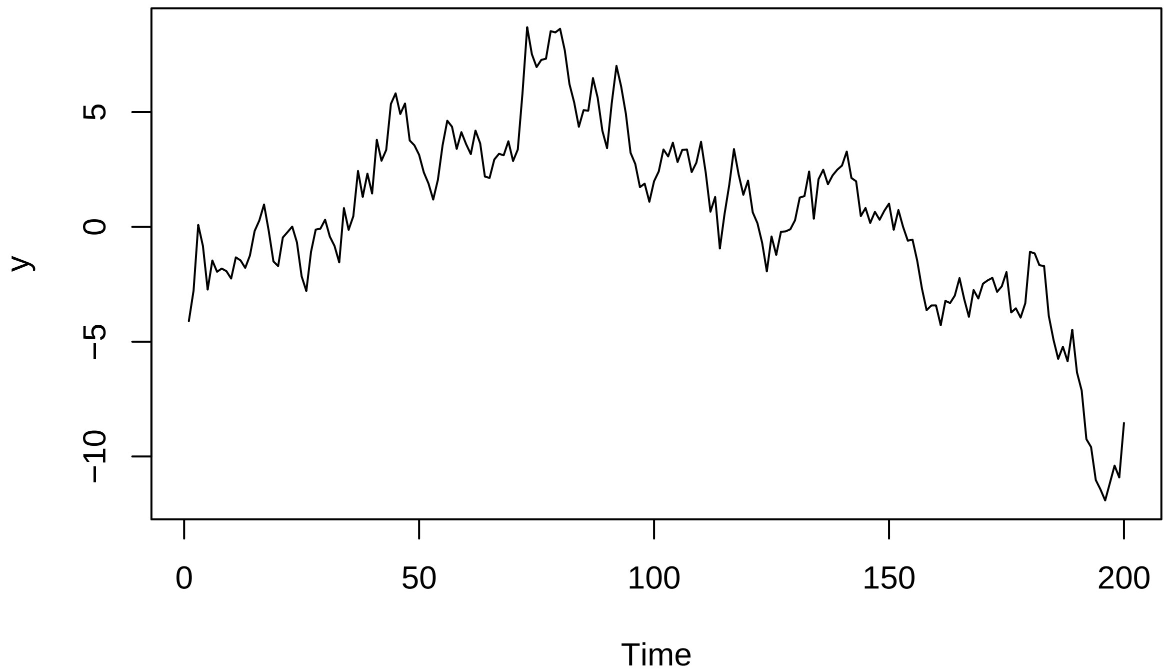

```
###### COMPARAR MODELOS AR con RandomWalks

# Simulate and plot AR model with slope 0.9
x <- arima.sim(model = list(ar = 0.9), n = 200)
ts.plot(x)
```



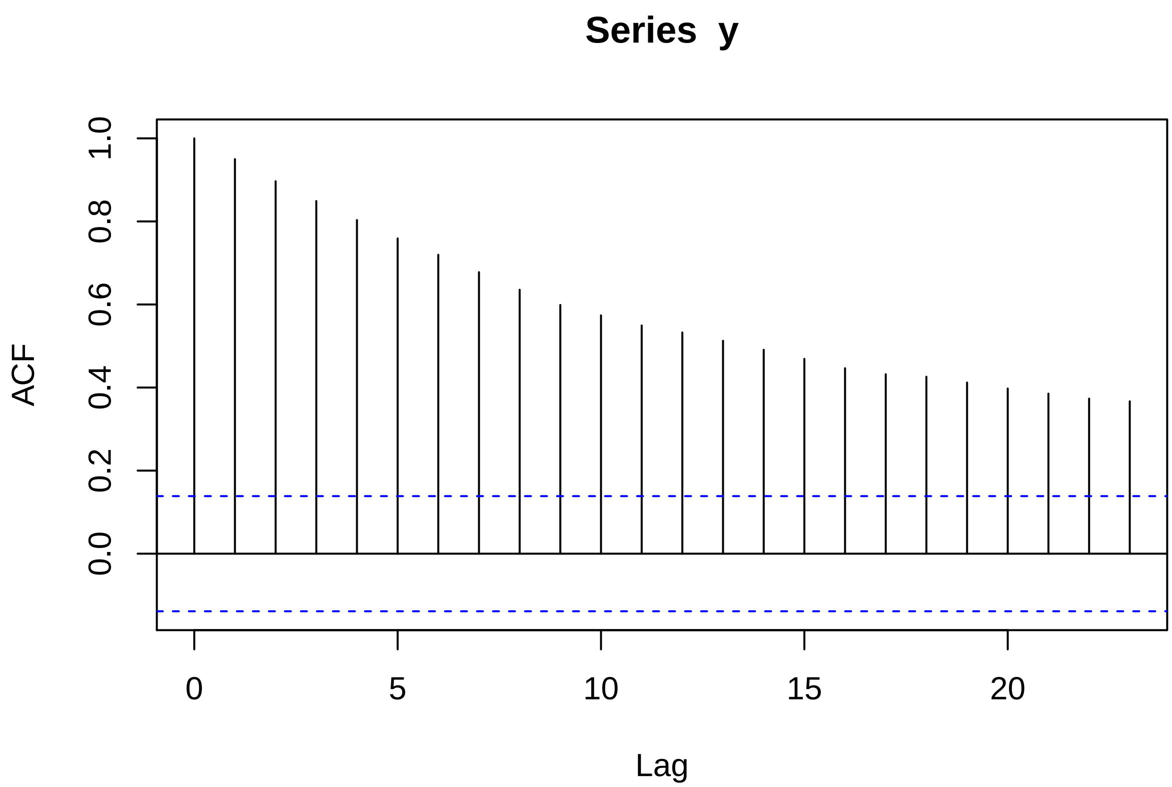

```
acf(x)
```



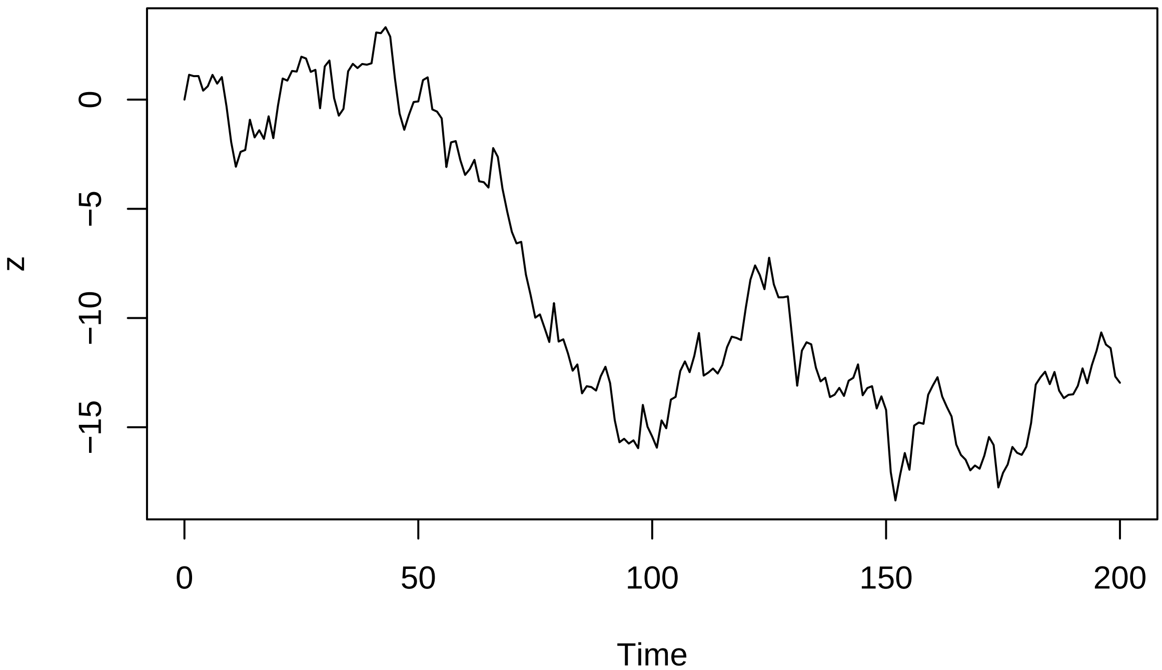

```
# Simulate and plot AR model with slope 0.98
y <- arima.sim(model = list(ar = 0.98), n = 200)
ts.plot(y)
```



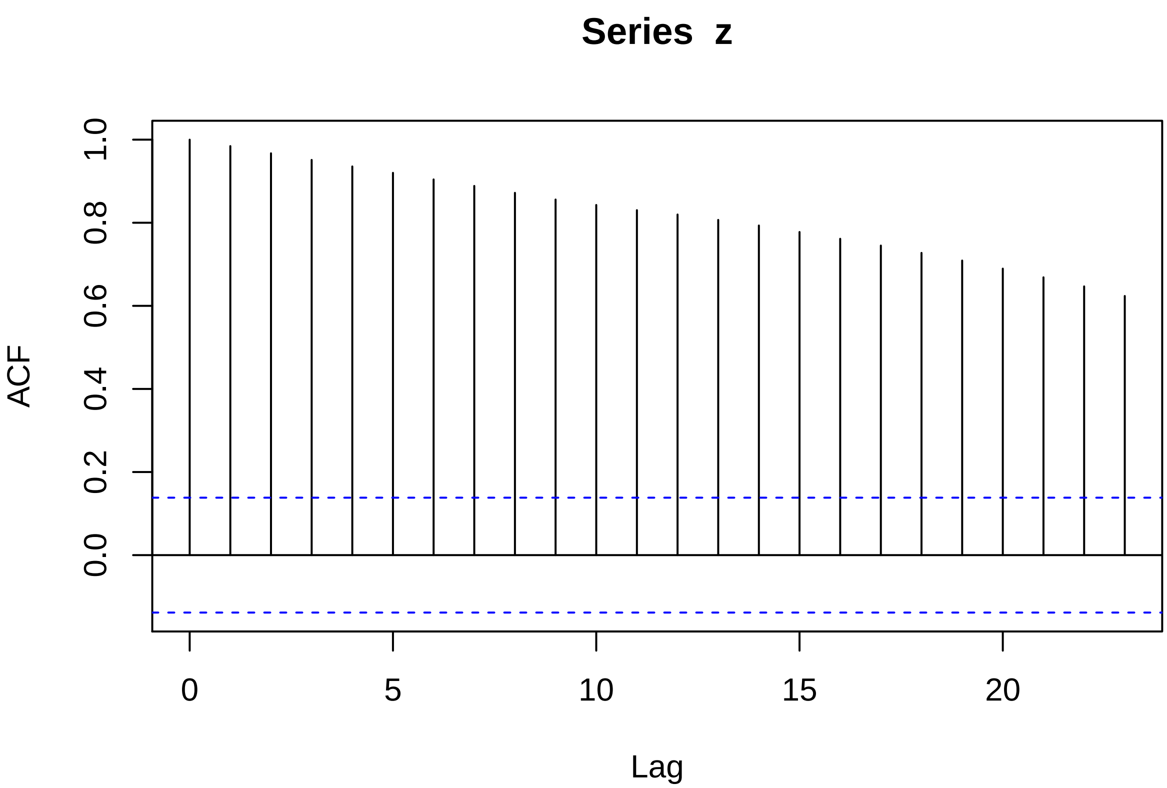

```
acf(y)
```




```
# Simulate and plot RW model
z <- arima.sim(model = list(order = c(0, 1, 0)), n = 200)
ts.plot(z)
```

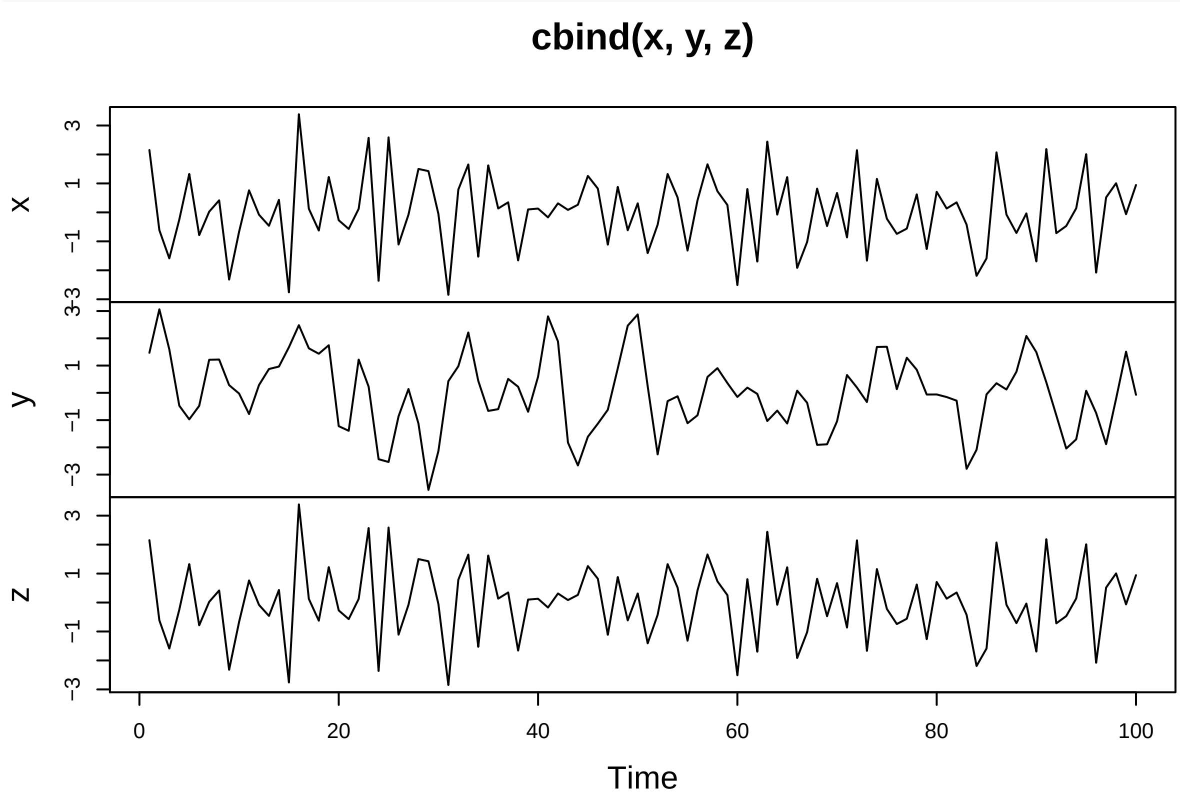



```
acf(z)
```




```
# # AJUSTAR UN AR
# # Fit an AR model to Nile
# AR_fit <-arima(Nile, order  = c(1,0,0))
# print(AR_fit)
#
# # Use predict() to make a 1-step forecast
```

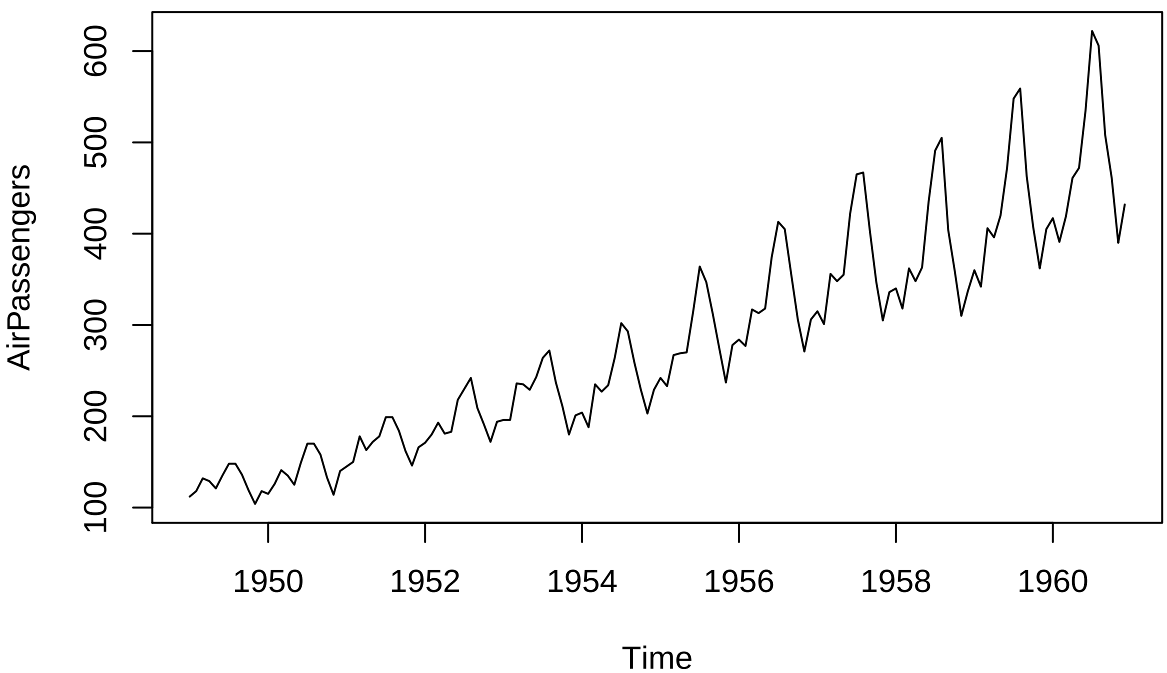
```r
# predict_AR <- predict(AR_fit)
#
# # Obtain the 1-step forecast using $pred[1]
# predict_AR$pred[1]
#
# # Use predict to make 1-step through 10-step forecasts
# predict(AR_fit, n.ahead = 10)
#
# # Run to plot the Nile series plus the forecast and 95% prediction intervals
# ts.plot(Nile, xlim = c(1871, 1980))
# AR_forecast <- predict(AR_fit, n.ahead = 10)$pred
# AR_forecast_se <- predict(AR_fit, n.ahead = 10)$se
# points(AR_forecast, type = "l", col = 2)
# points(AR_forecast - 2*AR_forecast_se, type = "l", col = 2, lty = 2)
# points(AR_forecast + 2*AR_forecast_se, type = "l", col = 2, lty = 2)

# # AJUSTAR UN MA
# Generate MA model with slope 0.5
x <- arima.sim(model = list(ma = 0.5), n = 100)

# Generate MA model with slope 0.9
y <- x <- arima.sim(model = list(ma = 0.9), n = 100)

# Generate MA model with slope -0.5
z <- x <- arima.sim(model = list(ma = -0.5), n = 100)

# Plot all three models together
plot.ts(cbind(x, y, z))
```



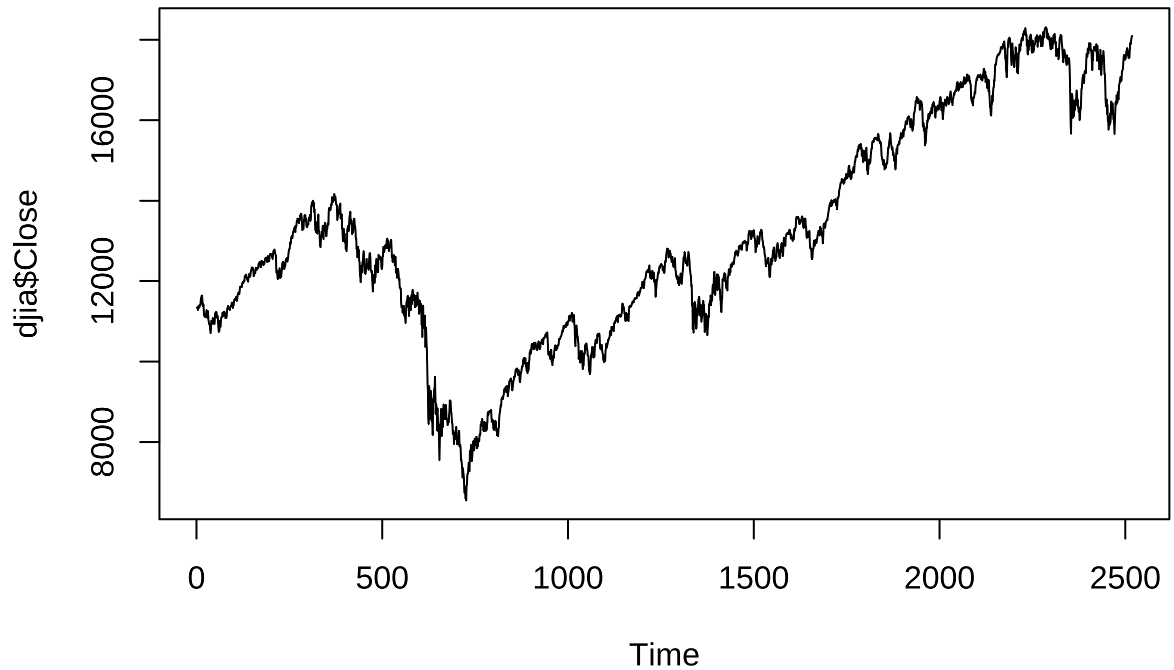
```r
# # # Ajustando un MA
#
# # Make a 1-step forecast based on MA
# predict_MA <- predict(MA, n.ahead = 1)
#
# # Obtain the 1-step forecast using $pred[1]
# predict_MA$pred[1]
#
# # Make a 1-step through 10-step forecast based on MA
# predict(MA, n.ahead = 10)
#
# # Plot the Nile series plus the forecast and 95% prediction intervals
# ts.plot(Nile, xlim = c(1871, 1980))
# MA_forecasts <- predict(MA, n.ahead = 10)$pred
# MA_forecast_se <- predict(MA, n.ahead = 10)$se
# points(MA_forecasts, type = "l", col = 2)
# points(MA_forecasts - 2*MA_forecast_se, type = "l", col = 2, lty = 2)
# points(MA_forecasts + 2*MA_forecast_se, type = "l", col = 2, lty = 2)
```

