Time Series Analysis Lab B

Thijs Quast (thiqu264)

9/23/2019

Contents

Assignment 1. Computations with simulated data

a

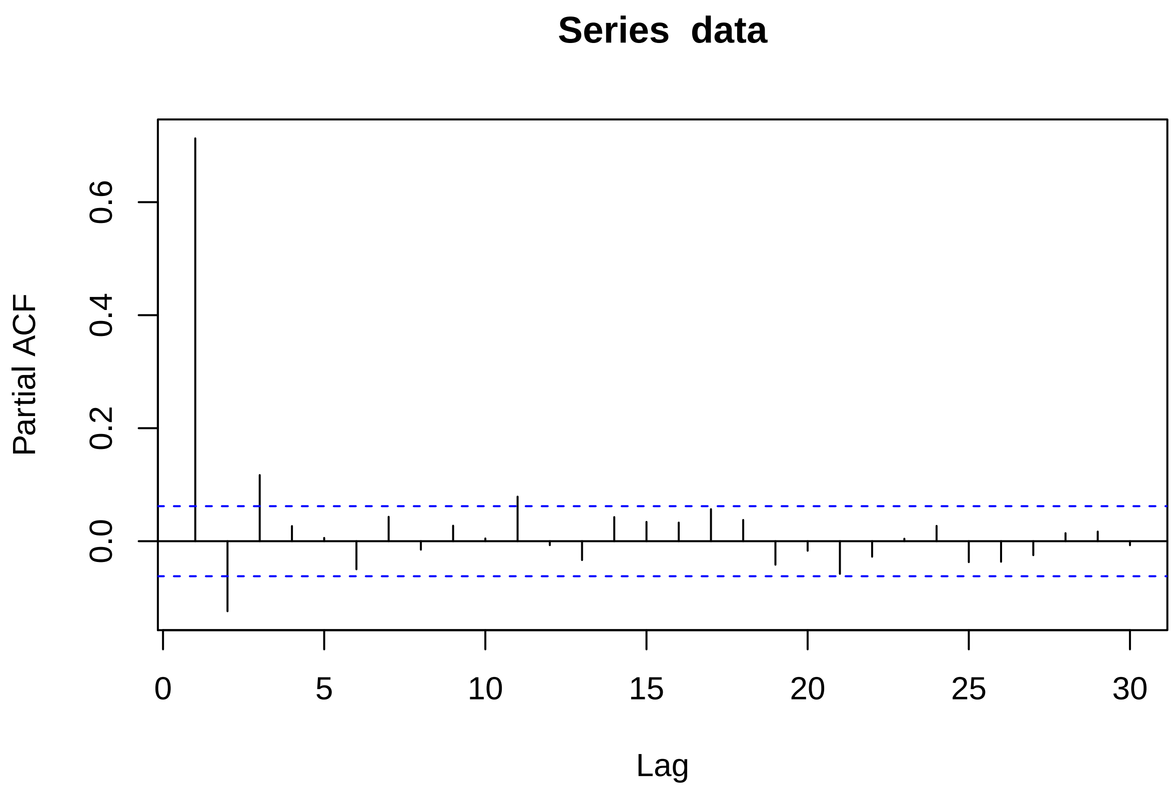
Generate 1000 observations from AR(3) process with phi1 = 0.8, phi2 = -0.2, phi3 = 0.1. Use these data and the definition of PACF to compute phi33 from the sample, i.e. write your own code that performs linear regressions on necessarily lagged variables and then computes an appropriate correlation. Compare the result with the output of function pacf() and with the theoretical value of phi33.

$$X_t = 0.8X_{t-1} - 0.2X_{t-2} + 0.1X_{t-3} + w_t$$
$$W_t distributed N(0,1)$$

```
# pacf using linear regression
set.seed(12345)
data=arima.sim(list(ar=c(0.8,-0.2,0.1)), n=1000)
data1=ts.intersect(x=data, x1=lag(data,-1), x2=lag(data,-2), x3=lag(data,-3), dframe = T)

res1=lm(x~x1+x2,data=data1)
res2=lm(x3~x2+x1,data=data1)
r1=residuals(res1)
r2=residuals(res2)
cor(cbind(r1,r2))
```

```
##           r1        r2
## r1 1.0000000 0.1146076
## r2 0.1146076 1.0000000
```

```
# pacf function
g=pacf(data)
```



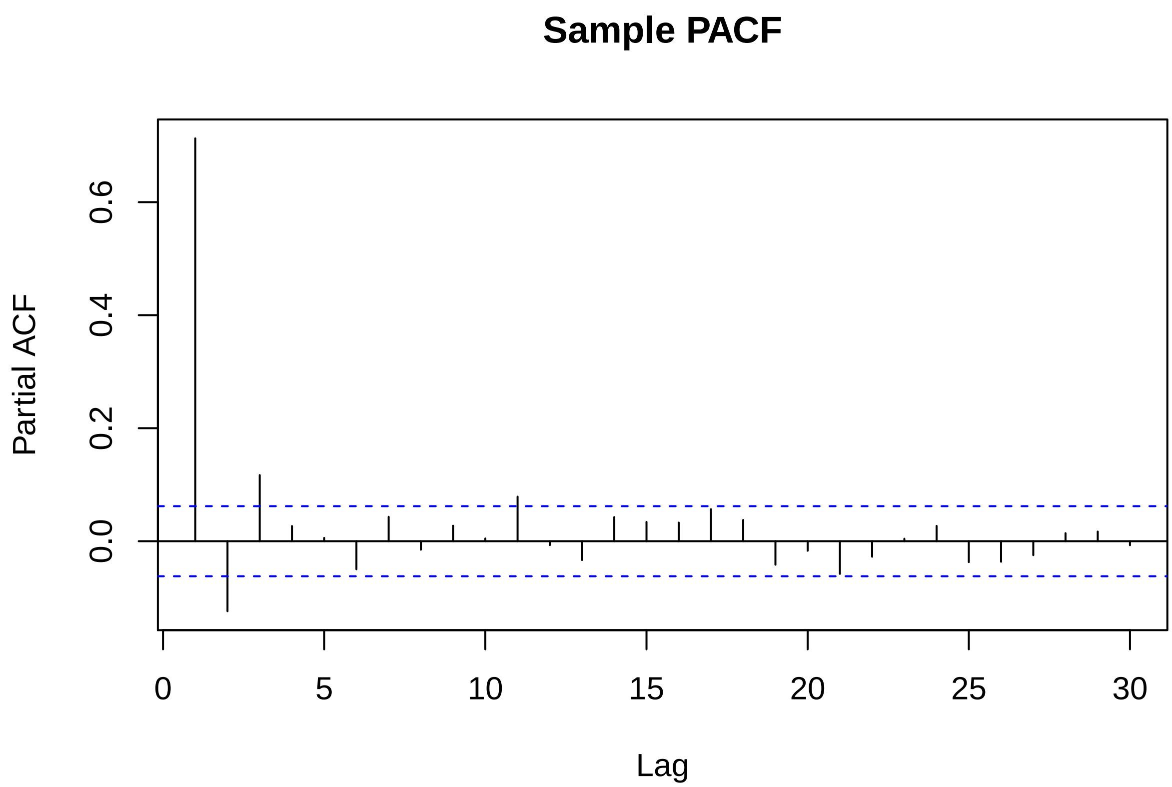

```r
g$acf[3]
```

```
## [1] 0.1170643
```