Pruebas series de tiempo de Time Series Analysis and its applications course

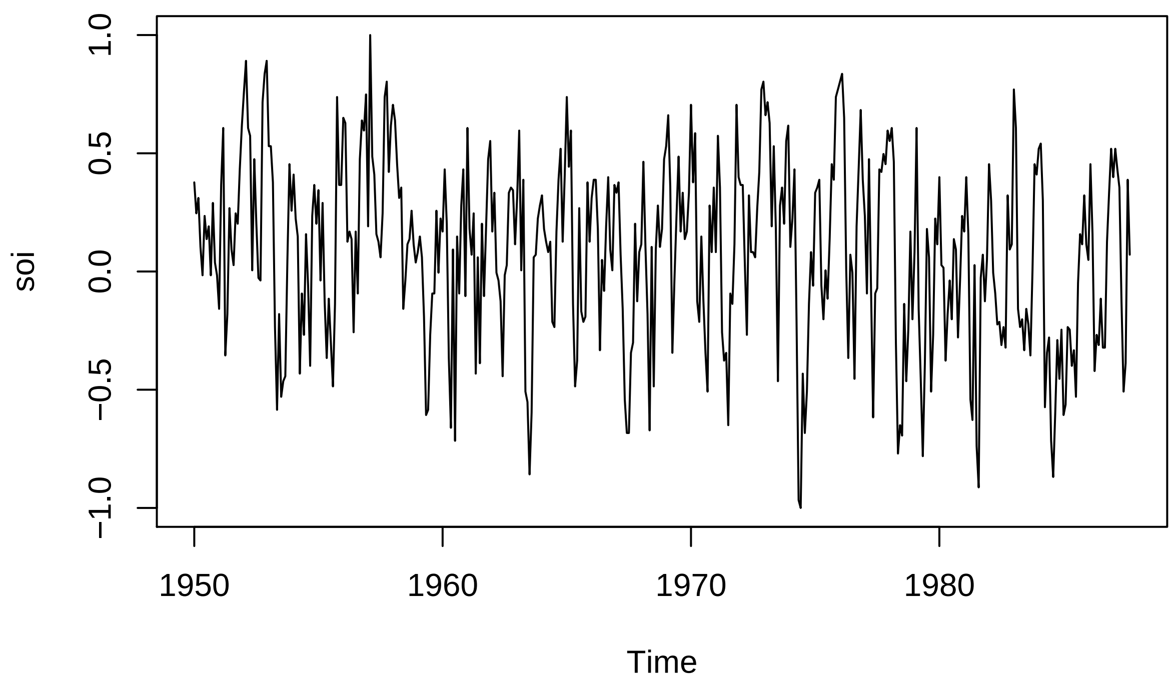
```r
# # R Arima base

# View a detailed description of AirPassengers
help(AirPassengers)

# Plot AirPassengers
ts.plot(AirPassengers)
```



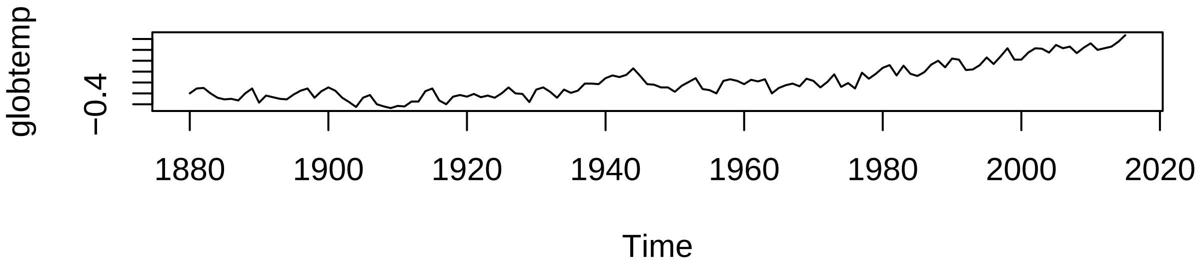

```r
# Plot the DJIA daily closings
ts.plot(djia$Close)
```

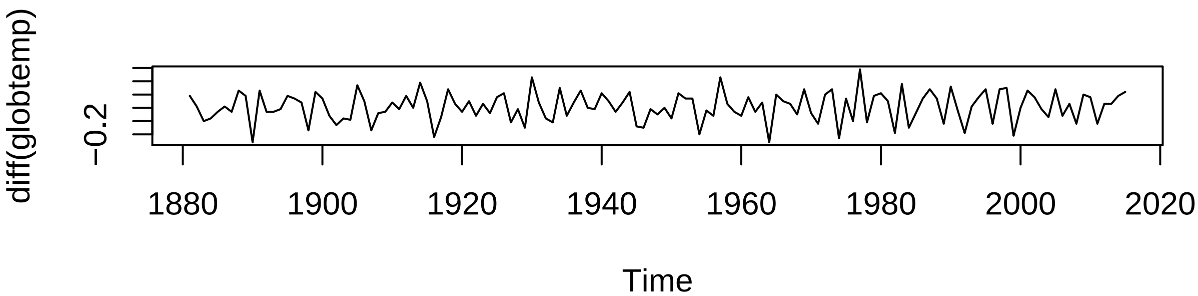



```r
# Plot the Southern Oscillation Index
plot(soi)
```

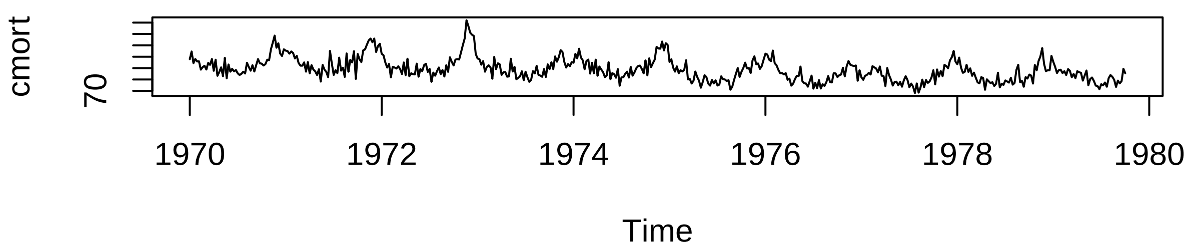


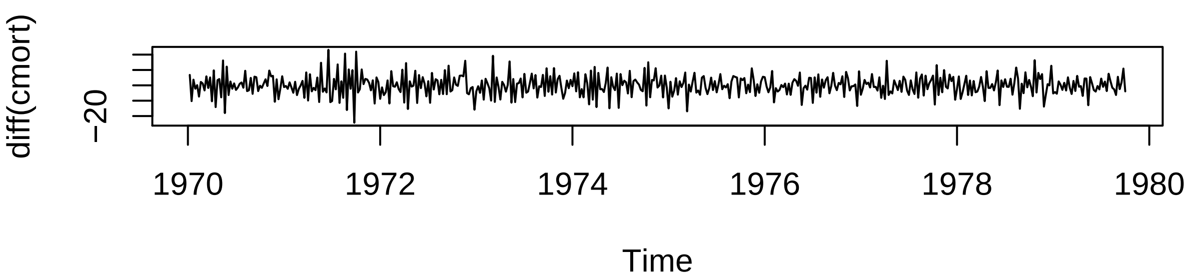

```r
# Plot globtemp and detrended globtemp
par(mfrow = c(2,1))
plot(globtemp)
plot(diff(globtemp))
```



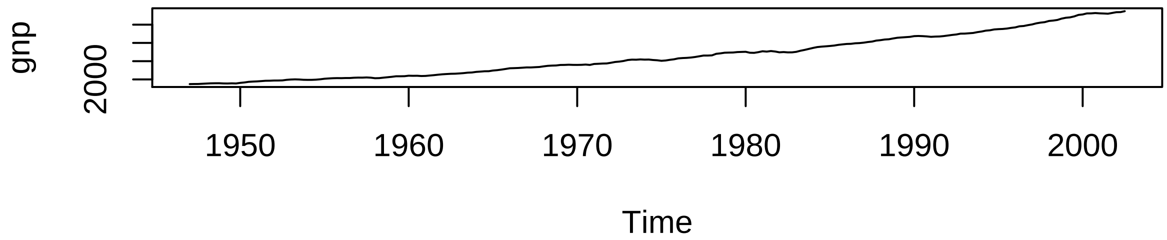


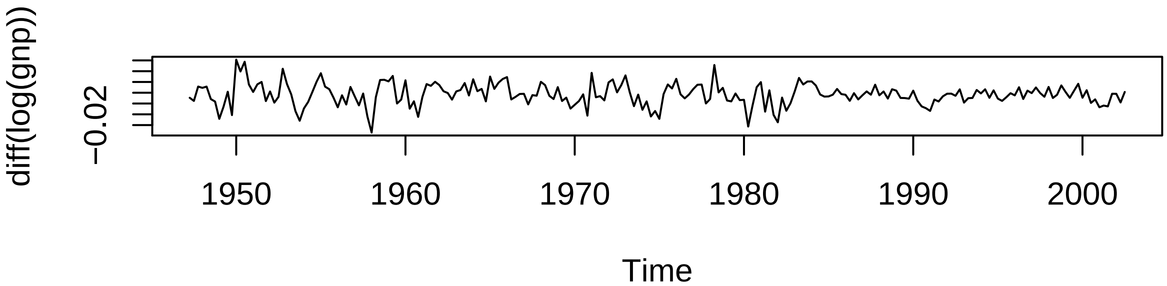

```
# Plot cmort and detrended cmort
par(mfrow = c(2,1))
plot(cmort)
plot(diff(cmort))
```



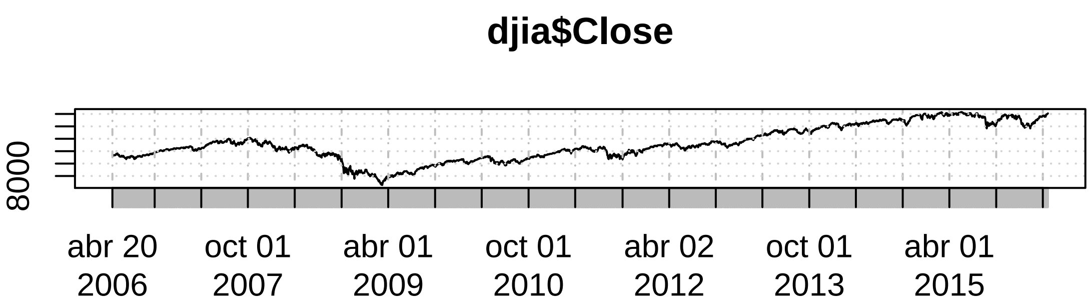


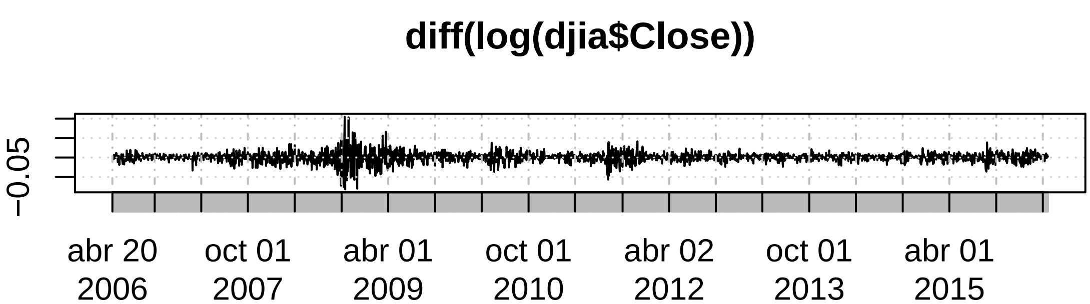

```
# Plot GNP series (gnp) and its growth rate
par(mfrow = c(2,1))
plot(gnp)
plot(diff(log(gnp)))
```





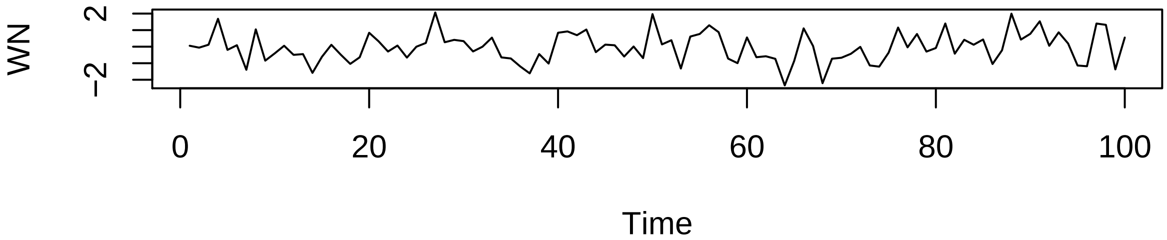

```r
# Plot DJIA closings (djia$Close) and its returns
par(mfrow = c(2,1))
plot(djia$Close)
plot(diff(log(djia$Close)))
```

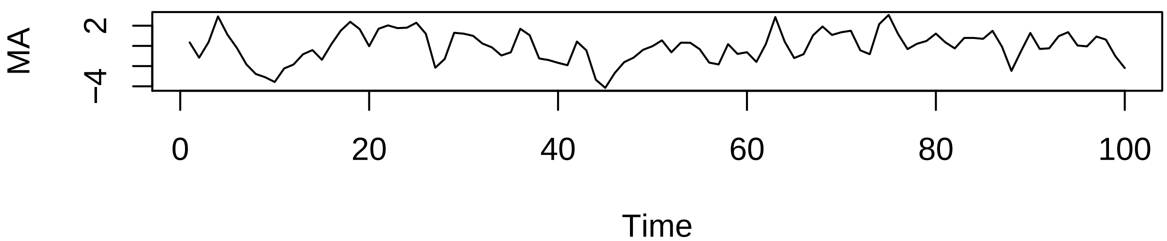




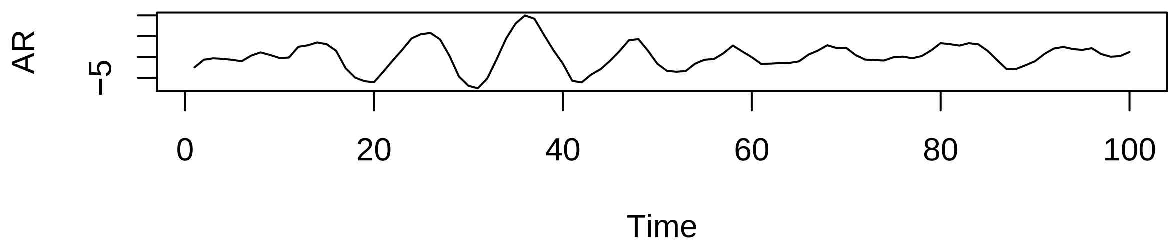

```r
##SIMULAR MA & AR

# Generate and plot white noise
WN <- arima.sim(model=list(order=c(0,0,0)),n=100)
plot(WN)
```

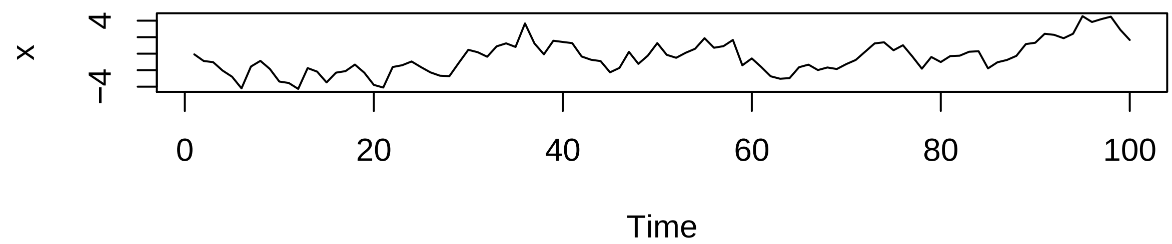
```r
# Generate and plot an MA(1) with parameter .9
MA <- arima.sim(model=list(order=c(0,0,1), ma = 0.9),n=100)
plot(MA)
```



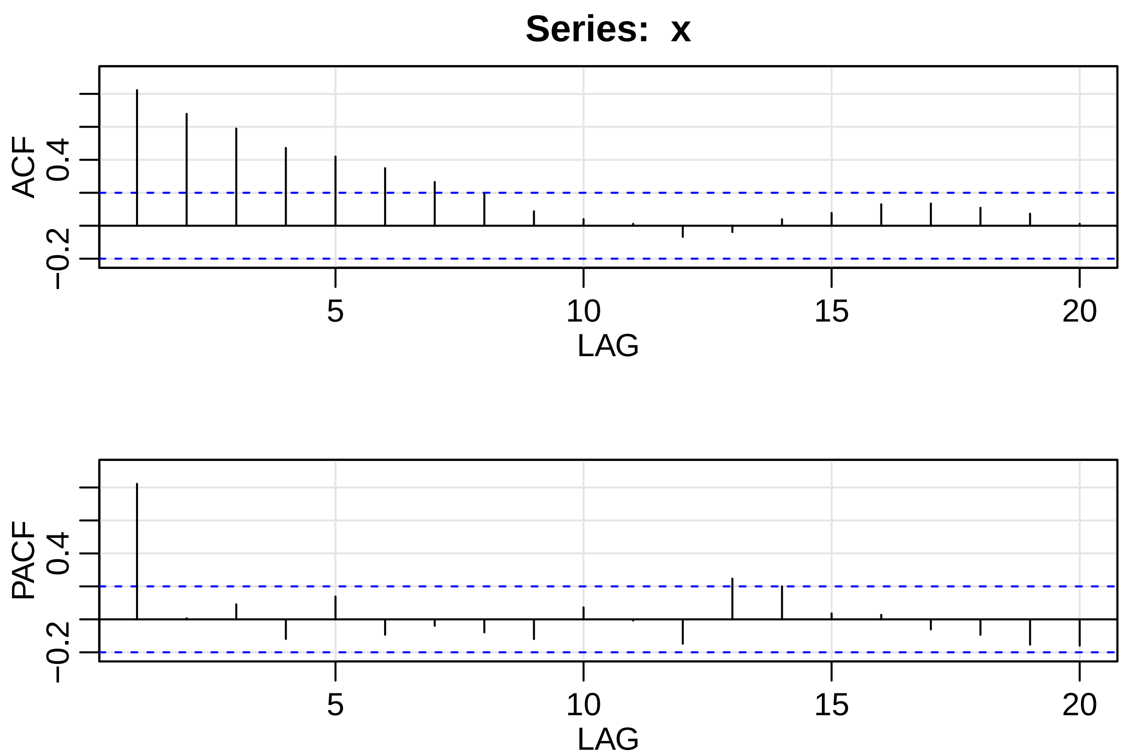



```r
# Generate and plot an AR(2) with parameters 1.5 and -.75
AR <- arima.sim(model=list(order=c(2,0,0), ar =c(1.5,-0.75)),n=100)
plot(AR)

# # SIMULAR Y AJUSTAR AR


# Generate 100 observations from the AR(1) model
x <- arima.sim(model = list(order = c(1, 0, 0), ar = .9), n = 100)

# Plot the generated data
plot(x)
```






```r
# Plot the sample P/ACF pair
acf2(x)
```