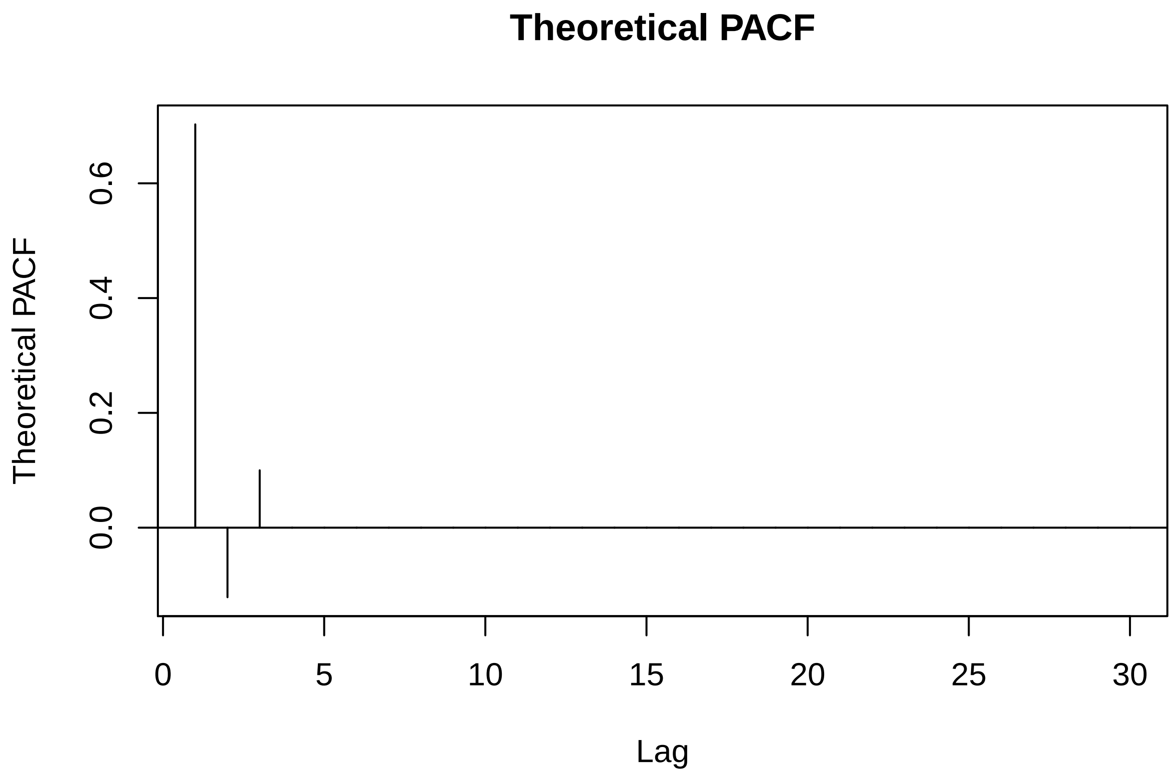
```r
# Theoretical pacf
theoretical_pacf <- ARMAacf(ar=c(0.8, -0.2, 0.1), lag.max = 30, pacf = TRUE)
theoretical_pacf[3]
```

```
## [1] 0.1
```

```r
# Comparing in plots
pacf(data, main="Sample PACF")
```




```
plot(theoretical_pacf, xlab = "Lag", ylab = "Theoretical PACF", type = "h",
     main = "Theoretical PACF")
abline(h=0)
```



b

```
set.seed(12345)
AR2 <- arima.sim(model = list(ar=c(0.8, 0.1)), n = 100)

# Yule-Walker
AR2_yw <- ar(x = AR2 ,method = "yw")

# Conditional Least Squares
AR2_cls <- arima(x = AR2, order = c(2,0,0), method = "CSS")

# ML
AR2_ML <- arima(x = AR2, order = c(2,0,0), method = "ML")
```

```
AR2_yw
```

```
##
## Call:
## ar(x = AR2, method = "yw")
##
## Coefficients:
##       1
## 0.8958
```

```
##
## Order selected 1  sigma^2 estimated as  1.267
```

```
AR2_cls
```

```
##
## Call:
## arima(x = AR2, order = c(2, 0, 0), method = "CSS")
##
## Coefficients:
##          ar1     ar2  intercept
##       0.8067  0.1205     0.6028
## s.e.  0.0984  0.0994     1.5134
##
## sigma^2 estimated as 1.129:  part log likelihood = -147.94
```

```
AR2_ML
```

```
##
## Call:
## arima(x = AR2, order = c(2, 0, 0), method = "ML")
##
## Coefficients:
##          ar1     ar2  intercept
##       0.7967  0.1189     0.8290
## s.e.  0.0992  0.1000     1.1385
##
## sigma^2 estimated as 1.126:  log likelihood = -148.71,  aic = 305.41
```

```
# Theoretical value phi2
```

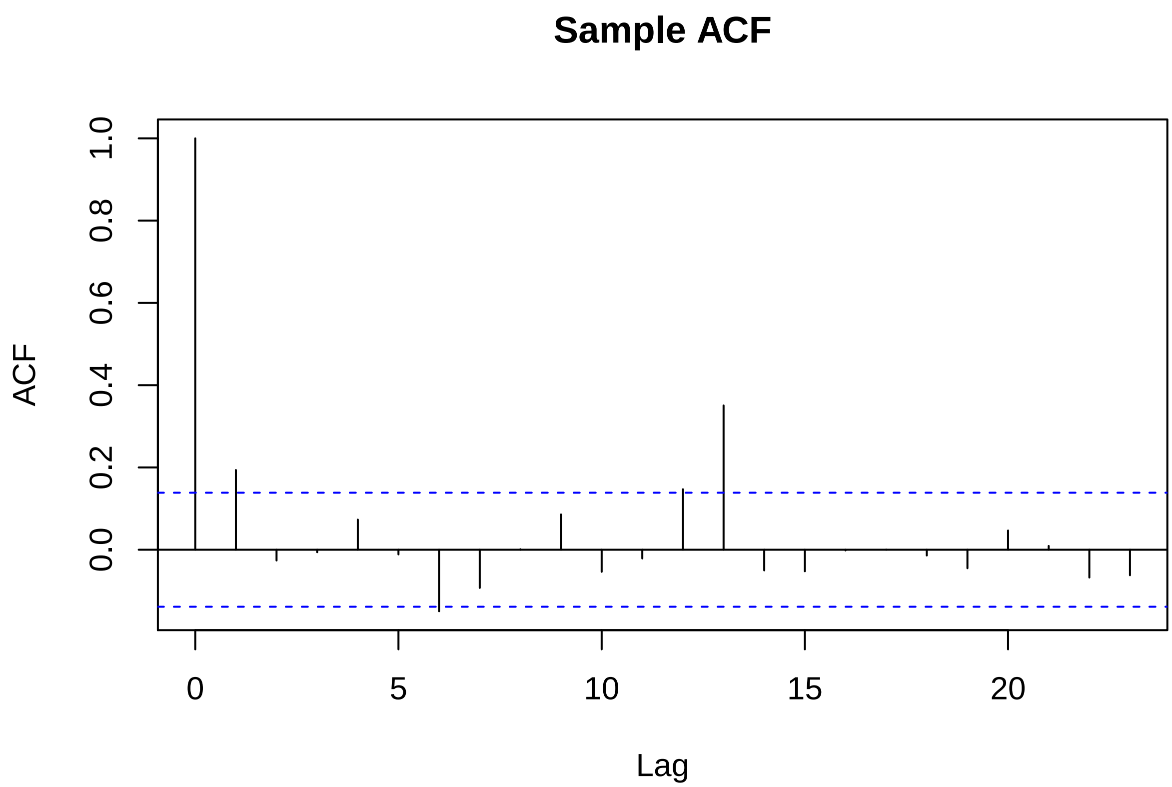
```
confint(AR2_ML)
```

```
##                 2.5 %    97.5 %
## ar1        0.60226569 0.9911182
## ar2       -0.07717562 0.3148831
## intercept -1.40236886 3.0603000
```