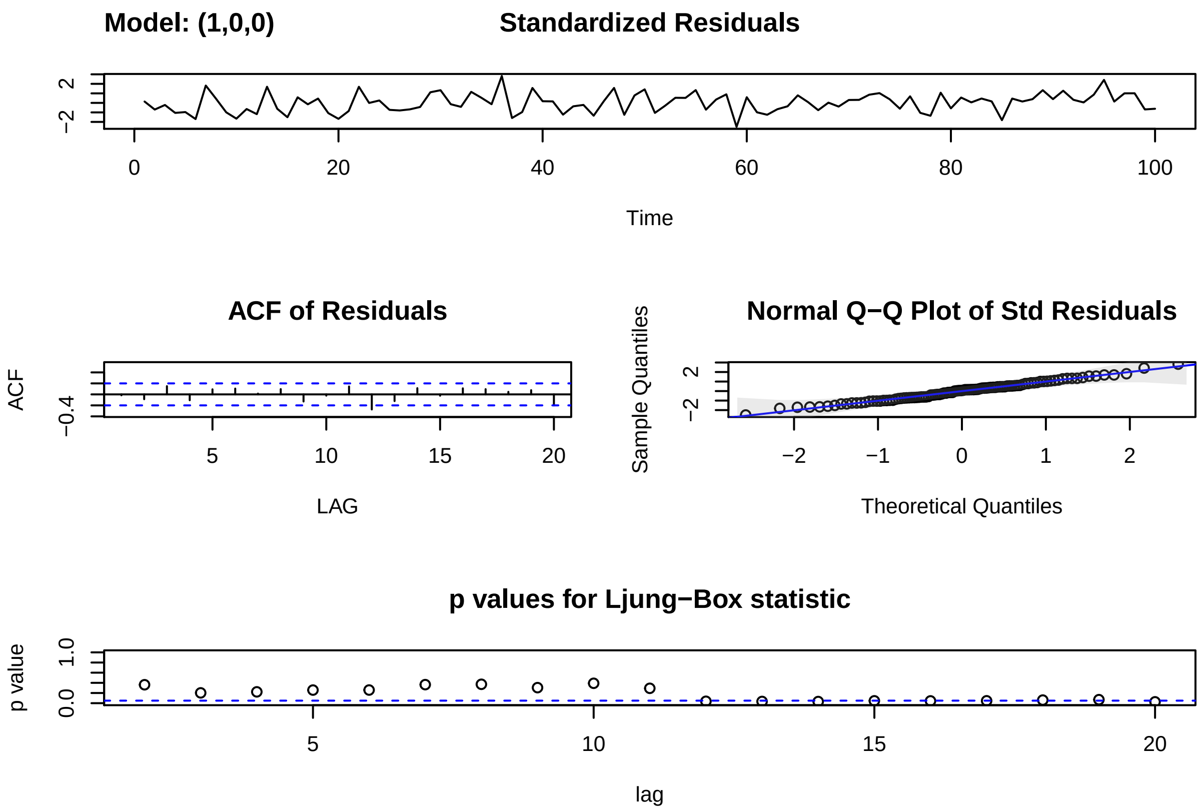



```
##        ACF  PACF
## [1,] 0.82  0.82
## [2,] 0.68  0.01
## [3,] 0.59  0.09
## [4,] 0.47 -0.12
## [5,] 0.42  0.14
```

```
##  [6,]  0.35 -0.09
##  [7,]  0.27 -0.04
##  [8,]  0.19 -0.08
##  [9,]  0.09 -0.12
## [10,]  0.04  0.07
## [11,]  0.01 -0.01
## [12,] -0.07 -0.15
## [13,] -0.04  0.25
## [14,]  0.04  0.20
## [15,]  0.08  0.04
## [16,]  0.13  0.03
## [17,]  0.14 -0.06
## [18,]  0.11 -0.09
## [19,]  0.07 -0.15
## [20,]  0.01 -0.16
```

```
# Fit an AR(1) to the data and examine the -table
sarima(x, 1, 0, 0)
```

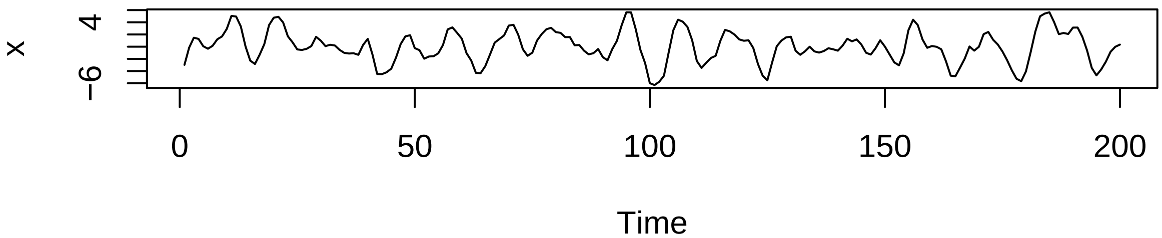
```
## initial  value 0.661628
## iter   2 value 0.084199
## iter   3 value 0.084066
## iter   4 value 0.084032
## iter   5 value 0.084014
## iter   6 value 0.083974
## iter   7 value 0.083969
## iter   8 value 0.083967
## iter   9 value 0.083965
## iter  10 value 0.083962
## iter  11 value 0.083962
## iter  12 value 0.083961
## iter  13 value 0.083961
## iter  14 value 0.083961
## iter  15 value 0.083961
## iter  15 value 0.083961
## iter  15 value 0.083961
## final  value 0.083961
## converged
## initial  value 0.084956
## iter   2 value 0.084898
## iter   3 value 0.084845
## iter   4 value 0.084844
## iter   5 value 0.084843
## iter   6 value 0.084842
## iter   7 value 0.084842
## iter   8 value 0.084842
## iter   9 value 0.084842
## iter  10 value 0.084842
## iter  10 value 0.084842
## iter  10 value 0.084842
## final  value 0.084842
## converged
```

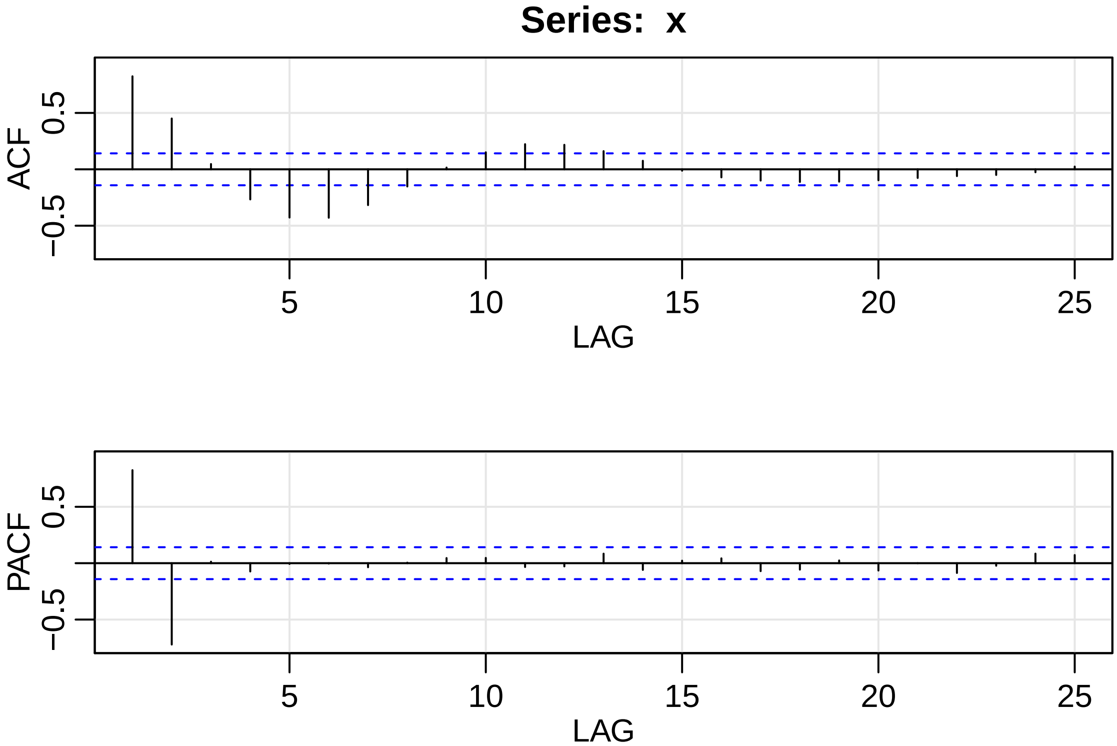



```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), xreg = xmean, include.mean = FALSE, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1    xmean
##       0.8247  -0.3500
## s.e.  0.0550   0.5914
##
## sigma^2 estimated as 1.172:  log likelihood = -150.38,  aic = 306.76
##
## $degrees_of_freedom
## [1] 98
##
## $ttable
##       Estimate     SE t.value p.value
## ar1     0.8247 0.0550 14.9926  0.0000
## xmean  -0.3500 0.5914 -0.5918  0.5554
##
## $AIC
## [1] 1.198286
##
## $AICc
## [1] 1.220786
##
```

```
## $BIC
## [1] 0.2503895
```

```r
# # aJUSTAR MA(2)

# astsa is preloaded
x <- arima.sim(model = list(order = c(2, 0, 0), ar = c(1.5, -.75)), n = 200)
# Plot x
plot(x)

# Plot the sample P/ACF of x
acf2(x)
```



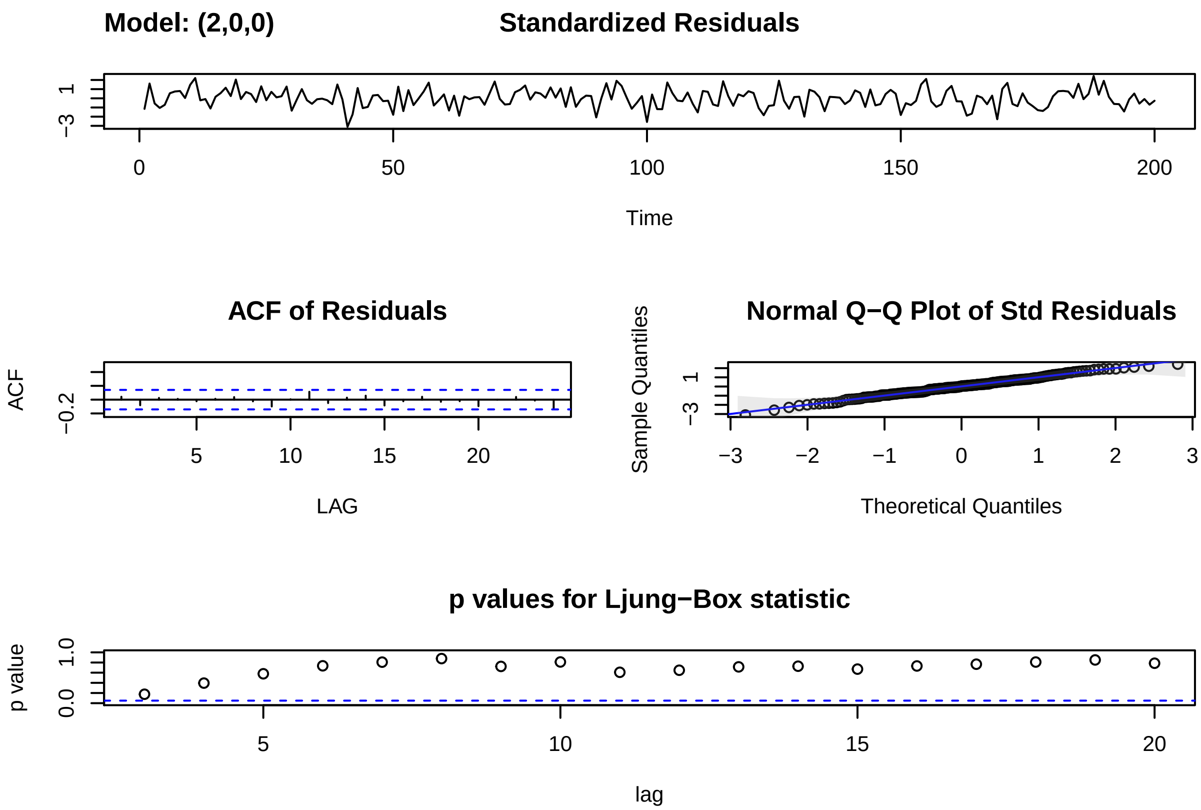



```
##         ACF  PACF
##  [1,]  0.83  0.83
##  [2,]  0.45 -0.72
##  [3,]  0.05  0.01
##  [4,] -0.27 -0.07
##  [5,] -0.43 -0.01
##  [6,] -0.43  0.00
##  [7,] -0.32 -0.04
##  [8,] -0.15  0.00
##  [9,]  0.02  0.05
## [10,]  0.15  0.05
```

```
## [11,]  0.22 -0.03
## [12,]  0.22 -0.03
## [13,]  0.16  0.08
## [14,]  0.08 -0.06
## [15,] -0.01  0.02
## [16,] -0.07  0.04
## [17,] -0.10 -0.07
## [18,] -0.11 -0.06
## [19,] -0.11  0.02
## [20,] -0.10 -0.07
## [21,] -0.08  0.00
## [22,] -0.06 -0.09
## [23,] -0.05 -0.02
## [24,] -0.03  0.08
## [25,]  0.02  0.07
```

```r
# Fit an AR(2) to the data and examine the t-table
sarima(x,2,0,0)
```

```
## initial  value 0.941414
## iter   2 value 0.785436
## iter   3 value 0.410362
## iter   4 value 0.254312
## iter   5 value -0.018321
## iter   6 value -0.033822
## iter   7 value -0.060554
## iter   8 value -0.062409
## iter   9 value -0.063509
## iter  10 value -0.064166
## iter  11 value -0.064209
## iter  12 value -0.064209
## iter  13 value -0.064210
## iter  14 value -0.064210
## iter  14 value -0.064210
## iter  14 value -0.064210
## final  value -0.064210
## converged
## initial  value -0.052052
## iter   2 value -0.052099
## iter   3 value -0.052124
## iter   4 value -0.052126
## iter   5 value -0.052126
## iter   6 value -0.052126
## iter   6 value -0.052126
## iter   6 value -0.052126
## final  value -0.052126
## converged
```

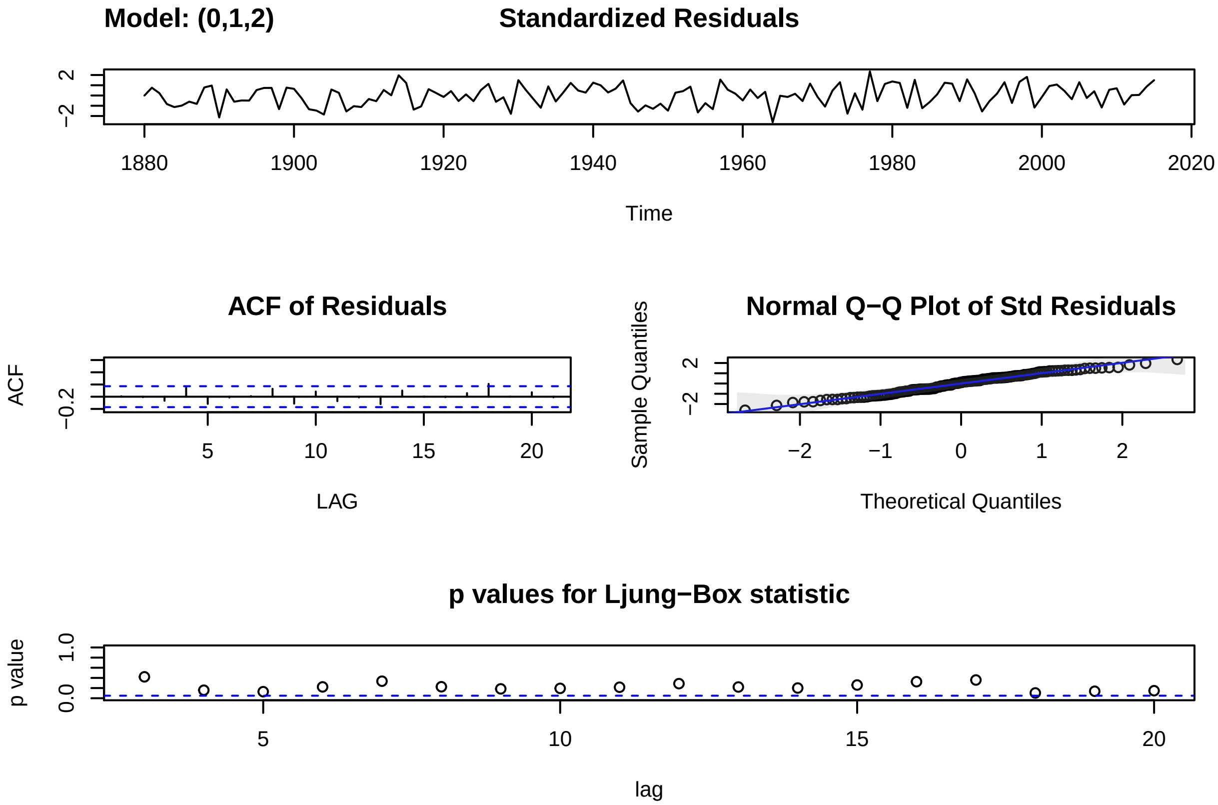



```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), xreg = xmean, include.mean = FALSE, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ar2    xmean
##       1.4562  -0.7540  -0.0229
## s.e.  0.0463   0.0461   0.2240
##
## sigma^2 estimated as 0.8883:  log likelihood = -273.36,  aic = 554.73
##
## $degrees_of_freedom
## [1] 197
##
## $ttable
##       Estimate     SE  t.value p.value
## ar1     1.4562 0.0463  31.4225  0.0000
## ar2    -0.7540 0.0461 -16.3540  0.0000
## xmean  -0.0229 0.2240  -0.1022  0.9187
##
## $AIC
## [1] 0.911498
##
## $AICc
## [1] 0.9225237
```

```
##
## $BIC
## [1] -0.03902719
```

```r
# # # SIMULANDO ARIMAS

# Fit an ARIMA(0,1,2) to globtemp and check the fit
sarima(globtemp, 0, 1, 2)
```

```
## initial  value -2.220513
## iter   2 value -2.294887
## iter   3 value -2.307682
## iter   4 value -2.309170
## iter   5 value -2.310360
## iter   6 value -2.311251
## iter   7 value -2.311636
## iter   8 value -2.311648
## iter   9 value -2.311649
## iter   9 value -2.311649
## iter   9 value -2.311649
## final  value -2.311649
## converged
## initial  value -2.310187
## iter   2 value -2.310197
## iter   3 value -2.310199
## iter   4 value -2.310201
## iter   5 value -2.310202
## iter   5 value -2.310202
## iter   5 value -2.310202
## final  value -2.310202
## converged
```