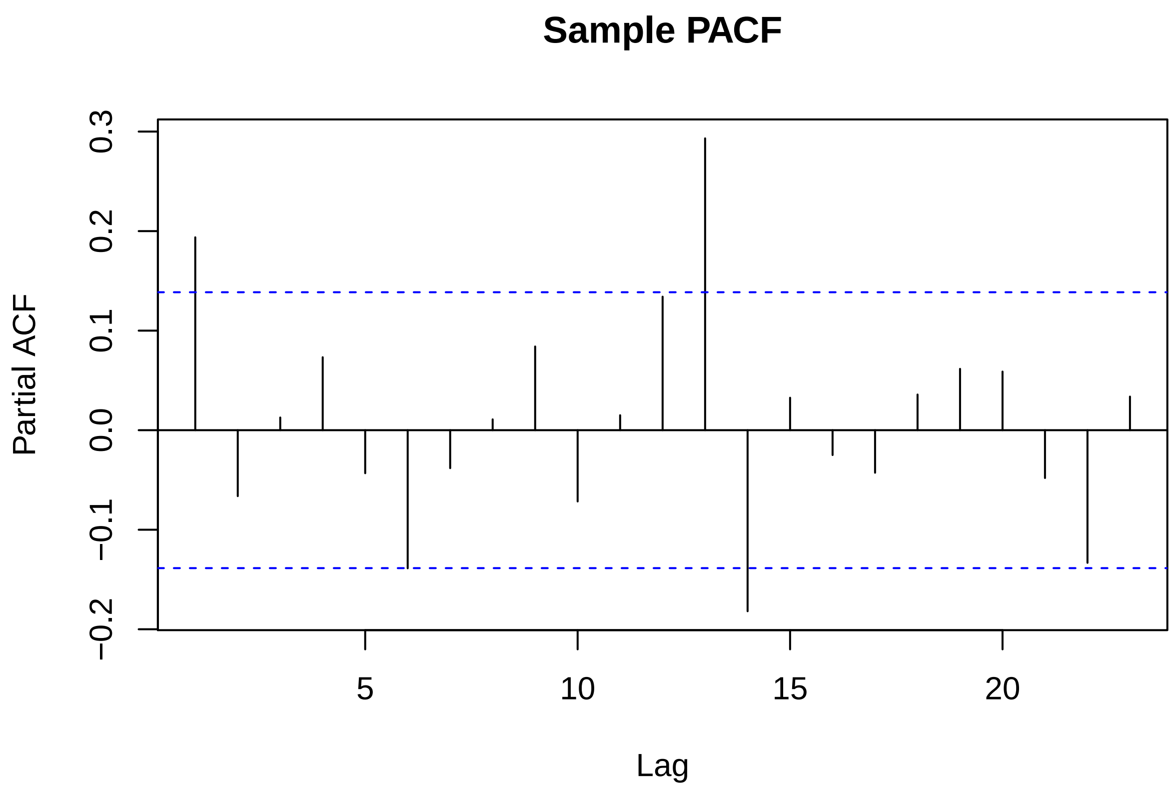
c

```
omega <- c(0.3, rep(0, 11), 0.6)
sAR <- arima.sim(model = list(order=c(0,0,13), ma=omega), n = 200)
```

```
# Sample ACF and Sample PACF
acf(sAR, main="Sample ACF")
```




```
pacf(sAR, main="Sample PACF")
```

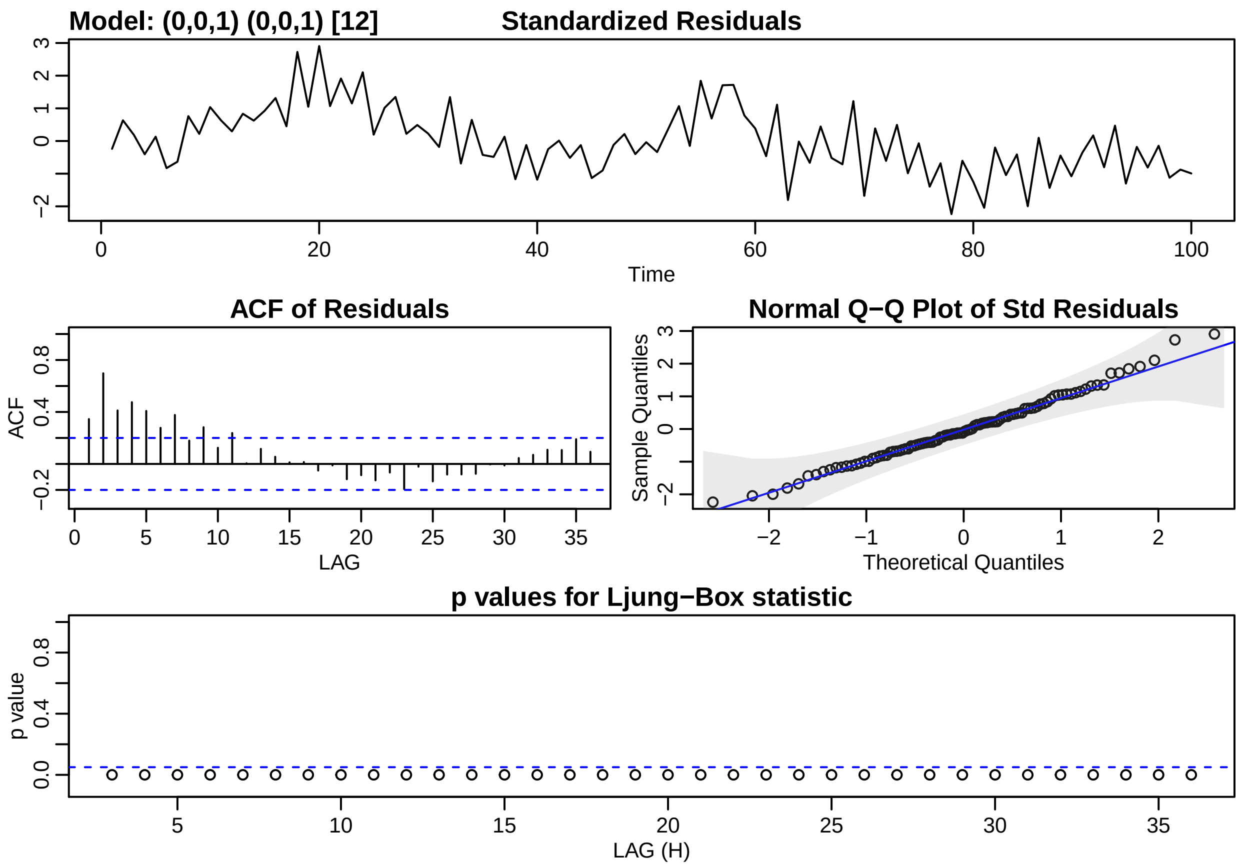


d

```r
library(astsa)
x <- arima.sim(n = 200, list(order(0,0,13), ma=c(0.3, rep(0, 10), 0.6, 0.18)))
fit <- sarima(xdata = AR2, 0, 0, 1, 0, 0, 1, 12)
```

```
## initial  value 0.919367
## iter   2 value 0.451723
## iter   3 value 0.448274
## iter   4 value 0.447462
## iter   5 value 0.447344
## iter   6 value 0.447337
## iter   7 value 0.447337
## iter   7 value 0.447337
## iter   7 value 0.447337
## final  value 0.447337
## converged
## initial  value 0.454905
## iter   2 value 0.454714
## iter   3 value 0.454623
## iter   4 value 0.454615
## iter   5 value 0.454614
## iter   6 value 0.454614
```

```
## iter   6 value 0.454614
## iter   6 value 0.454614
## final  value 0.454614
## converged
```

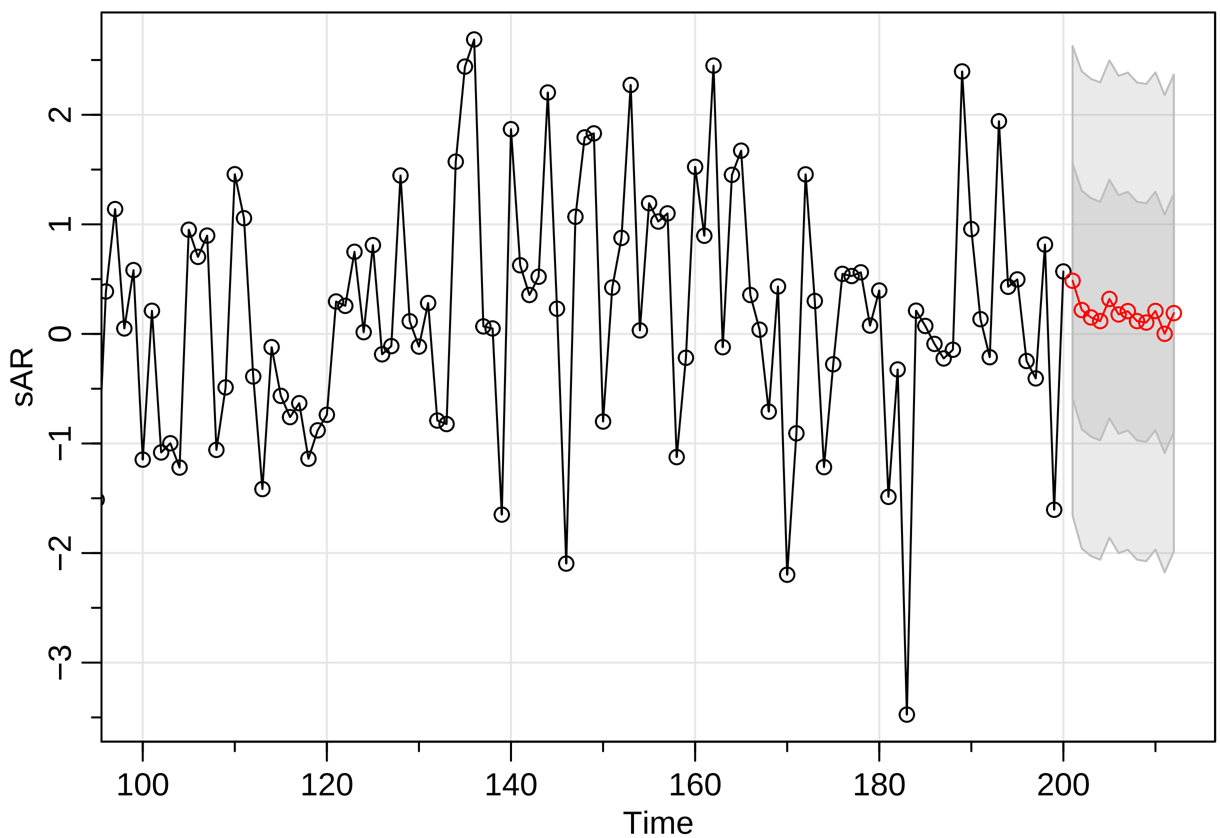



```
fit
```

```
## $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, transform.pars = trans,
##     fixed = fixed, optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##          ma1    sma1   xmean
##       0.8506  0.2455  1.1267
## s.e.  0.0673  0.1082  0.3507
##
## sigma^2 estimated as 2.434:  log likelihood = -187.36,  aic = 382.71
##
## $degrees_of_freedom
## [1] 97
##
## $ttable
```

```
##        Estimate     SE t.value p.value
## ma1      0.8506 0.0673 12.6438  0.0000
## sma1     0.2455 0.1082  2.2690  0.0255
## xmean    1.1267 0.3507  3.2125  0.0018
##
## $AIC
## [1] 3.827105
##
## $AICc
## [1] 3.829605
##
## $BIC
## [1] 3.931312
```

```r
sarima.for(xdata = sAR, 12, 0, 0, 1, 0, 0, 1, 12)
```




```
## $pred
## Time Series:
## Start = 201
## End = 212
## Frequency = 1
##  [1] 0.485089368 0.218369359 0.150341444 0.117319376 0.319330229
##  [6] 0.177696260 0.208118541 0.117070073 0.103611749 0.209873681
## [11] 0.001392272 0.190146786
##
```

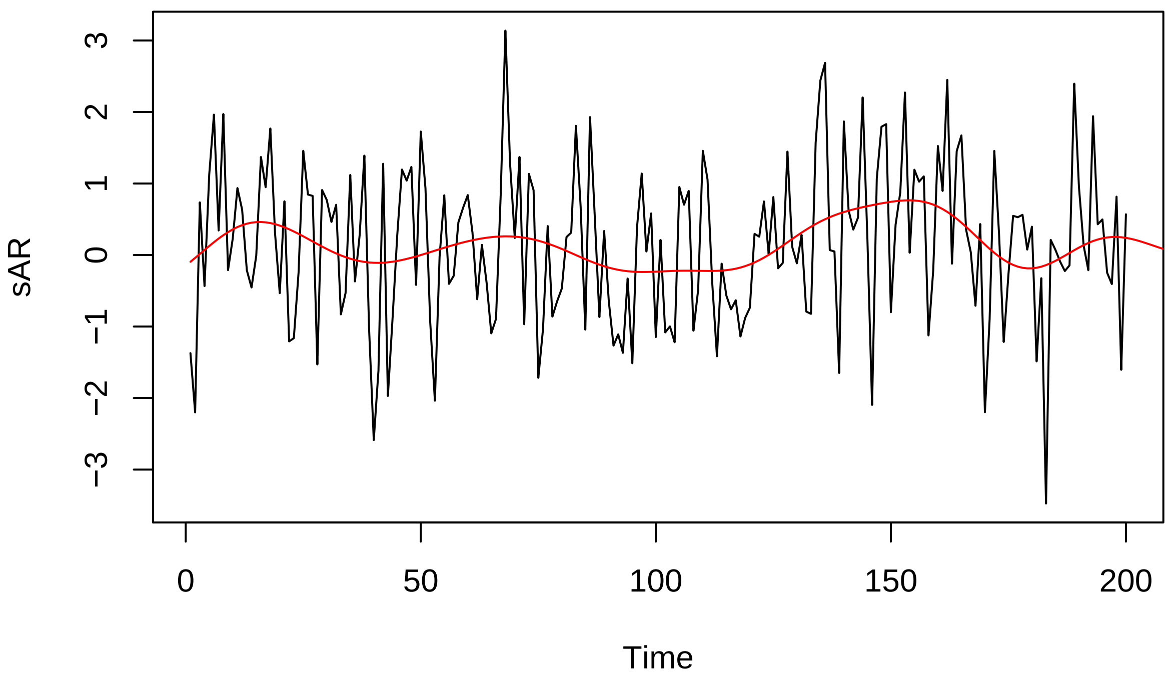
```
## $se
## Time Series:
## Start = 201
## End = 212
## Frequency = 1
##  [1] 1.071607 1.088828 1.088828 1.088828 1.088828 1.088828 1.088828
##  [8] 1.088828 1.088828 1.088828 1.088828 1.088828
```

```
# Forecast, using kernlab package
library(kernlab)
x <- c(1:200)
x_test <- c(0:230)
fit_gausspr <- gausspr(x = x, y=sAR)
```

```
## Using automatic sigma estimation (sigest) for RBF or laplace kernel
```

```
# Add noise to predictions
#noise <- rnorm(30)
y_pred <- predict(fit_gausspr, x_test)
```