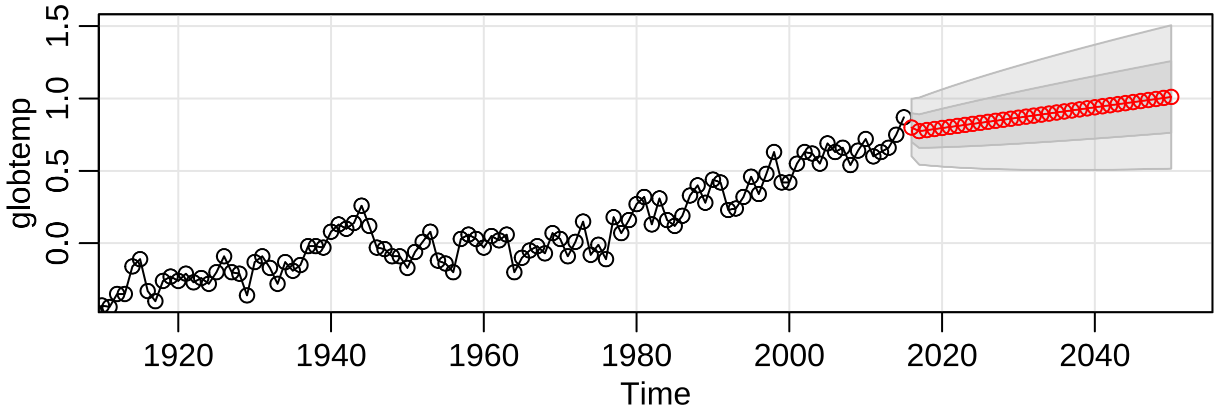

```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), xreg = constant, optim.control = list(trace = trc, REPORT = 1,
##     reltol = tol))
##
## Coefficients:
##           ma1      ma2  constant
##       -0.3984  -0.2173    0.0072
## s.e.   0.0808   0.0768    0.0033
##
## sigma^2 estimated as 0.00982:  log likelihood = 120.32,  aic = -232.64
##
## $degrees_of_freedom
## [1] 133
##
## $ttable
##          Estimate     SE t.value p.value
## ma1       -0.3984 0.0808 -4.9313  0.0000
## ma2       -0.2173 0.0768 -2.8303  0.0054
## constant   0.0072 0.0033  2.1463  0.0337
##
## $AIC
## [1] -3.579224
##
## $AICc
## [1] -3.562273
```

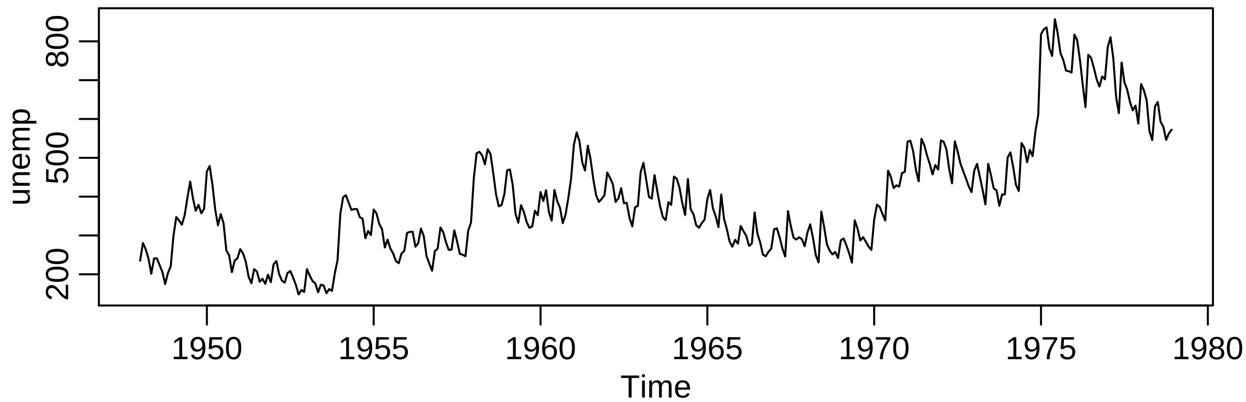
```
##
## $BIC
## [1] -4.514974
```

```r
# Forecast data 35 years into the future
sarima.for (globtemp,n.ahead = 35, p = 0, d = 1, q = 2)
```

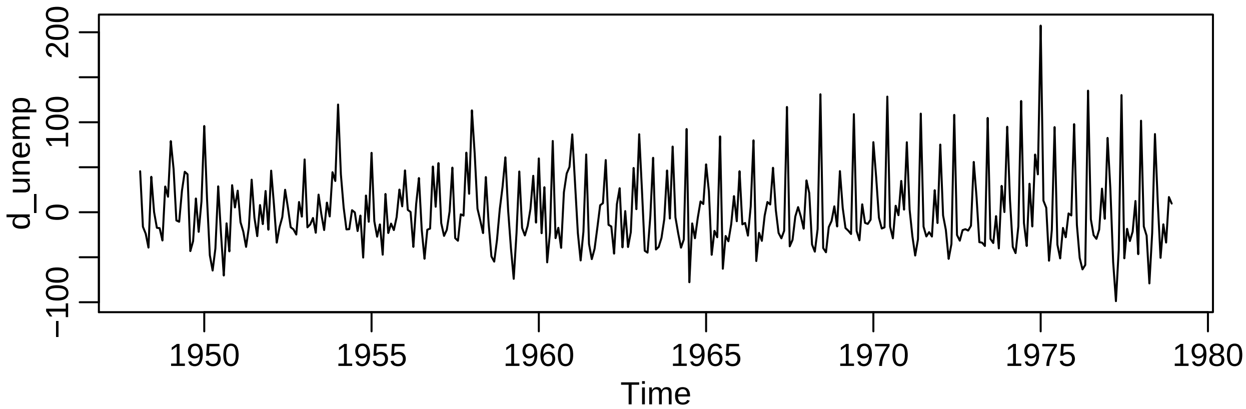
```
## $pred
## Time Series:
## Start = 2016
## End = 2050
## Frequency = 1
##  [1] 0.7995567 0.7745381 0.7816919 0.7888457 0.7959996 0.8031534 0.8103072
##  [8] 0.8174611 0.8246149 0.8317688 0.8389226 0.8460764 0.8532303 0.8603841
## [15] 0.8675379 0.8746918 0.8818456 0.8889995 0.8961533 0.9033071 0.9104610
## [22] 0.9176148 0.9247687 0.9319225 0.9390763 0.9462302 0.9533840 0.9605378
## [29] 0.9676917 0.9748455 0.9819994 0.9891532 0.9963070 1.0034609 1.0106147
##
## $se
## Time Series:
## Start = 2016
## End = 2050
## Frequency = 1
##  [1] 0.09909556 0.11564576 0.12175580 0.12757353 0.13313729 0.13847769
##  [7] 0.14361964 0.14858376 0.15338730 0.15804492 0.16256915 0.16697084
## [13] 0.17125943 0.17544322 0.17952954 0.18352490 0.18743511 0.19126540
## [19] 0.19502047 0.19870459 0.20232164 0.20587515 0.20936836 0.21280424
## [25] 0.21618551 0.21951471 0.22279416 0.22602604 0.22921235 0.23235497
## [31] 0.23545565 0.23851603 0.24153763 0.24452190 0.24747019
```

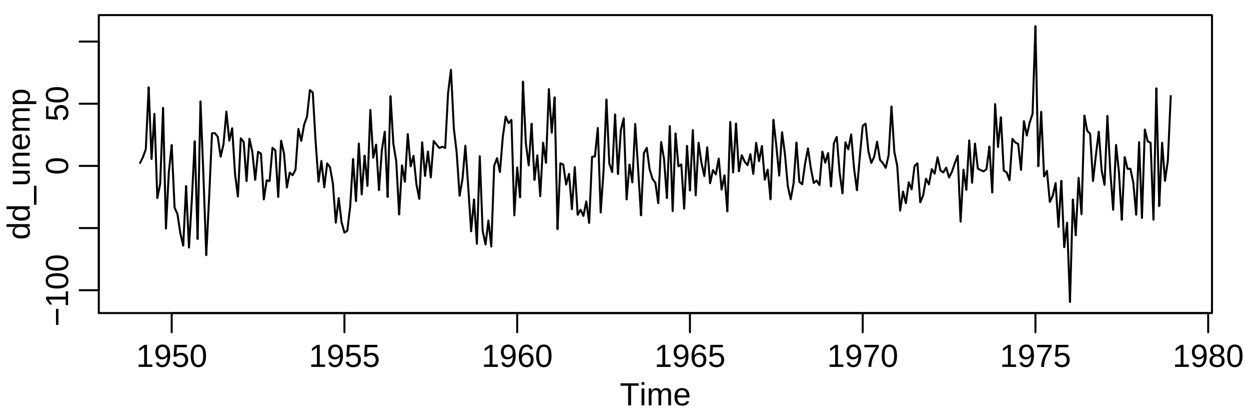
```r
### SERIES DE TIEMPO ESTACIONALES

# Plot unemp
plot(unemp)
```



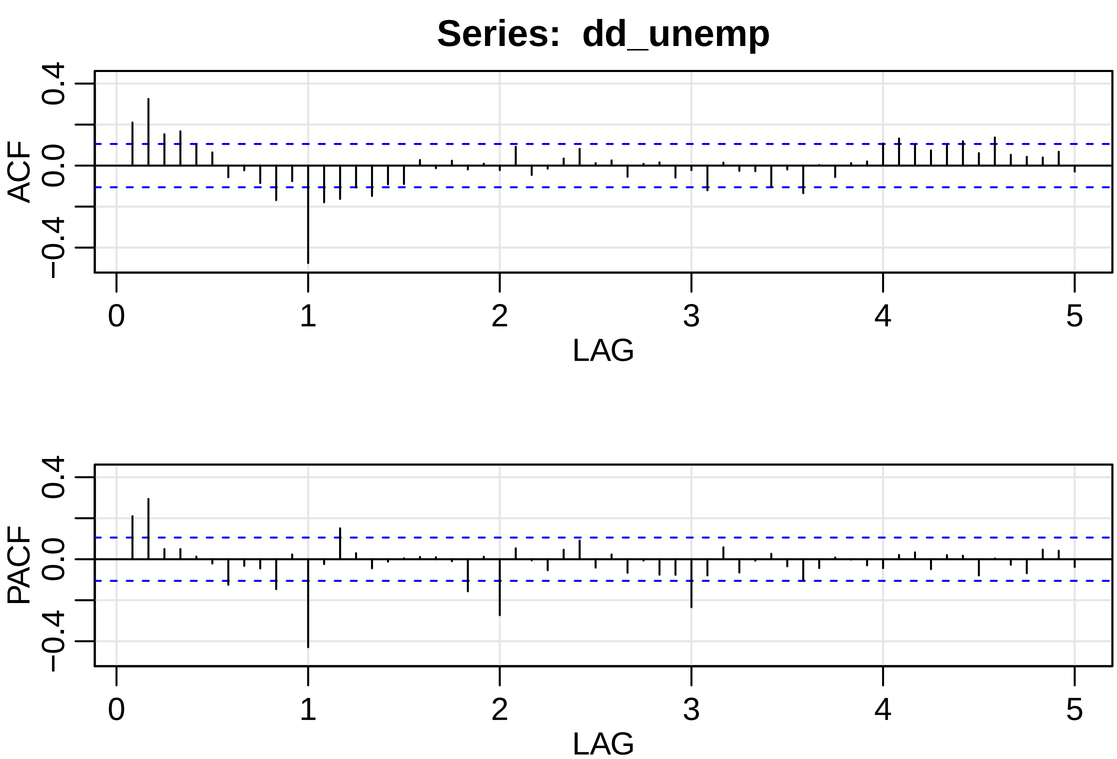



```
# Difference your data and plot it
d_unemp <- diff(unemp)
plot(d_unemp)

# Seasonally difference d_unemp and plot it
dd_unemp <- diff(d_unemp, lag = 12)
plot(dd_unemp)
```






```
# Plot P/ACF pair of fully differenced data to lag 60
dd_unemp <- diff(diff(unemp), lag = 12)
acf2(dd_unemp, max.lag = 60)
```

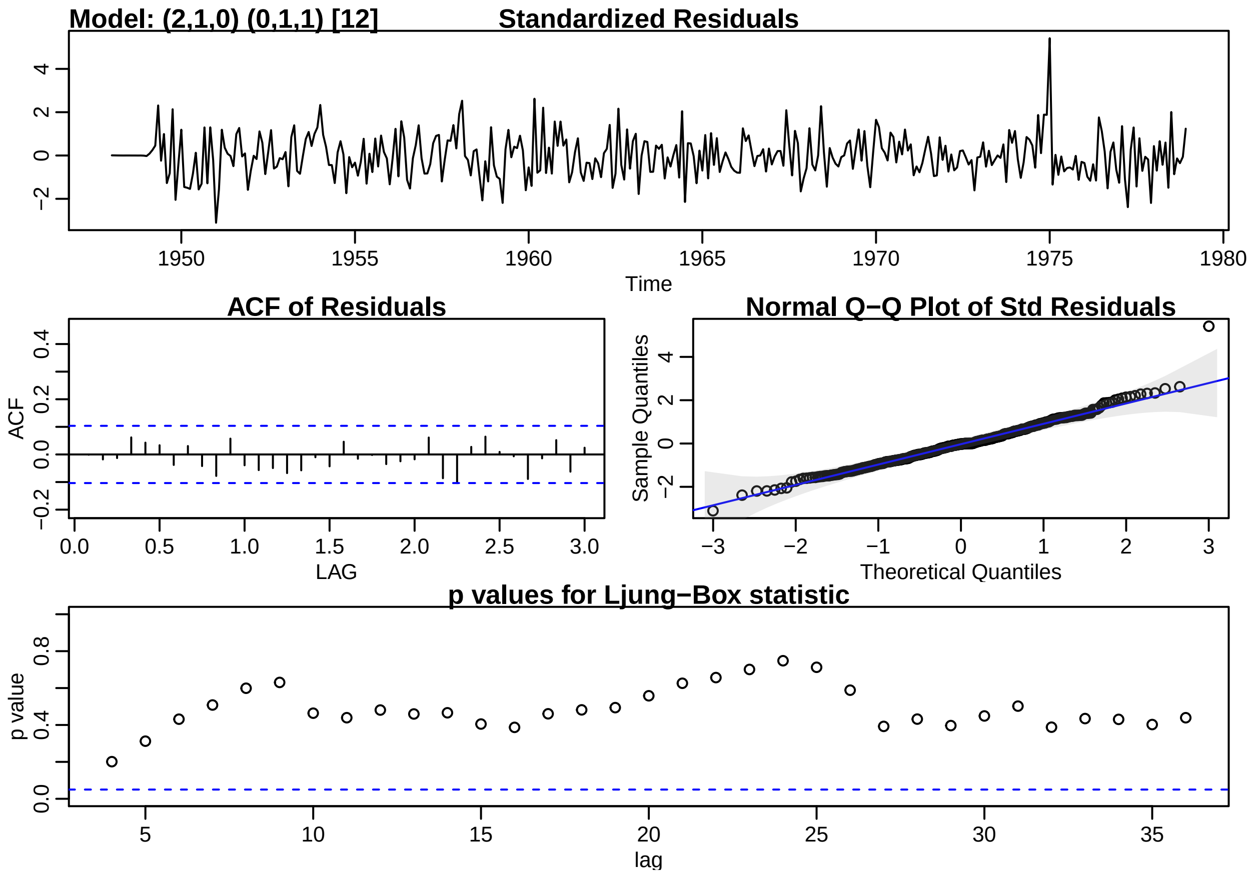


```
##         ACF  PACF
##  [1,]  0.21  0.21
##  [2,]  0.33  0.29
##  [3,]  0.15  0.05
##  [4,]  0.17  0.05
##  [5,]  0.10  0.01
##  [6,]  0.06 -0.02
##  [7,] -0.06 -0.12
##  [8,] -0.02 -0.03
##  [9,] -0.09 -0.05
## [10,] -0.17 -0.15
## [11,] -0.08  0.02
## [12,] -0.48 -0.43
## [13,] -0.18 -0.02
## [14,] -0.16  0.15
## [15,] -0.11  0.03
## [16,] -0.15 -0.04
## [17,] -0.09 -0.01
## [18,] -0.09  0.00
## [19,]  0.03  0.01
## [20,] -0.01  0.01
## [21,]  0.02 -0.01
## [22,] -0.02 -0.16
## [23,]  0.01  0.01
## [24,] -0.02 -0.27
## [25,]  0.09  0.05
## [26,] -0.05 -0.01
## [27,] -0.01 -0.05
## [28,]  0.03  0.05
## [29,]  0.08  0.09
## [30,]  0.01 -0.04
## [31,]  0.03  0.02
## [32,] -0.05 -0.07
## [33,]  0.01 -0.01
## [34,]  0.02 -0.08
## [35,] -0.06 -0.08
## [36,] -0.02 -0.23
## [37,] -0.12 -0.08
## [38,]  0.01  0.06
## [39,] -0.03 -0.07
## [40,] -0.03 -0.01
## [41,] -0.10  0.03
## [42,] -0.02 -0.03
## [43,] -0.13 -0.11
## [44,]  0.00 -0.04
## [45,] -0.06  0.01
## [46,]  0.01  0.00
## [47,]  0.02 -0.03
## [48,]  0.11 -0.04
## [49,]  0.13  0.02
## [50,]  0.10  0.03
## [51,]  0.07 -0.05
## [52,]  0.10  0.02
## [53,]  0.12  0.02
```

```
## [54,]  0.06 -0.08
## [55,]  0.14  0.00
## [56,]  0.05 -0.03
## [57,]  0.04 -0.07
## [58,]  0.04  0.05
## [59,]  0.07  0.04
## [60,] -0.03 -0.04
```

```r
# Fit an appropriate model
sarima(unemp,2,1,0,0,1,1, S=12)
```

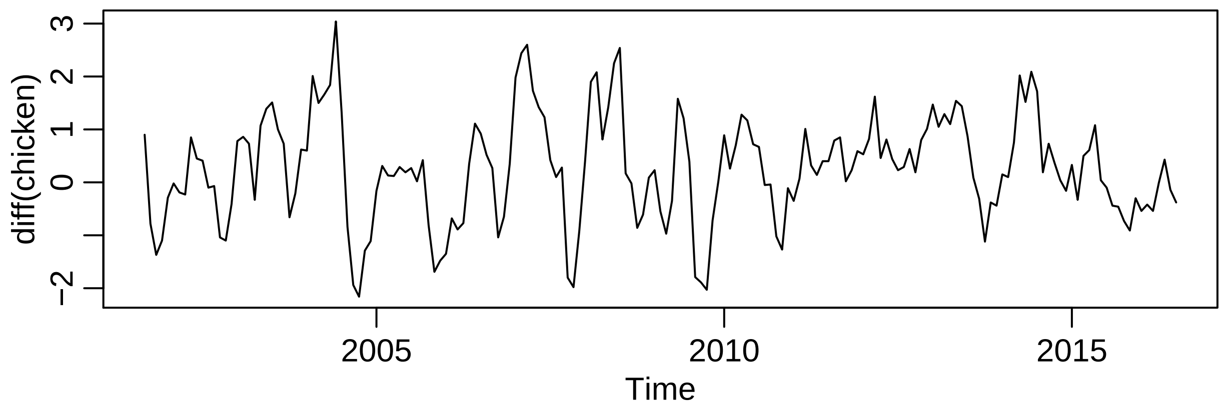
```
## initial  value 3.340809
## iter   2 value 3.105512
## iter   3 value 3.086631
## iter   4 value 3.079778
## iter   5 value 3.069447
## iter   6 value 3.067659
## iter   7 value 3.067426
## iter   8 value 3.067418
## iter   8 value 3.067418
## final  value 3.067418
## converged
## initial  value 3.065481
## iter   2 value 3.065478
## iter   3 value 3.065477
## iter   3 value 3.065477
## iter   3 value 3.065477
## final  value 3.065477
## converged
```

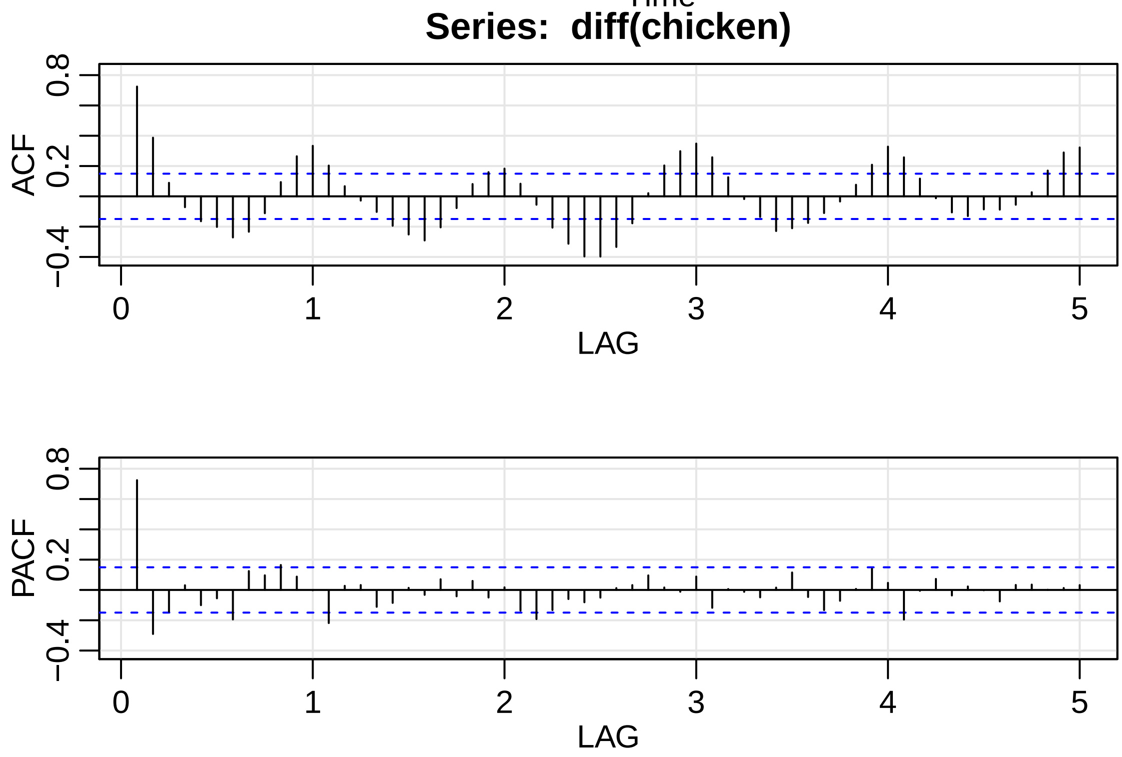



```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1     ar2     sma1
##       0.1351  0.2464  -0.6953
## s.e.  0.0513  0.0515   0.0381
##
## sigma^2 estimated as 449.6:  log likelihood = -1609.91,  aic = 3227.81
##
## $degrees_of_freedom
## [1] 369
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1    0.1351 0.0513   2.6326  0.0088
## ar2    0.2464 0.0515   4.7795  0.0000
## sma1  -0.6953 0.0381 -18.2362  0.0000
##
## $AIC
## [1] 7.12457
##
```

```
## $AICc
## [1] 7.130239
##
## $BIC
## [1] 6.156174
```

```r
# Plot differenced chicken
plot(diff(chicken))

# Plot P/ACF pair of differenced data to lag 60
acf2(diff(chicken), max.lag = 60)
```



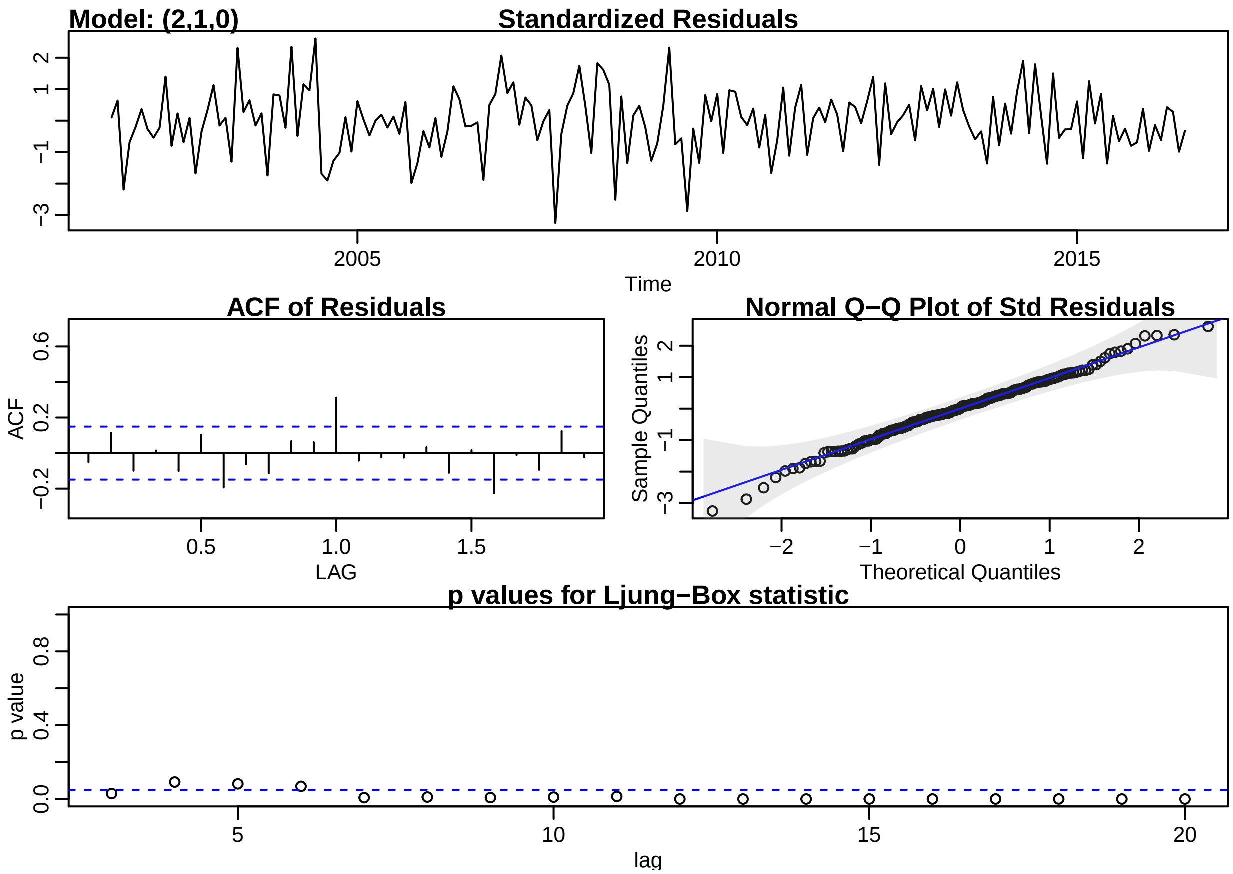



```
##          ACF  PACF
##   [1,]  0.72  0.72
##   [2,]  0.39 -0.29
##   [3,]  0.09 -0.14
##   [4,] -0.07  0.03
##   [5,] -0.16 -0.10
##   [6,] -0.20 -0.06
```

```
##  [7,] -0.27 -0.19
##  [8,] -0.23  0.12
##  [9,] -0.11  0.10
## [10,]  0.09  0.16
## [11,]  0.26  0.09
## [12,]  0.33  0.00
## [13,]  0.20 -0.22
## [14,]  0.07  0.03
## [15,] -0.03  0.03
## [16,] -0.10 -0.11
## [17,] -0.19 -0.09
## [18,] -0.25  0.01
## [19,] -0.29 -0.03
## [20,] -0.20  0.07
## [21,] -0.08 -0.04
## [22,]  0.08  0.06
## [23,]  0.16 -0.05
## [24,]  0.18  0.02
## [25,]  0.08 -0.14
## [26,] -0.06 -0.19
## [27,] -0.21 -0.13
## [28,] -0.31 -0.06
## [29,] -0.40 -0.08
## [30,] -0.40 -0.05
## [31,] -0.33  0.01
## [32,] -0.18  0.03
## [33,]  0.02  0.10
## [34,]  0.20  0.02
## [35,]  0.30 -0.01
## [36,]  0.35  0.09
## [37,]  0.26 -0.12
## [38,]  0.13  0.01
## [39,] -0.02 -0.01
## [40,] -0.14 -0.05
## [41,] -0.23  0.02
## [42,] -0.21  0.12
## [43,] -0.18 -0.05
## [44,] -0.11 -0.13
## [45,] -0.03 -0.07
## [46,]  0.08  0.01
## [47,]  0.21  0.14
## [48,]  0.33  0.05
## [49,]  0.26 -0.20
## [50,]  0.12 -0.01
## [51,] -0.01  0.07
## [52,] -0.11 -0.04
## [53,] -0.13  0.02
## [54,] -0.09  0.00
## [55,] -0.09 -0.08
## [56,] -0.06  0.03
## [57,]  0.03  0.04
## [58,]  0.17  0.00
## [59,]  0.29  0.01
## [60,]  0.32  0.03
```

```r
# Fit ARIMA(2,1,0) to chicken - not so good
sarima(chicken, 2,1,0)
```

```
## initial  value 0.001863
## iter   2 value -0.156034
## iter   3 value -0.359181
## iter   4 value -0.424164
## iter   5 value -0.430212
## iter   6 value -0.432744
## iter   7 value -0.432747
## iter   8 value -0.432749
## iter   9 value -0.432749
## iter  10 value -0.432751
## iter  11 value -0.432752
## iter  12 value -0.432752
## iter  13 value -0.432752
## iter  13 value -0.432752
## iter  13 value -0.432752
## final  value -0.432752
## converged
## initial  value -0.420883
## iter   2 value -0.420934
## iter   3 value -0.420936
## iter   4 value -0.420937
## iter   5 value -0.420937
## iter   6 value -0.420937
## iter   6 value -0.420937
## iter   6 value -0.420937
## final  value -0.420937
## converged
```

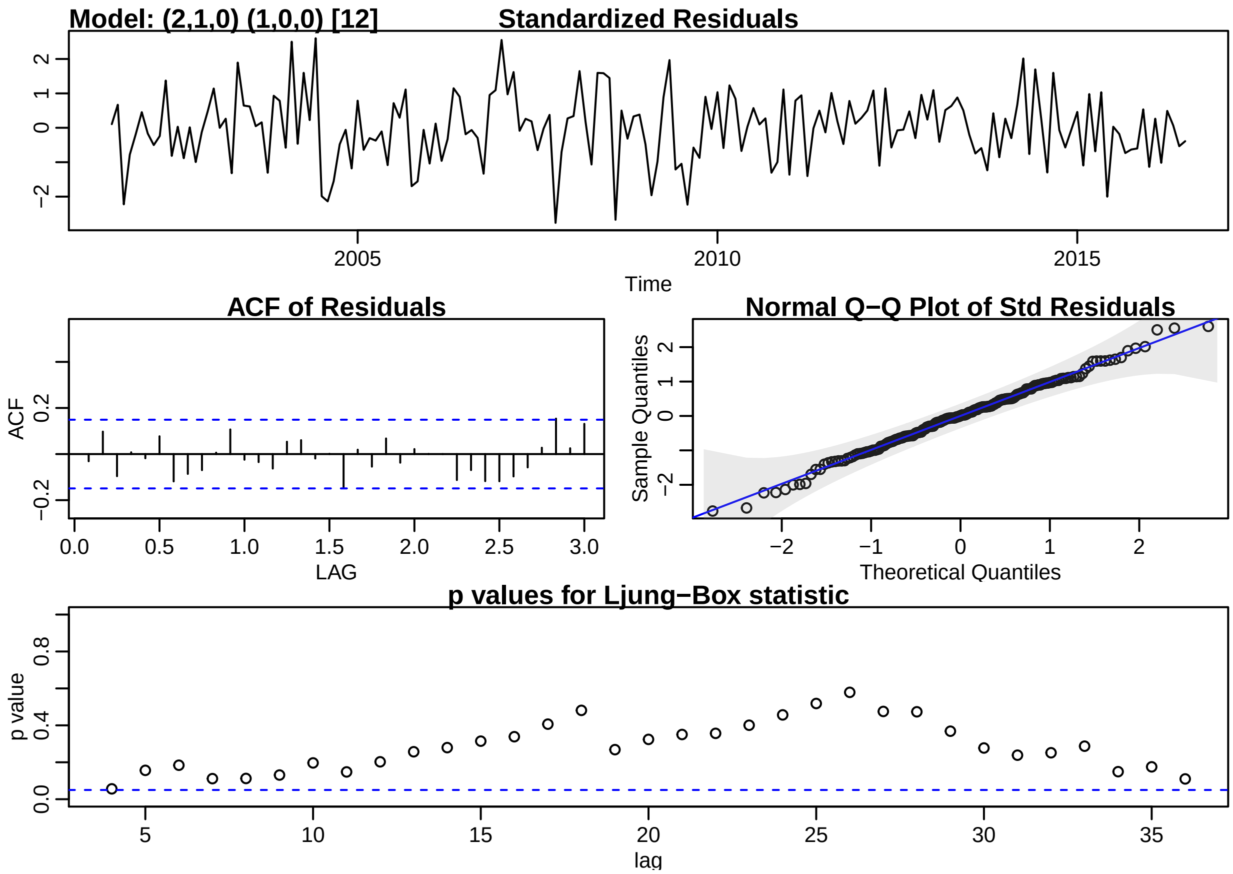



```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), xreg = constant, optim.control = list(trace = trc, REPORT = 1,
##     reltol = tol))
##
## Coefficients:
##          ar1      ar2  constant
##       0.9494  -0.3069    0.2632
## s.e.  0.0717   0.0718    0.1362
##
## sigma^2 estimated as 0.4286:  log likelihood = -178.64,  aic = 365.28
##
## $degrees_of_freedom
## [1] 177
##
## $ttable
##          Estimate     SE t.value p.value
## ar1        0.9494 0.0717 13.2339  0.0000
## ar2       -0.3069 0.0718 -4.2723  0.0000
## constant   0.2632 0.1362  1.9328  0.0549
##
## $AIC
## [1] 0.1861622
##
```

```
## $AICc
## [1] 0.1985432
##
## $BIC
## [1] -0.7606218
```

```r
# Fit SARIMA(2,1,0,1,0,0,12) to chicken - that works
sarima(chicken,2,1,0,1,0,0,12)
```

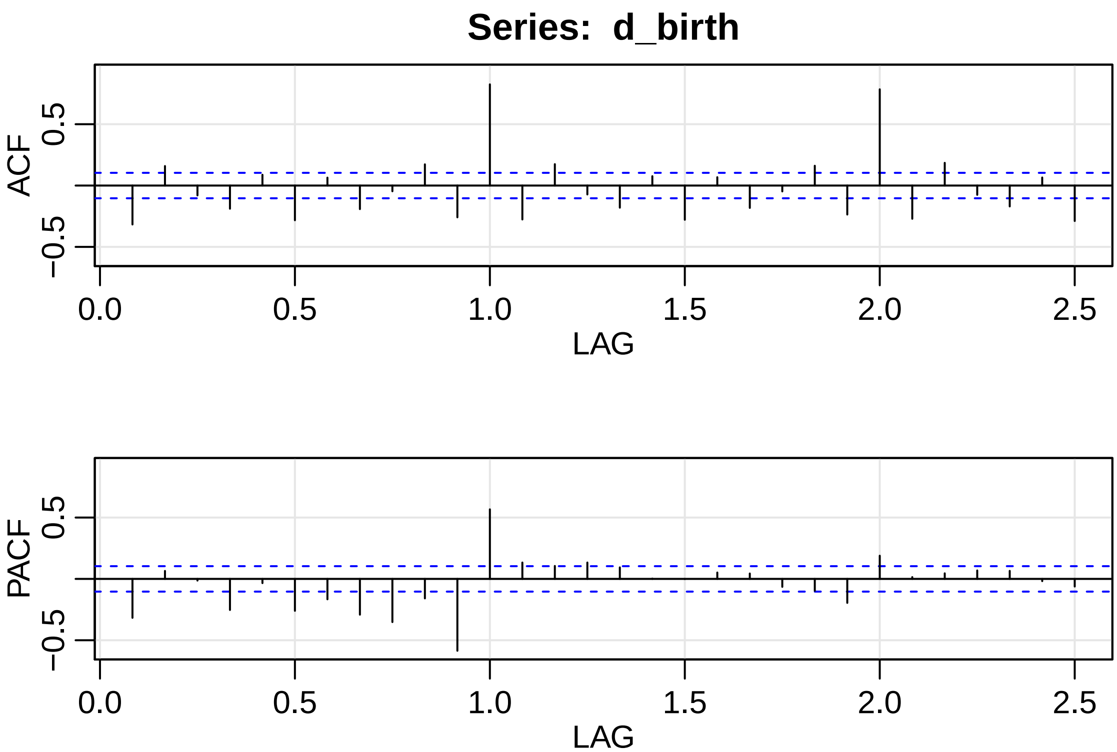
```
## initial  value 0.015039
## iter   2 value -0.226398
## iter   3 value -0.412955
## iter   4 value -0.460882
## iter   5 value -0.470787
## iter   6 value -0.471082
## iter   7 value -0.471088
## iter   8 value -0.471090
## iter   9 value -0.471092
## iter  10 value -0.471095
## iter  11 value -0.471095
## iter  12 value -0.471096
## iter  13 value -0.471096
## iter  14 value -0.471096
## iter  15 value -0.471097
## iter  16 value -0.471097
## iter  16 value -0.471097
## iter  16 value -0.471097
## final  value -0.471097
## converged
## initial  value -0.473585
## iter   2 value -0.473664
## iter   3 value -0.473721
## iter   4 value -0.473823
## iter   5 value -0.473871
## iter   6 value -0.473885
## iter   7 value -0.473886
## iter   8 value -0.473886
## iter   8 value -0.473886
## iter   8 value -0.473886
## final  value -0.473886
## converged
```




```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), xreg = constant, optim.control = list(trace = trc, REPORT = 1,
##     reltol = tol))
##
## Coefficients:
##          ar1      ar2    sar1  constant
##       0.9154  -0.2494  0.3237    0.2353
## s.e.  0.0733   0.0739  0.0715    0.1973
##
## sigma^2 estimated as 0.3828:  log likelihood = -169.16,  aic = 348.33
##
## $degrees_of_freedom
## [1] 176
##
## $ttable
##          Estimate     SE t.value p.value
## ar1        0.9154 0.0733 12.4955  0.0000
## ar2       -0.2494 0.0739 -3.3728  0.0009
## sar1       0.3237 0.0715  4.5238  0.0000
## constant   0.2353 0.1973  1.1923  0.2347
##
## $AIC
## [1] 0.0842377
```

```
##
## $AICc
## [1] 0.09726452
##
## $BIC
## [1] -0.8448077
```