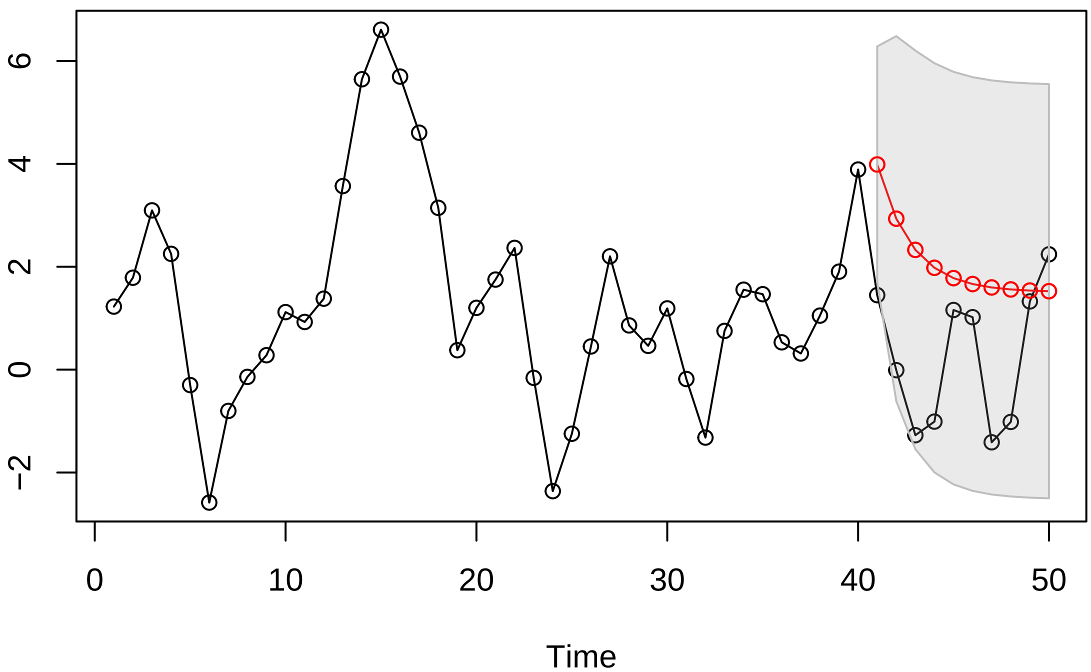
```
ts.plot(sAR)
lines(y_pred, col="red")
```



e

```r
set.seed(12345)
ARMA_11 <- arima.sim(model = list(order=c(1,0,1), ar=0.7, ma=0.5), n = 50)
ARMA_11_fit <- arima(x = ARMA_11[1:40], order = c(1,0,1))
```

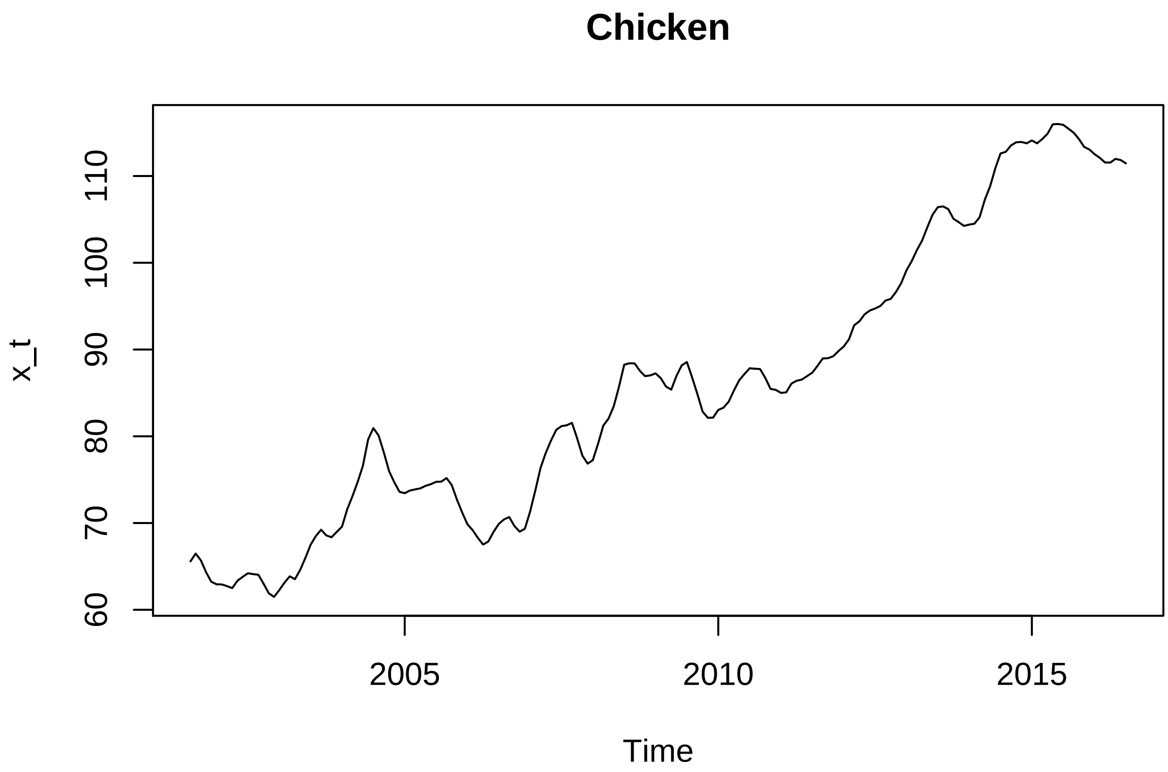
```r
prediction <- predict(ARMA_11_fit, n.ahead = 10)
ts.plot(ts(ARMA_11[1:50]), prediction$pred, col=c(1:2), type="o")


U = prediction$pred+2*prediction$se
L = prediction$pred-2*prediction$se
xx = c(time(U), rev(time(U)));  yy = c(L, rev(U))
polygon(xx, yy, border = 8, col = gray(.6, alpha = .2))
lines(prediction$pred, type="p", col=2)
```