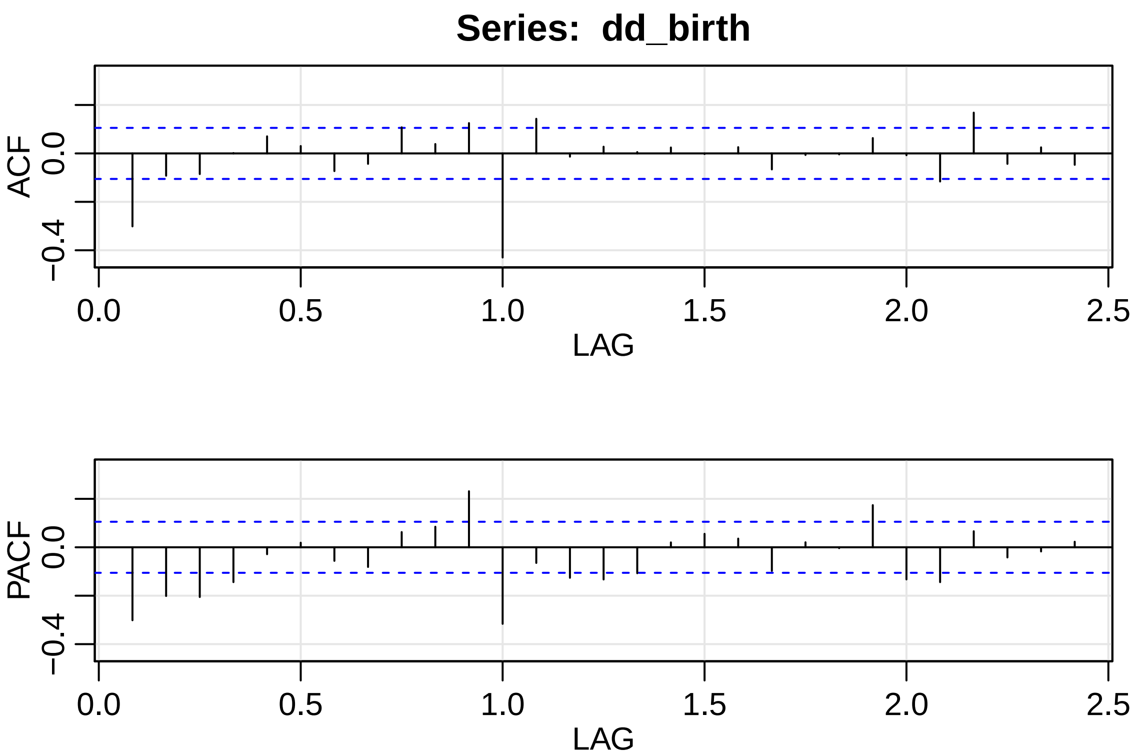
```
###### Natalidad US

# Plot P/ACF to lag 60 of differenced data
d_birth <- diff(birth)
acf2(d_birth)
```




```
##          ACF  PACF
##  [1,] -0.32 -0.32
##  [2,]  0.16  0.06
##  [3,] -0.08 -0.01
##  [4,] -0.19 -0.25
##  [5,]  0.09 -0.03
##  [6,] -0.28 -0.26
##  [7,]  0.06 -0.17
##  [8,] -0.19 -0.29
##  [9,] -0.05 -0.35
## [10,]  0.17 -0.16
## [11,] -0.26 -0.59
## [12,]  0.82  0.57
## [13,] -0.28  0.13
## [14,]  0.17  0.11
## [15,] -0.07  0.13
## [16,] -0.18  0.09
## [17,]  0.08  0.00
```

```
## [18,] -0.28  0.00
## [19,]  0.07  0.05
## [20,] -0.18  0.04
## [21,] -0.05 -0.07
## [22,]  0.16 -0.10
## [23,] -0.24 -0.20
## [24,]  0.78  0.19
## [25,] -0.27  0.01
## [26,]  0.19  0.05
## [27,] -0.08  0.07
## [28,] -0.17  0.07
## [29,]  0.07 -0.02
## [30,] -0.29 -0.06
```

```
# Plot P/ACF to lag 60 of seasonal differenced data
dd_birth <- diff(d_birth, lag = 12)
acf2(dd_birth)
```

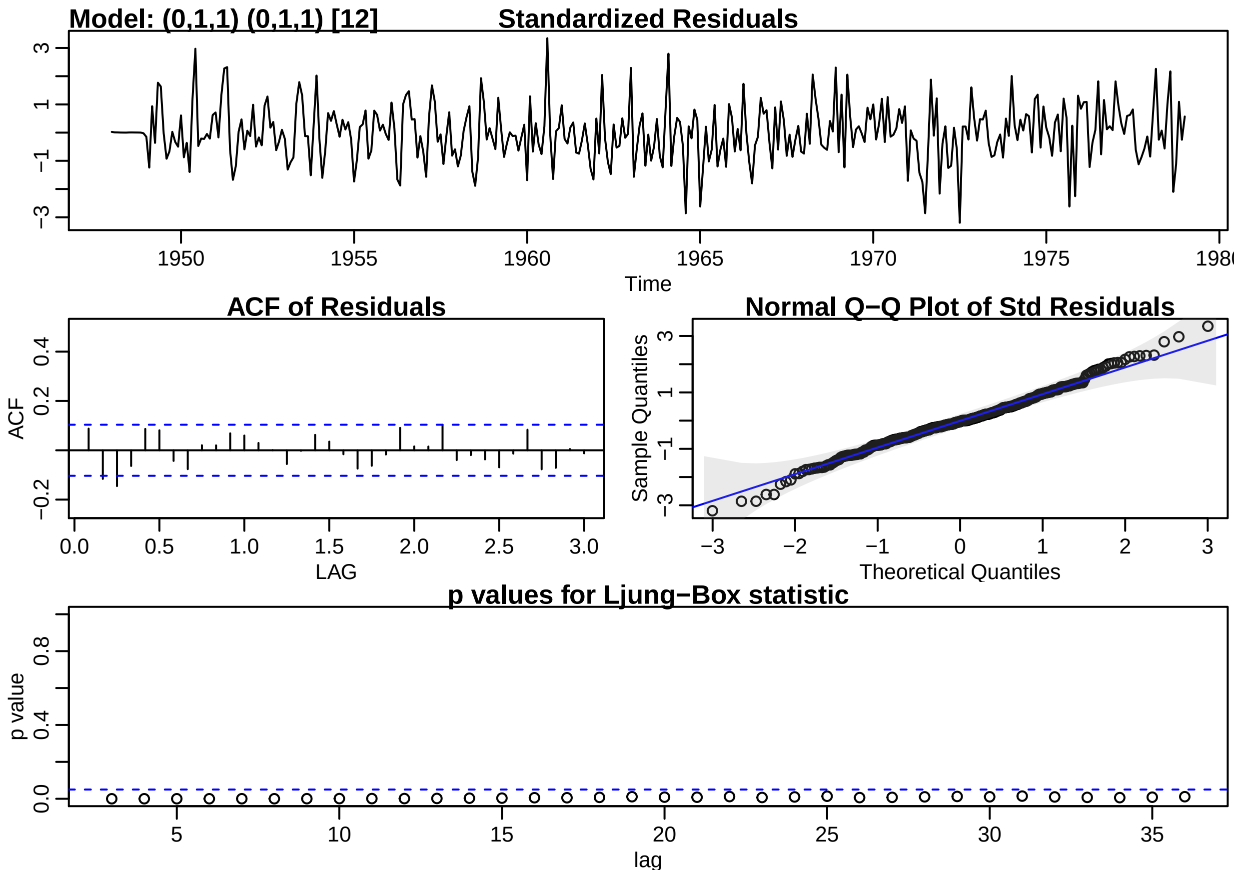



```
##         ACF  PACF
##  [1,] -0.30 -0.30
##  [2,] -0.09 -0.20
##  [3,] -0.09 -0.21
##  [4,]  0.00 -0.14
##  [5,]  0.07 -0.03
##  [6,]  0.03  0.02
##  [7,] -0.07 -0.06
##  [8,] -0.04 -0.08
##  [9,]  0.11  0.06
## [10,]  0.04  0.08
## [11,]  0.13  0.23
## [12,] -0.43 -0.32
```

```
## [13,]  0.14 -0.06
## [14,] -0.01 -0.13
## [15,]  0.03 -0.13
## [16,]  0.01 -0.11
## [17,]  0.02  0.02
## [18,]  0.00  0.06
## [19,]  0.03  0.04
## [20,] -0.07 -0.10
## [21,] -0.01  0.02
## [22,]  0.00  0.00
## [23,]  0.06  0.17
## [24,] -0.01 -0.13
## [25,] -0.12 -0.14
## [26,]  0.17  0.07
## [27,] -0.04 -0.04
## [28,]  0.03 -0.02
## [29,] -0.05  0.02
```

```r
# Fit SARIMA(0,1,1)x(0,1,1)_12. What happens?
sarima(birth,0,1,1,0,1,1,12)
```

```
## initial  value 2.219164
## iter   2 value 2.013310
## iter   3 value 1.988107
## iter   4 value 1.980026
## iter   5 value 1.967594
## iter   6 value 1.965384
## iter   7 value 1.965049
## iter   8 value 1.964993
## iter   9 value 1.964992
## iter   9 value 1.964992
## iter   9 value 1.964992
## final  value 1.964992
## converged
## initial  value 1.951264
## iter   2 value 1.945867
## iter   3 value 1.945729
## iter   4 value 1.945723
## iter   5 value 1.945723
## iter   5 value 1.945723
## iter   5 value 1.945723
## final  value 1.945723
## converged
```

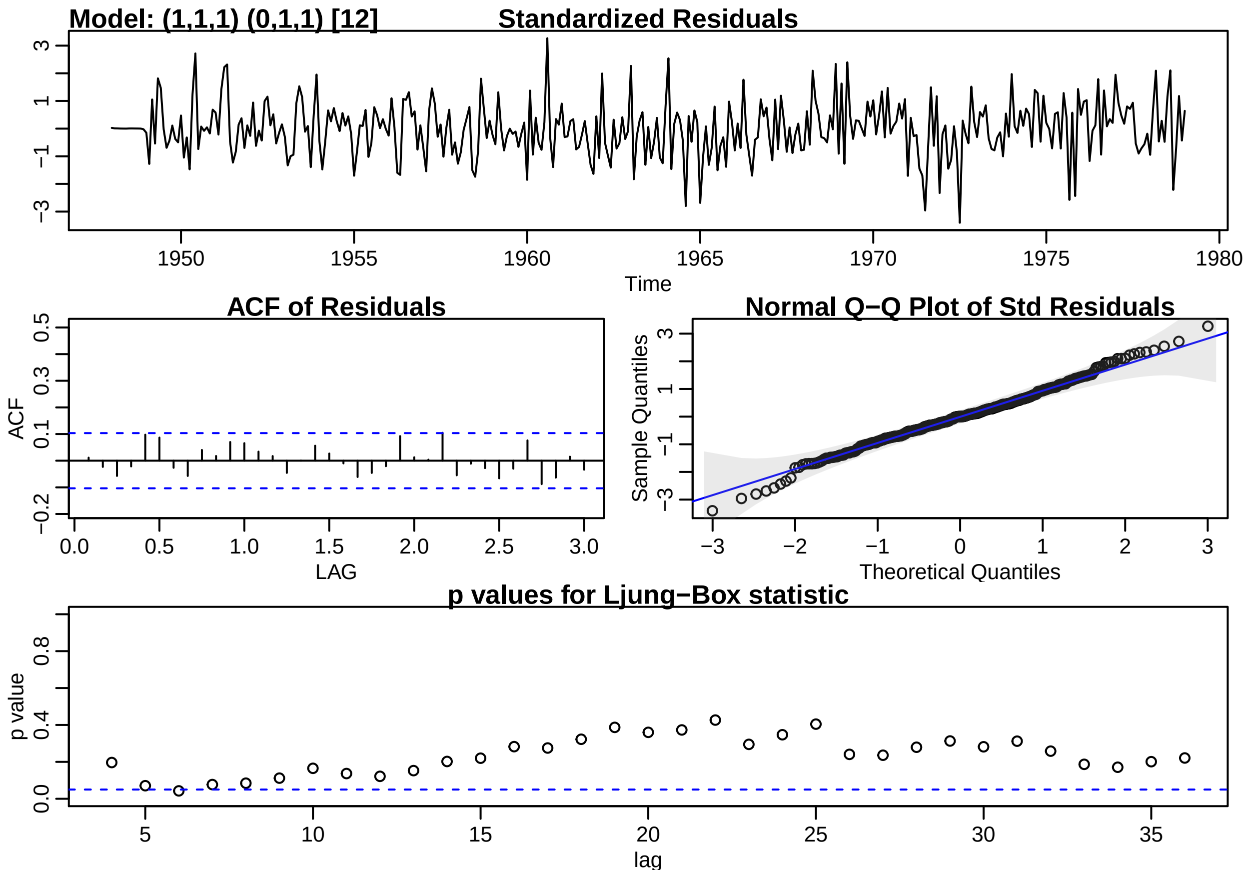



```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##           ma1     sma1
##       -0.4734  -0.7861
## s.e.   0.0598   0.0451
##
## sigma^2 estimated as 47.4:  log likelihood = -1211.28,  aic = 2428.56
##
## $degrees_of_freedom
## [1] 371
##
## $ttable
##      Estimate     SE  t.value p.value
## ma1   -0.4734 0.0598  -7.9097       0
## sma1  -0.7861 0.0451 -17.4227       0
##
## $AIC
## [1] 4.869388
##
## $AICc
```

```
## [1] 4.874924
##
## $BIC
## [1] 3.890415
```

```r
# Add AR term and conclude
sarima(birth,1,1,1,0,1,1,12)
```

```
## initial  value 2.218186
## iter   2 value 2.032584
## iter   3 value 1.982464
## iter   4 value 1.975643
## iter   5 value 1.971721
## iter   6 value 1.967284
## iter   7 value 1.963840
## iter   8 value 1.961106
## iter   9 value 1.960849
## iter  10 value 1.960692
## iter  11 value 1.960683
## iter  12 value 1.960675
## iter  13 value 1.960672
## iter  13 value 1.960672
## iter  13 value 1.960672
## final  value 1.960672
## converged
## initial  value 1.940459
## iter   2 value 1.934425
## iter   3 value 1.932752
## iter   4 value 1.931750
## iter   5 value 1.931074
## iter   6 value 1.930882
## iter   7 value 1.930860
## iter   8 value 1.930859
## iter   8 value 1.930859
## final  value 1.930859
## converged
```

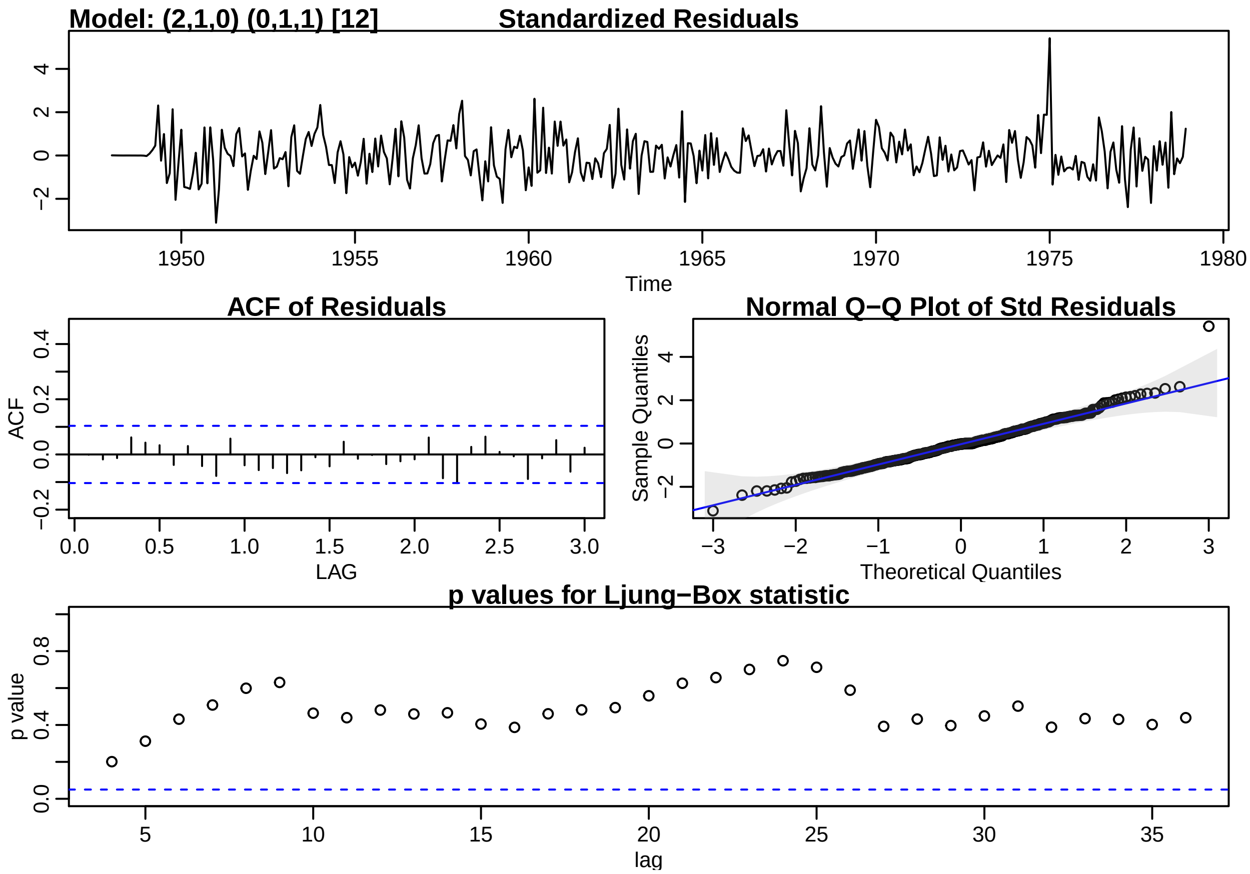



```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ma1     sma1
##       0.3038  -0.7006  -0.8000
## s.e.  0.0865   0.0604   0.0441
##
## sigma^2 estimated as 45.91:  log likelihood = -1205.93,  aic = 2419.85
##
## $degrees_of_freedom
## [1] 370
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1    0.3038 0.0865   3.5104   5e-04
## ma1   -0.7006 0.0604 -11.5984   0e+00
## sma1  -0.8000 0.0441 -18.1302   0e+00
##
## $AIC
## [1] 4.842869
##
```

```
## $AICc
## [1] 4.848523
##
## $BIC
## [1] 3.87441
```

```r
## FORECASTING SEASONAL DATA

# Fit your previous model to unemp and check the diagnostics
sarima(unemp, 2,1,0,0,1,1,12)
```

```
## initial  value 3.340809
## iter   2 value 3.105512
## iter   3 value 3.086631
## iter   4 value 3.079778
## iter   5 value 3.069447
## iter   6 value 3.067659
## iter   7 value 3.067426
## iter   8 value 3.067418
## iter   8 value 3.067418
## final  value 3.067418
## converged
## initial  value 3.065481
## iter   2 value 3.065478
## iter   3 value 3.065477
## iter   3 value 3.065477
## iter   3 value 3.065477
## final  value 3.065477
## converged
```



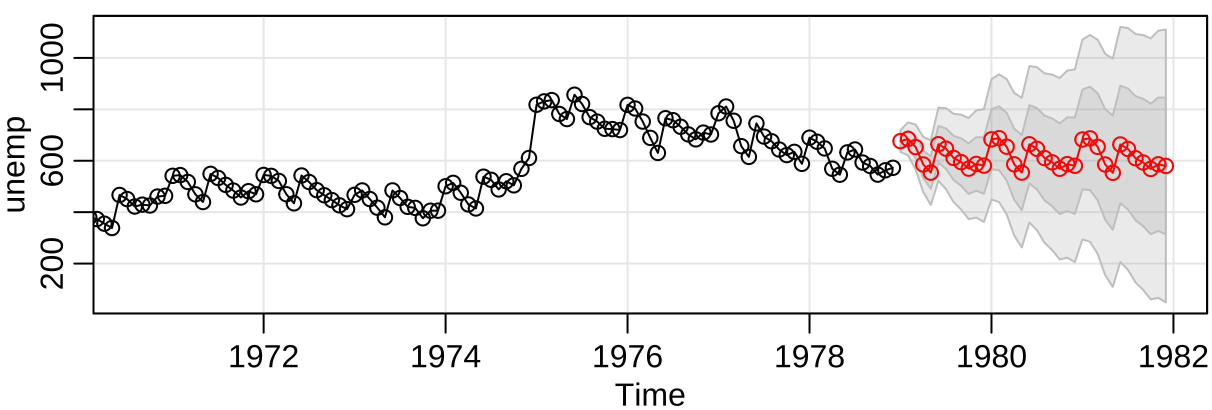

```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1     ar2     sma1
##       0.1351  0.2464  -0.6953
## s.e.  0.0513  0.0515   0.0381
##
## sigma^2 estimated as 449.6:  log likelihood = -1609.91,  aic = 3227.81
##
## $degrees_of_freedom
## [1] 369
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1    0.1351 0.0513   2.6326  0.0088
## ar2    0.2464 0.0515   4.7795  0.0000
## sma1  -0.6953 0.0381 -18.2362  0.0000
##
## $AIC
## [1] 7.12457
##
```

```
## $AICc
## [1] 7.130239
##
## $BIC
## [1] 6.156174
```

```
# Forecast the data 3 years into the future
sarima.for(unemp,n.ahead = 36,2,1,0,0,1,1,12)
```

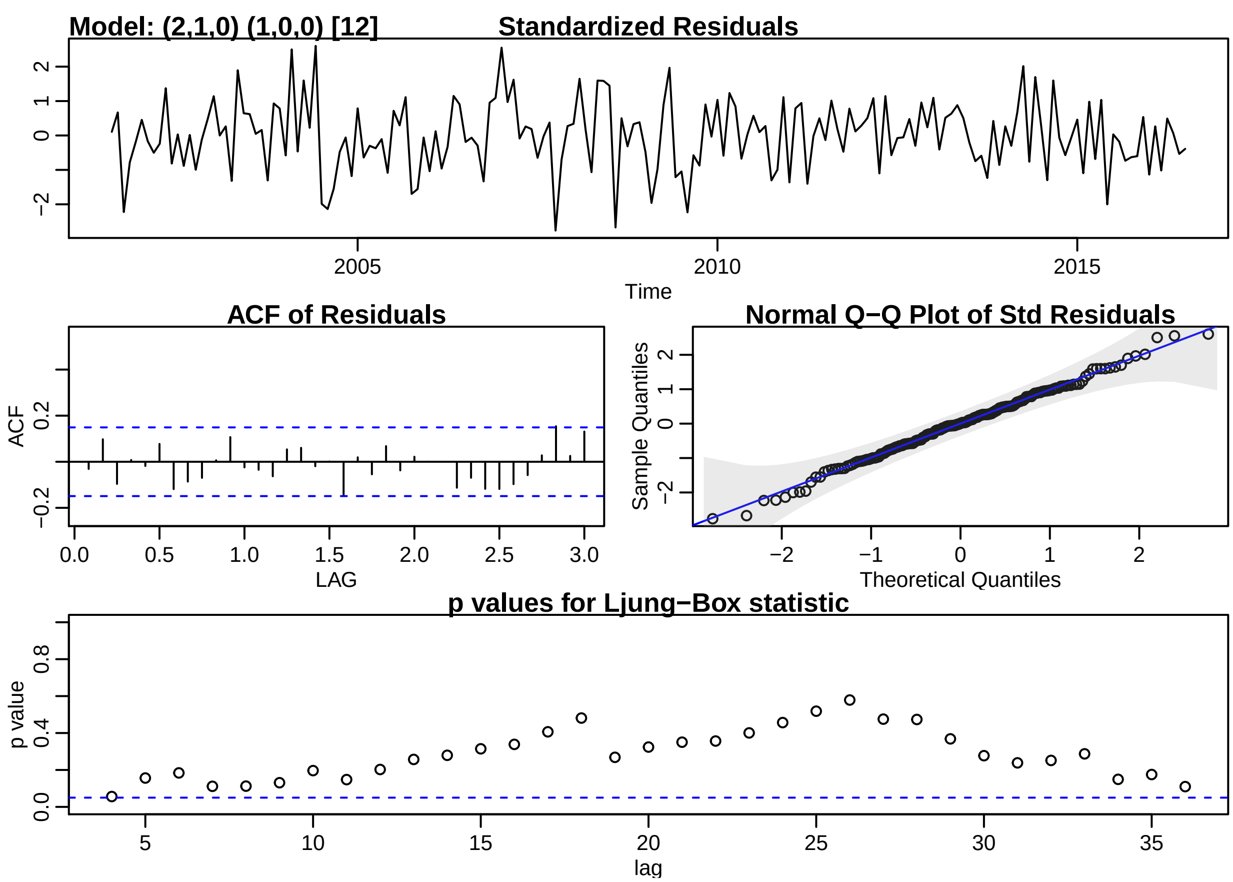
```
## $pred
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 1979 676.4664 685.1172 653.2388 585.6939 553.8813 664.4072 647.0657
## 1980 683.3045 687.7649 654.8658 586.1507 553.9285 664.1108 646.6220
## 1981 682.6406 687.0977 654.1968 585.4806 553.2579 663.4398 645.9508
##           Aug      Sep      Oct      Nov      Dec
## 1979 611.0828 594.6414 569.3997 587.5801 581.1833
## 1980 610.5345 594.0427 568.7684 586.9320 580.5249
## 1981 609.8632 593.3713 568.0970 586.2606 579.8535
##
## $se
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 1979  21.20465  32.07710  43.70167  53.66329  62.85364  71.12881  78.73590
## 1980 116.99599 124.17344 131.51281 138.60466 145.49706 152.12863 158.52302
## 1981 194.25167 201.10648 208.17066 215.11503 221.96039 228.64285 235.16874
##            Aug       Sep       Oct       Nov       Dec
## 1979  85.75096  92.28663  98.41329 104.19488 109.67935
## 1980 164.68623 170.63839 176.39520 181.97333 187.38718
## 1981 241.53258 247.74268 253.80549 259.72970 265.52323
```

```
# Fit the chicken model again and check diagnostics
sarima(chicken,2,1,0,1,0,0,12)
```




```
## initial  value 0.015039
## iter   2 value -0.226398
## iter   3 value -0.412955
## iter   4 value -0.460882
## iter   5 value -0.470787
## iter   6 value -0.471082
## iter   7 value -0.471088
## iter   8 value -0.471090
## iter   9 value -0.471092
## iter  10 value -0.471095
## iter  11 value -0.471095
```

```
## iter  12 value -0.471096
## iter  13 value -0.471096
## iter  14 value -0.471096
## iter  15 value -0.471097
## iter  16 value -0.471097
## iter  16 value -0.471097
## iter  16 value -0.471097
## final  value -0.471097
## converged
## initial  value -0.473585
## iter   2 value -0.473664
## iter   3 value -0.473721
## iter   4 value -0.473823
## iter   5 value -0.473871
## iter   6 value -0.473885
## iter   7 value -0.473886
## iter   8 value -0.473886
## iter   8 value -0.473886
## iter   8 value -0.473886
## final  value -0.473886
## converged
```



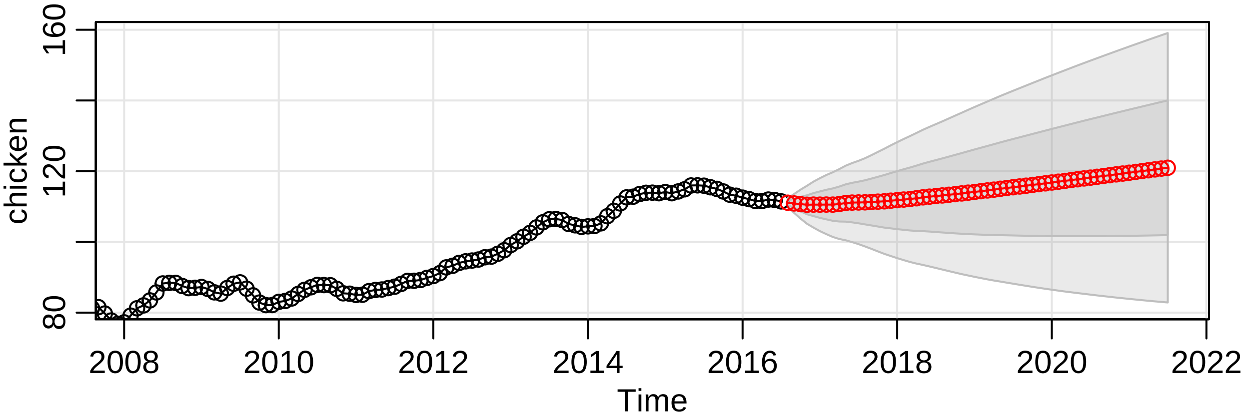

```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
```

```
##     Q), period = S), xreg = constant, optim.control = list(trace = trc, REPORT = 1,
##     reltol = tol))
##
## Coefficients:
##          ar1      ar2    sar1  constant
##       0.9154  -0.2494  0.3237    0.2353
## s.e.  0.0733   0.0739  0.0715    0.1973
##
## sigma^2 estimated as 0.3828:  log likelihood = -169.16,  aic = 348.33
##
## $degrees_of_freedom
## [1] 176
##
## $ttable
##          Estimate     SE t.value p.value
## ar1        0.9154 0.0733 12.4955  0.0000
## ar2       -0.2494 0.0739 -3.3728  0.0009
## sar1       0.3237 0.0715  4.5238  0.0000
## constant   0.2353 0.1973  1.1923  0.2347
##
## $AIC
## [1] 0.0842377
##
## $AICc
## [1] 0.09726452
##
## $BIC
## [1] -0.8448077
```

```r
# Forecast the chicken data 5 years into the future
sarima.for(chicken,n.ahead = 60,2,1,0,1,0,0,12)
```

```
## $pred
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2016
## 2017 110.5358 110.5612 110.5480 110.7055 111.0047 111.1189 111.1552
## 2018 111.8108 111.9782 112.1330 112.3431 112.5991 112.7952 112.9661
## 2019 114.1331 114.3464 114.5556 114.7827 115.0247 115.2473 115.4617
## 2020 116.7942 117.0224 117.2492 117.4819 117.7193 117.9505 118.1790
## 2021 119.5651 119.7980 120.0306 120.2650 120.5010 120.7350 120.9681
##           Aug      Sep      Oct      Nov      Dec
## 2016 111.0907 110.8740 110.6853 110.5045 110.5527
## 2017 111.1948 111.2838 111.3819 111.4825 111.6572
## 2018 113.1380 113.3260 113.5168 113.7085 113.9242
## 2019 115.6765 115.8965 116.1174 116.3386 116.5675
## 2020 118.4077 118.6380 118.8686 119.0993 119.3326
## 2021
##
## $se
##             Jan        Feb        Mar        Apr        May        Jun
## 2016
## 2017  3.7414959  4.1793190  4.5747009  4.9373266  5.2742129  5.5903499
## 2018  8.2010253  8.5605811  8.9054714  9.2372195  9.5572539  9.8667955
## 2019 12.0038164 12.2921541 12.5738417 12.8492868 13.1188976 13.3830477
## 2020 15.1557253 15.3959082 15.6323906 15.8653300 16.0948844 16.3212022
```

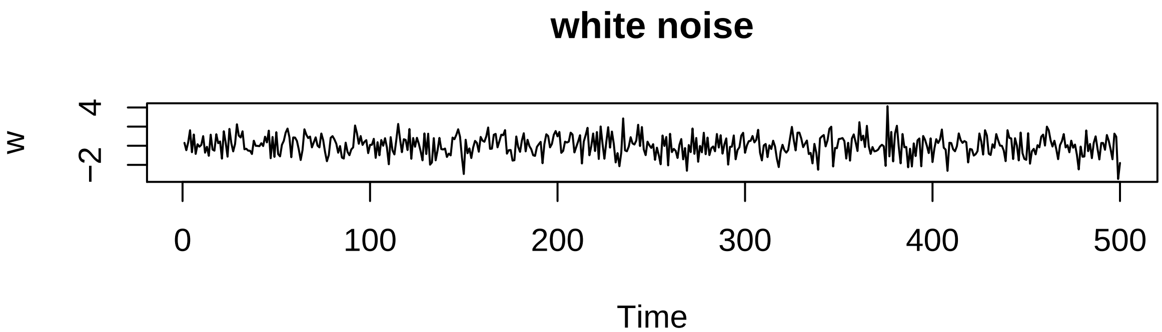
```
## 2021 17.8397890 18.0473081 18.2524651 18.4553364 18.6559977 18.8545213
##             Jul        Aug        Sep        Oct        Nov        Dec
## 2016                0.6187194  1.3368594  2.0462419  2.6867986  3.2486625
## 2017  5.8893133  6.2367345  6.6253573  7.0309771  7.4344077  7.8255932
## 2018 10.1668604 10.4736807 10.7857727 11.0980056 11.4063211 11.7085266
## 2019 13.6420693 13.9002670 14.1573839 14.4122197 14.6638269 14.9117124
## 2020 16.5444204 16.7657634 16.9852163 17.2025022 17.4174076 17.6298379
## 2021 19.0509752
```

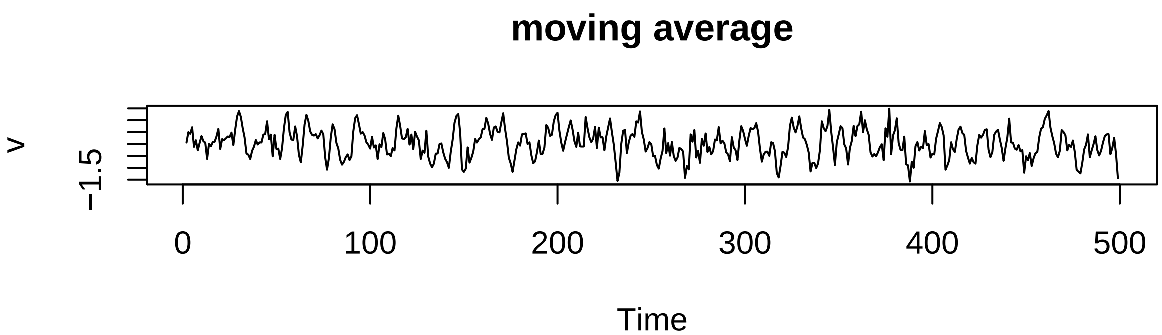


Pruebas series de tiempo de Time Series Analysis and its applications text

Página 13 Moving Average

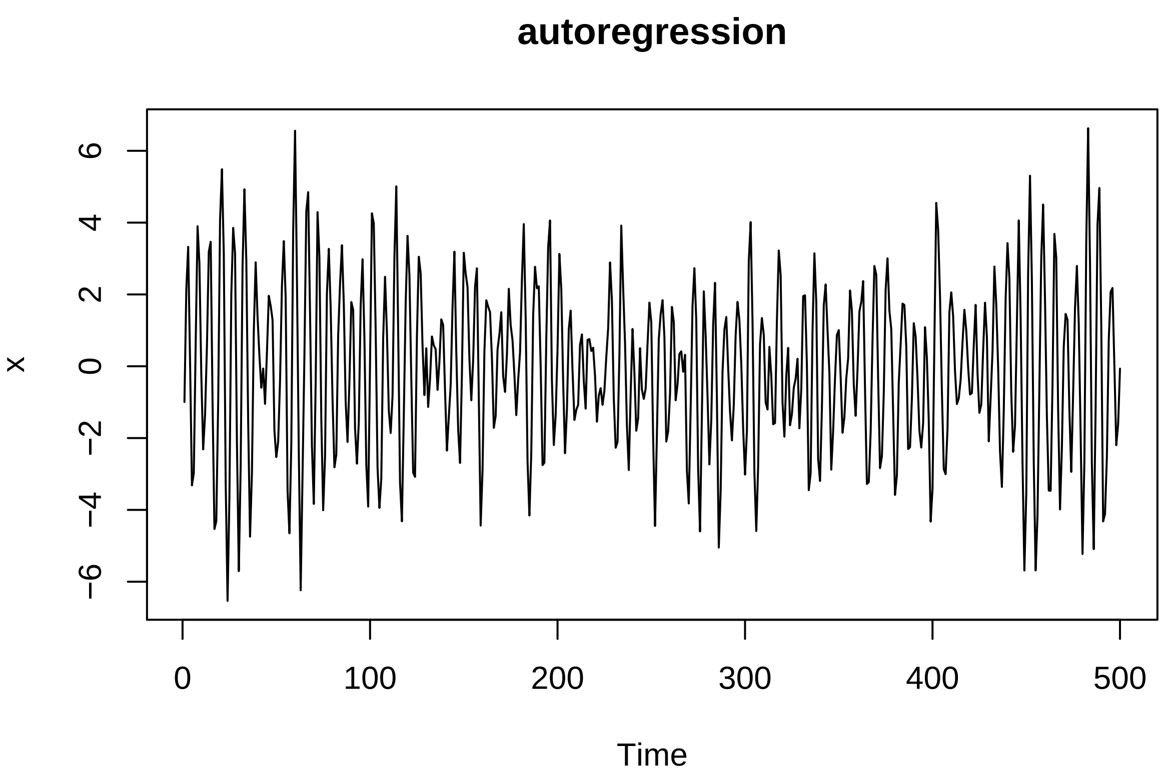
```
w = rnorm (500,0,1)
v = filter(w, sides = 2, rep(1/3,3))
par(mfrow=c(2,1))
plot.ts(w,main = "white noise")
plot.ts(v, main = "moving average")
```





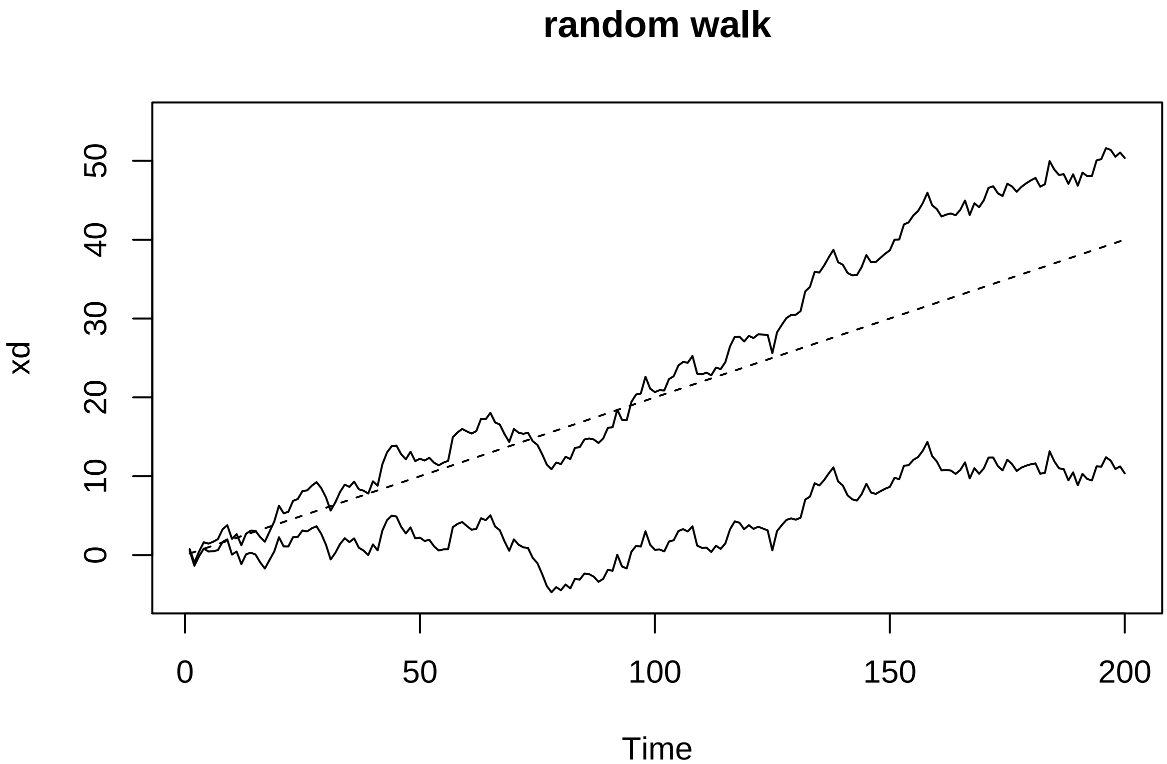
Autoregesivo

```
w = rnorm(550,0,1)
x = filter (w, filter = c(1,-0.9), method = "recursive")[-(1:50)]
plot.ts(x, main = "autoregression")
```