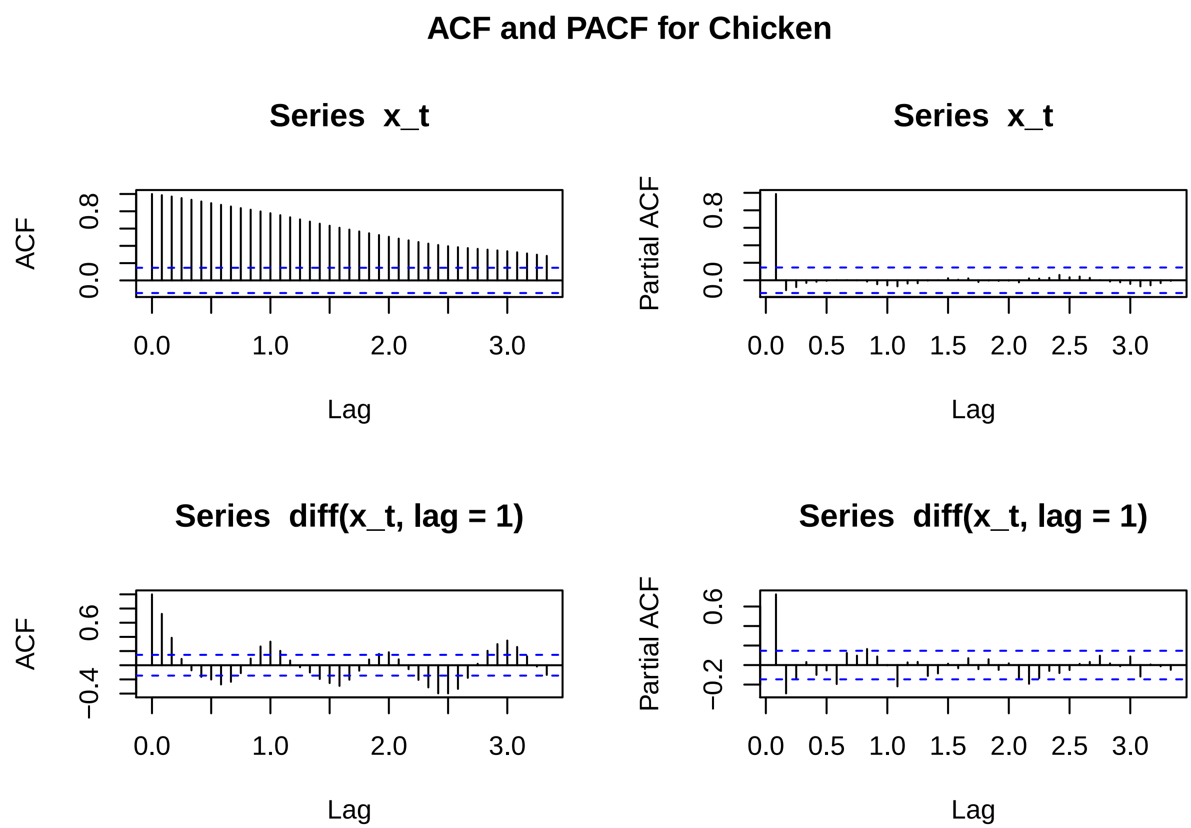
Assignment 2

a

```r
# Chicken
x_t <- chicken
ts.plot(x_t)
title(main = "Chicken")
```



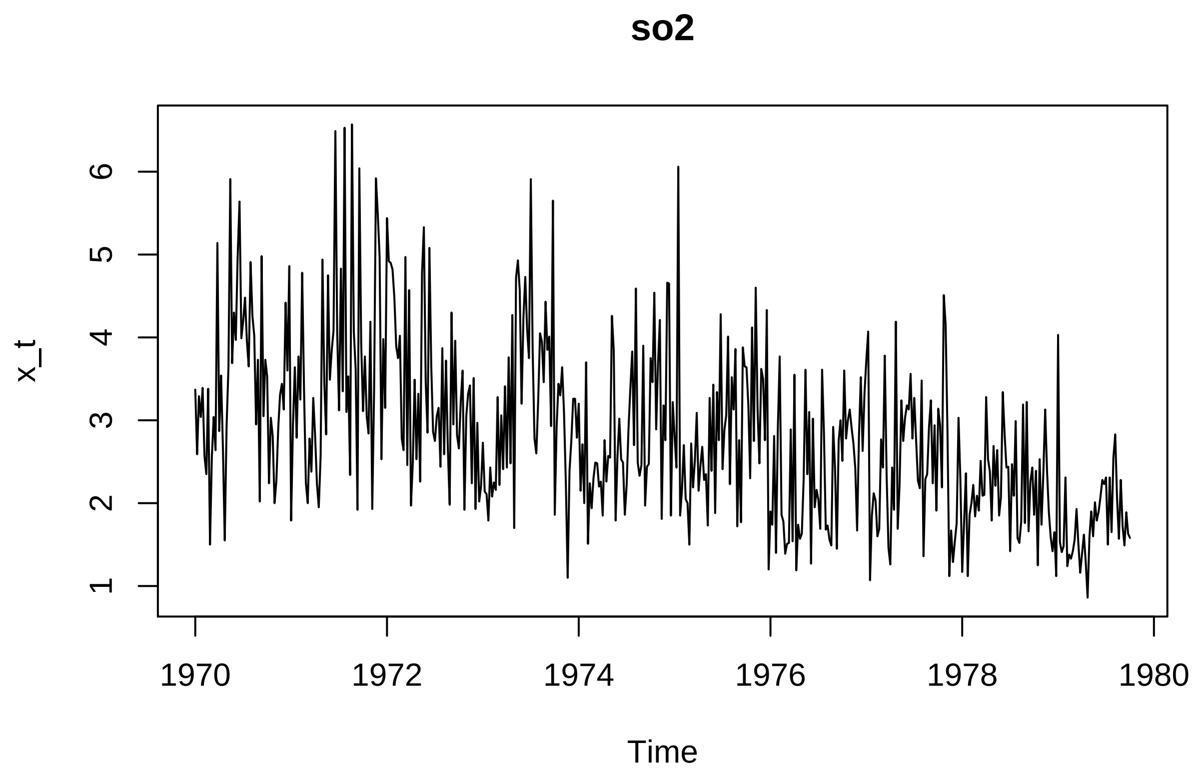

```r
# Chicken
par(mfrow=c(2,2), oma=c(0,0,2,0))
acf(x_t, lag.max = 40)
pacf(x_t, lag.max = 40)
acf(diff(x_t, lag = 1), lag.max = 40)
pacf(diff(x_t, lag = 1), lag.max = 40)
title(main = "ACF and PACF for Chicken", outer = TRUE)
```



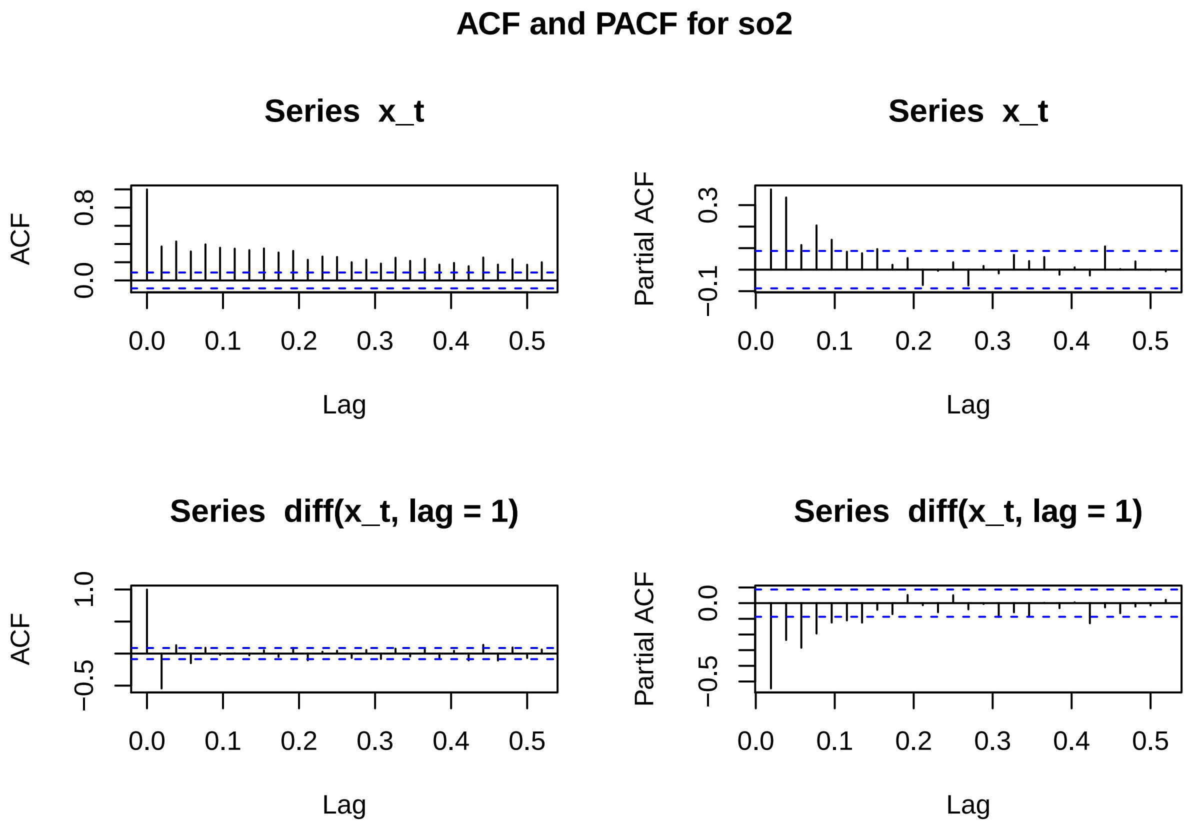
Based on the fact that there is high ACF, but PACF seems to be low, in addition there seems to be seasonality for 12 lags. So, I'd say we go for an ARIMA(1,1,0)x(1,0,0)12

b

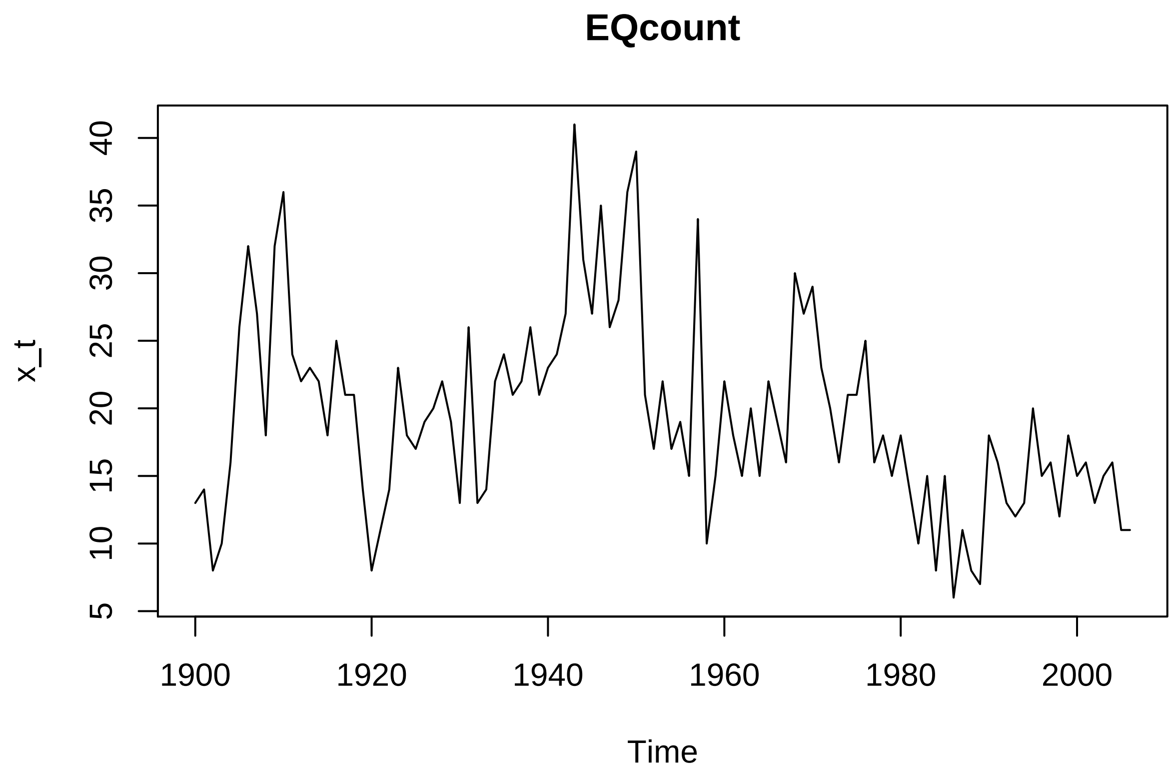
```r
#so2
x_t <- so2
ts.plot(x_t)
title(main = "so2")
```



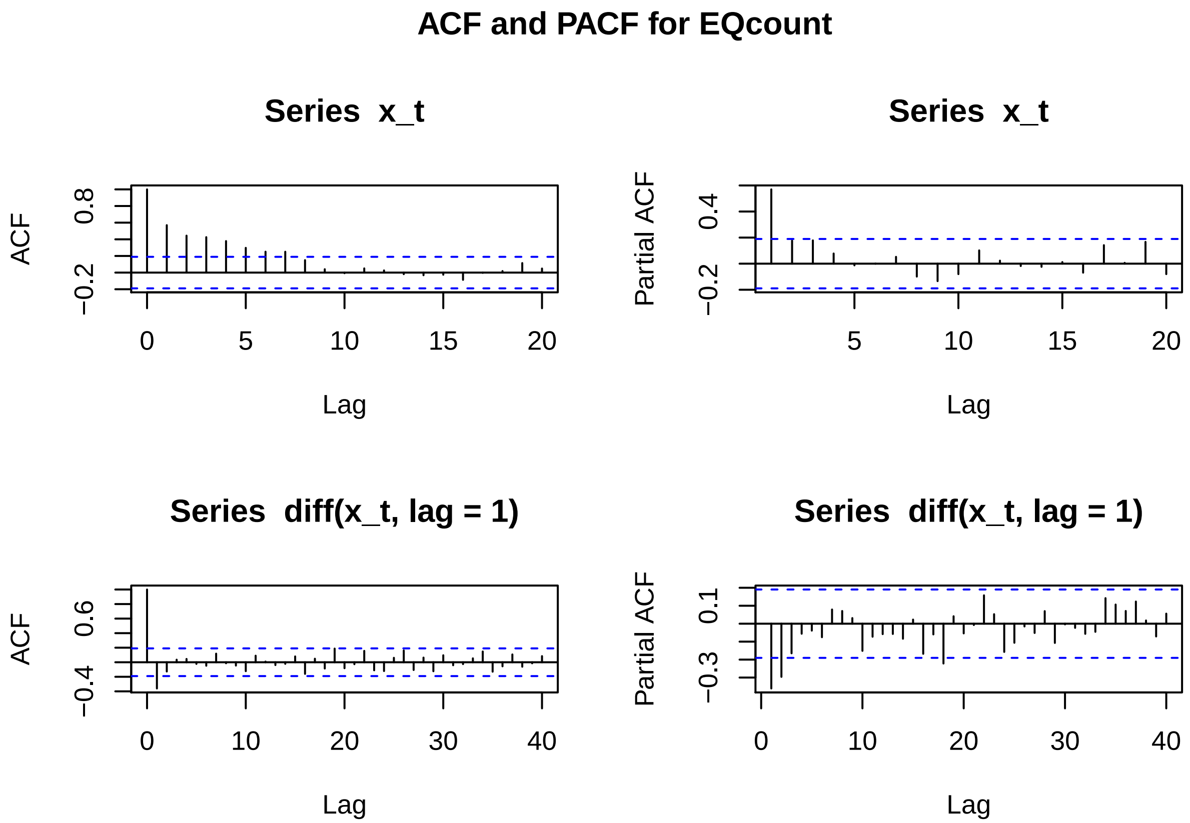

```r
# so2
par(mfrow=c(2,2), oma=c(0,0,2,0))
acf(x_t)
pacf(x_t)
acf(diff(x_t, lag = 1))
pacf(diff(x_t, lag = 1))
title(main = "ACF and PACF for so2", outer = TRUE)
```