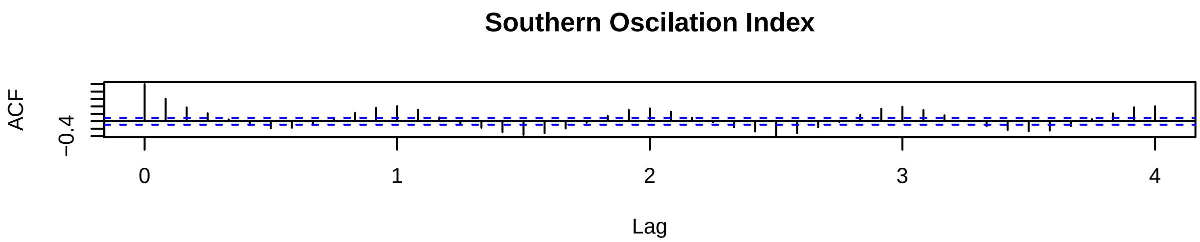
Autoregesivo con drift

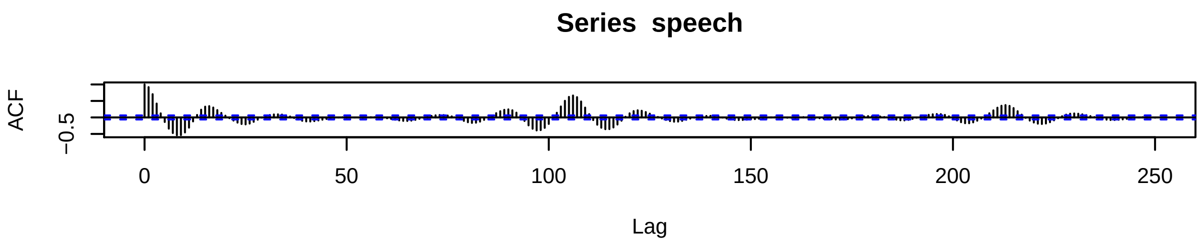
```
set.seed(154)
w = rnorm(200,0,1); x = cumsum (w)
wd = w + .2; xd = cumsum(wd)
plot.ts(xd, ylim = c(-5,55), main = "random walk")
lines(x); lines(0.2*(1:200), lty = "dashed")
```



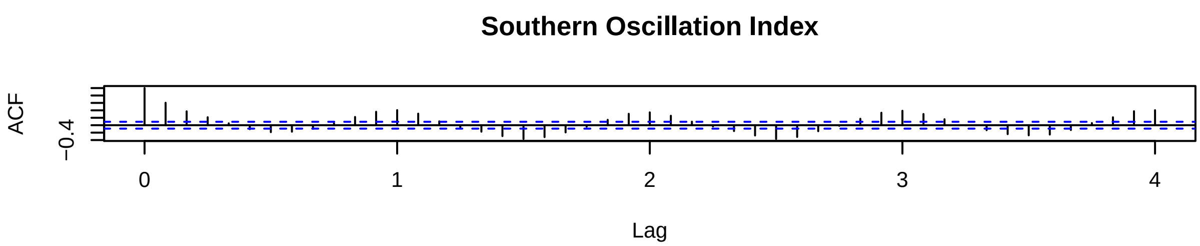
ACF

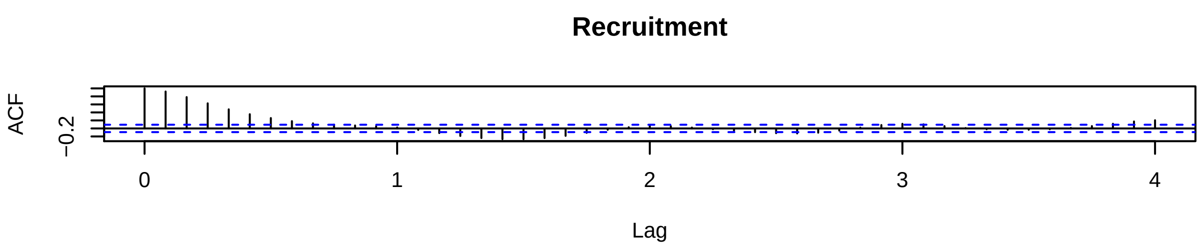
```
par(mfrow = c(3,1))
acf(soi, 48, main = "Southern Oscilation Index")
acf(speech, 250)

par(mfrow=c(3,1))
```

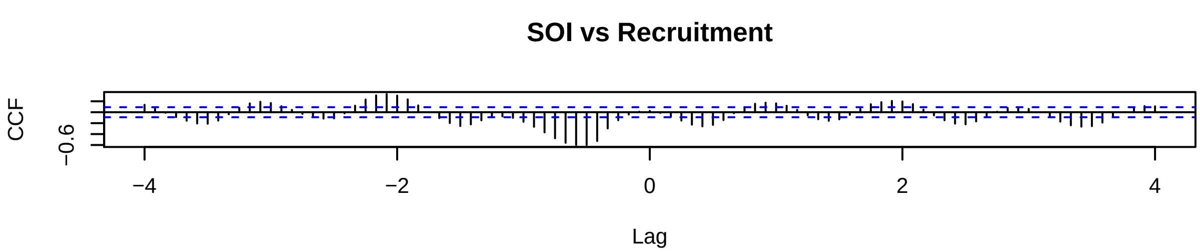




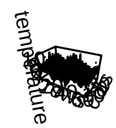

```r
acf(soi, 48, main="Southern Oscillation Index")
acf(rec, 48, main="Recruitment")
ccf(soi, rec, 48, main="SOI vs Recruitment", ylab="CCF")
```

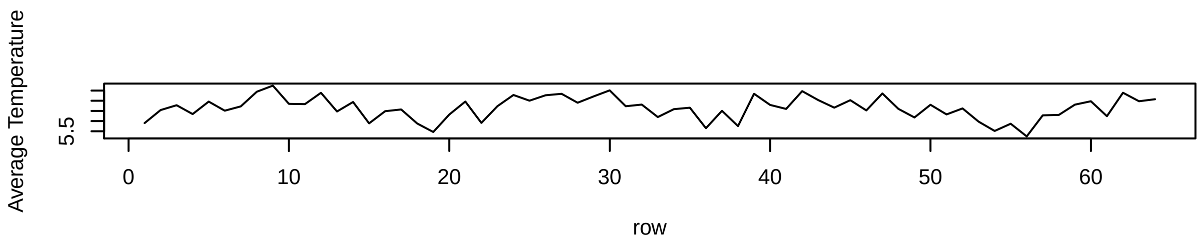






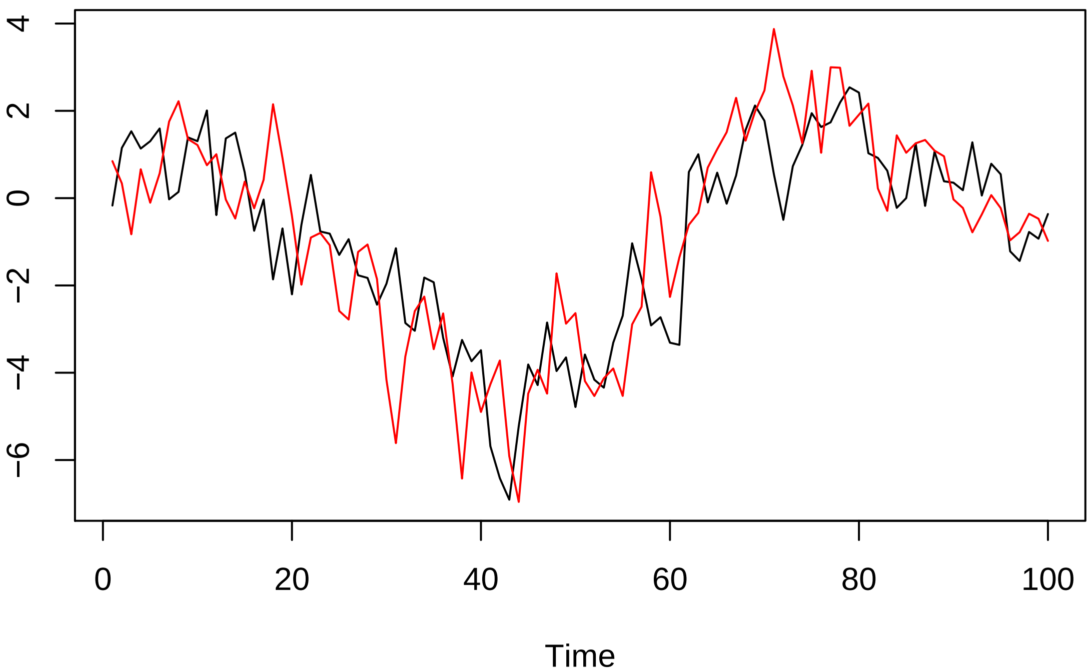

```r
# Varios vectores

persp(1:64, 1:36, soiltemp, phi=30, theta=30, scale=FALSE, expand=4, ticktype="detailed", xlab="rows", y

plot.ts(rowMeans(soiltemp), xlab="row", ylab="Average Temperature")
```





VECTORES AUTOREGRESIVOS

```
B1<-matrix(c(0.7, 0.2, 0.2, 0.7), 2)
var1<-VAR.sim(B=B1,n=100,include="none")
ts.plot(var1, type="l", col=c(1,2))
```




```
VARselect(var1, lag.max=9, type="const")
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      1      1      1      1
##
## $criteria
##                   1          2          3          4         5         6
## AIC(n) -0.004781104 0.05373217 0.05239784 0.05727888 0.1294645 0.1453231
```

```
## HQ(n)   0.062008485 0.16504815 0.20824021 0.25764765 0.3743596 0.4347447
## SC(n)   0.160770072 0.32965079 0.43868392 0.55393241 0.7364854 0.8627116
## FPE(n)  0.995277884 1.05543577 1.05443691 1.06032595 1.1409261 1.1609733
##                 7         8         9
## AIC(n) 0.1819582 0.2581421 0.3365710
## HQ(n)  0.5159061 0.6366164 0.7595717
## SC(n)  1.0097141 1.1962654 1.3850617
## FPE(n) 1.2068686 1.3060686 1.4176983
```

```r
var1_est <- VAR(var1,p = 1)
```

```
## Warning in VAR(var1, p = 1): No column names supplied in y, using: y1, y2 , instead.
```

```r
var1_est
```

```
##
## VAR Estimation Results:
## =======================
##
## Estimated coefficients for equation y1:
## =======================================
## Call:
## y1 = y1.l1 + y2.l1 + const
##
##       y1.l1       y2.l1       const
##  0.67774002  0.23472999 -0.09862114
##
##
## Estimated coefficients for equation y2:
## =======================================
## Call:
## y2 = y1.l1 + y2.l1 + const
##
##      y1.l1      y2.l1      const
## 0.41445790 0.58815891 0.01955353
```

MODELOS ESTADO ESPACIO

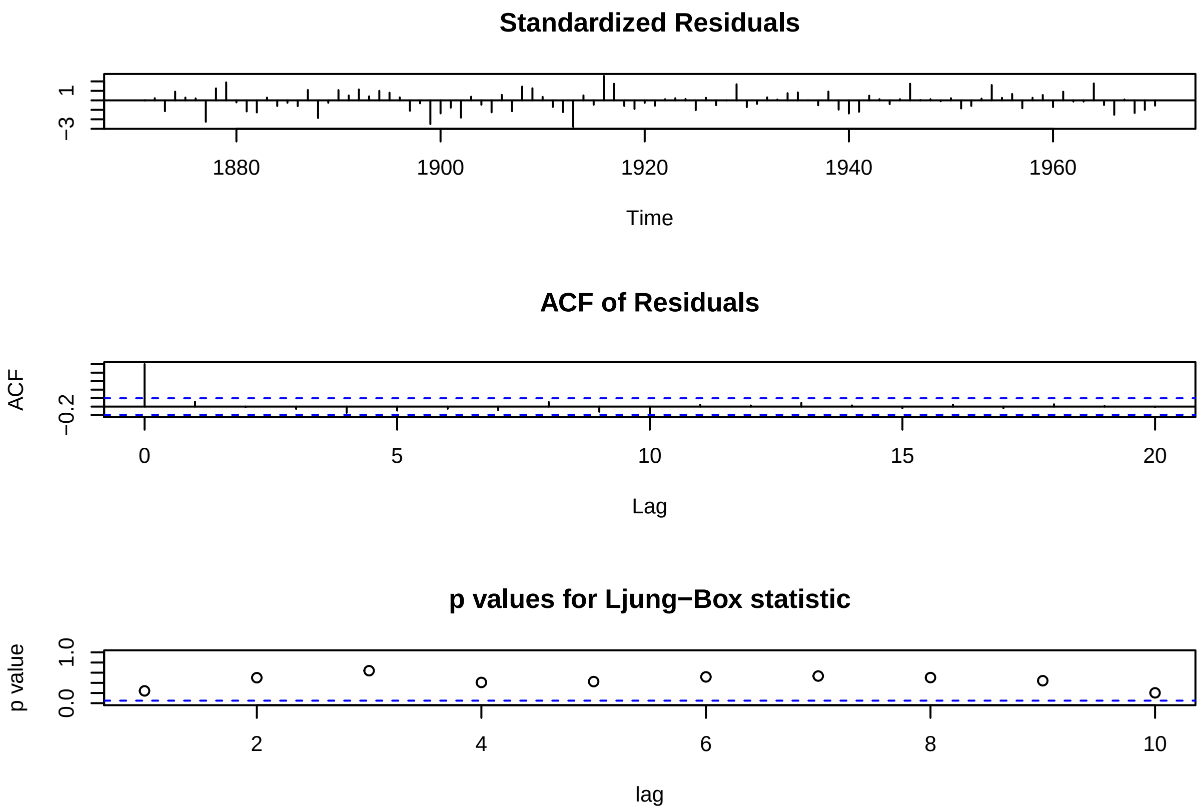
El paquete base de R permite evaluar modelos estado espacio con la función

Más información sobre modelos estado espacio se puede encontrar aquí: link StructTS

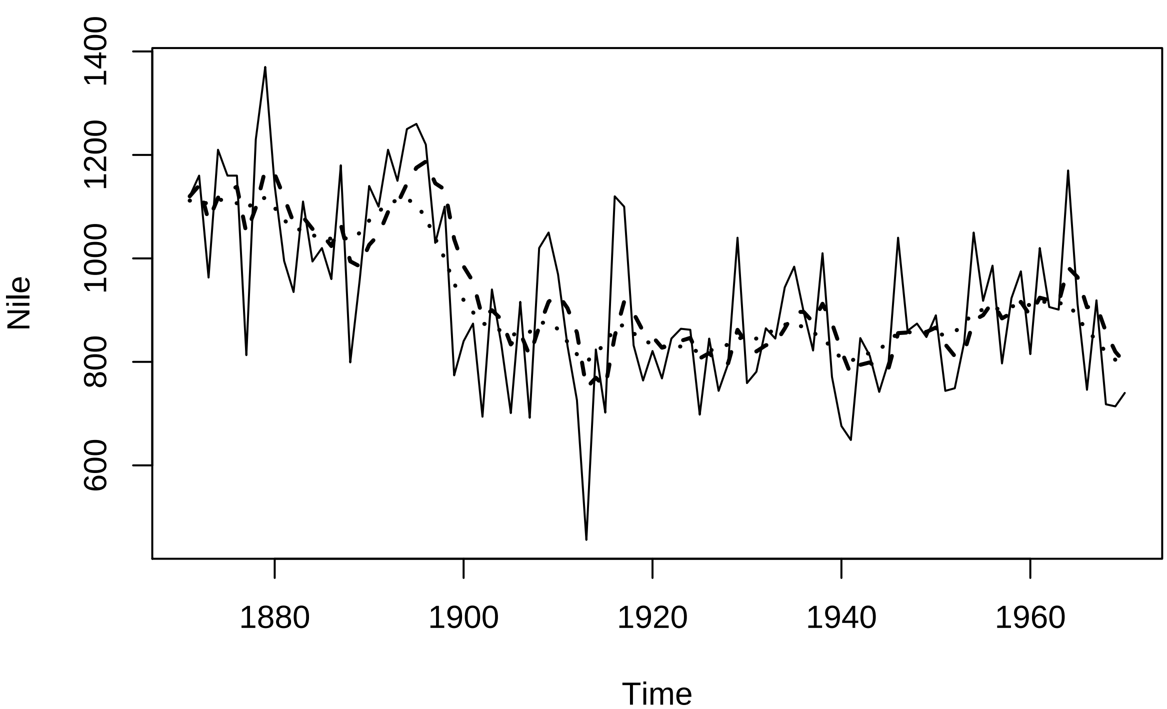
```r
# rm(list = ls())
fitNile <- StructTS(Nile, "level")
fitNile
```

```
##
## Call:
## StructTS(x = Nile, type = "level")
##
## Variances:
##   level  epsilon
##    1469    15099
```

```r
tsdiag(fitNile)
```

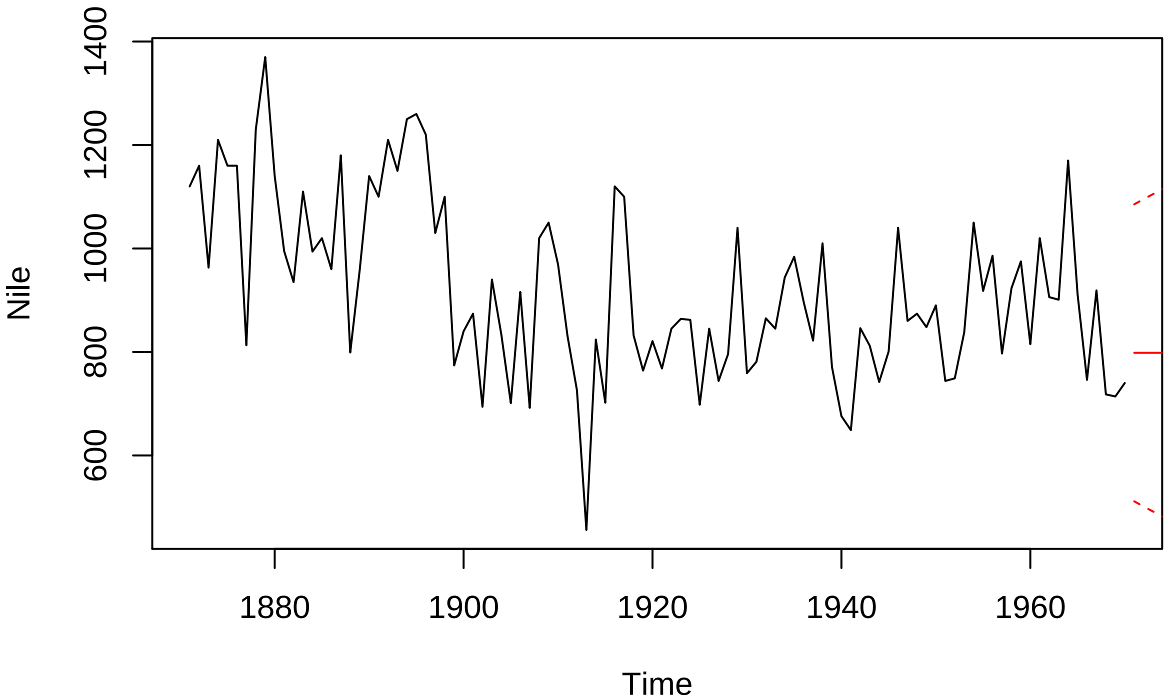



```r
plot(Nile, type = "l")
lines(fitted(fitNile), lty = "dashed", lwd = 2)
lines(tsSmooth(fitNile), lty = "dotted", lwd = 2)
```




```r
SP_fit <- StructTS(Nile, "level")
SP_forecast <- predict(SP_fit, n.ahead = 100)$pred
```

```r
SP_forecast_se <- predict(SP_fit, n.ahead = 100)$se
plot(Nile, type = "l")
points(SP_forecast, type = "l", col = 2)
points(SP_forecast - 2*SP_forecast_se, type = "l", col = 2, lty = 2)
points(SP_forecast + 2*SP_forecast_se, type = "l", col = 2, lty = 2)
```




```r
# Del link anterior

## Structural time series models
par(mfrow = c(3, 1))
plot(Nile)
## local level model
(fit <- StructTS(Nile, type = "level"))
```

```
##
## Call:
## StructTS(x = Nile, type = "level")
##
## Variances:
##   level  epsilon
##    1469    15099
```

```r
lines(fitted(fit), lty = 2)              # contemporaneous smoothing
lines(tsSmooth(fit), lty = 2, col = 4)   # fixed-interval smoothing
plot(residuals(fit)); abline(h = 0, lty = 3)
## local trend model
(fit2 <- StructTS(Nile, type = "trend")) ## constant trend fitted
```

```
##
## Call:
## StructTS(x = Nile, type = "trend")
##
```

```
## Variances:
##   level    slope  epsilon
##    1427        0    15047
```

```r
pred <- predict(fit, n.ahead = 30)
## with 50% confidence interval
ts.plot(Nile, pred$pred,
        pred$pred + 0.67*pred$se, pred$pred -0.67*pred$se)
```