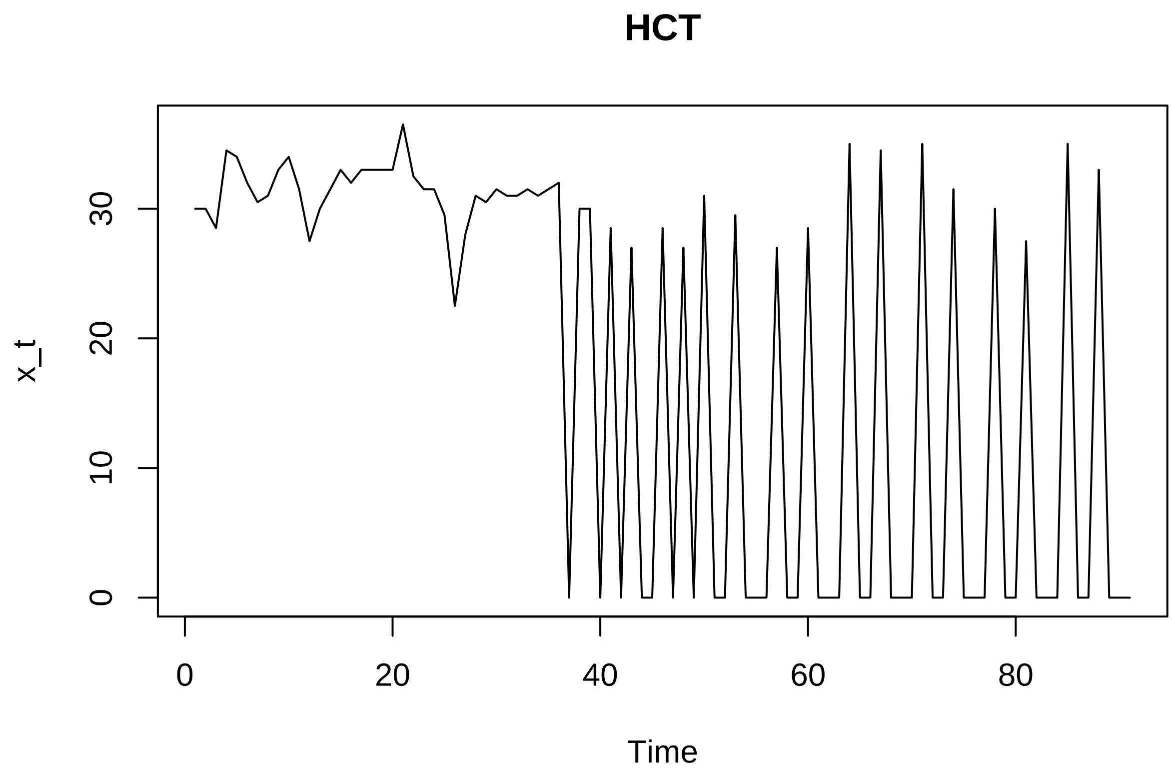

```r
# EQCount
x_t <- EQcount
ts.plot(x_t)
title(main = "EQcount")
```



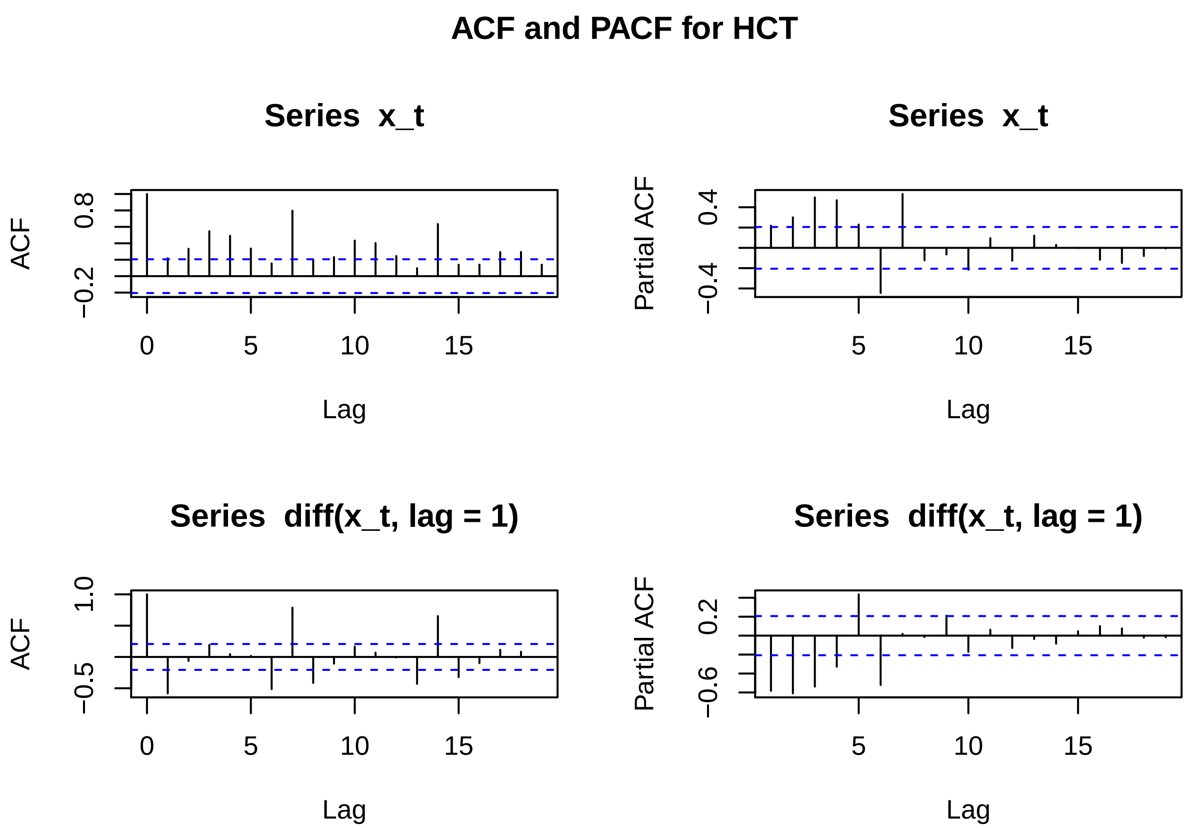

```
# EQCount
par(mfrow=c(2,2), oma=c(0,0,2,0))
acf(x_t)
pacf(x_t)
acf(diff(x_t, lag = 1),lag.max=40)
pacf(diff(x_t, lag = 1),lag.max=40)
title(main = "ACF and PACF for EQcount", outer = TRUE)
```




```r
x_t <- HCT
ts.plot(x_t)
title(main = "HCT")
```



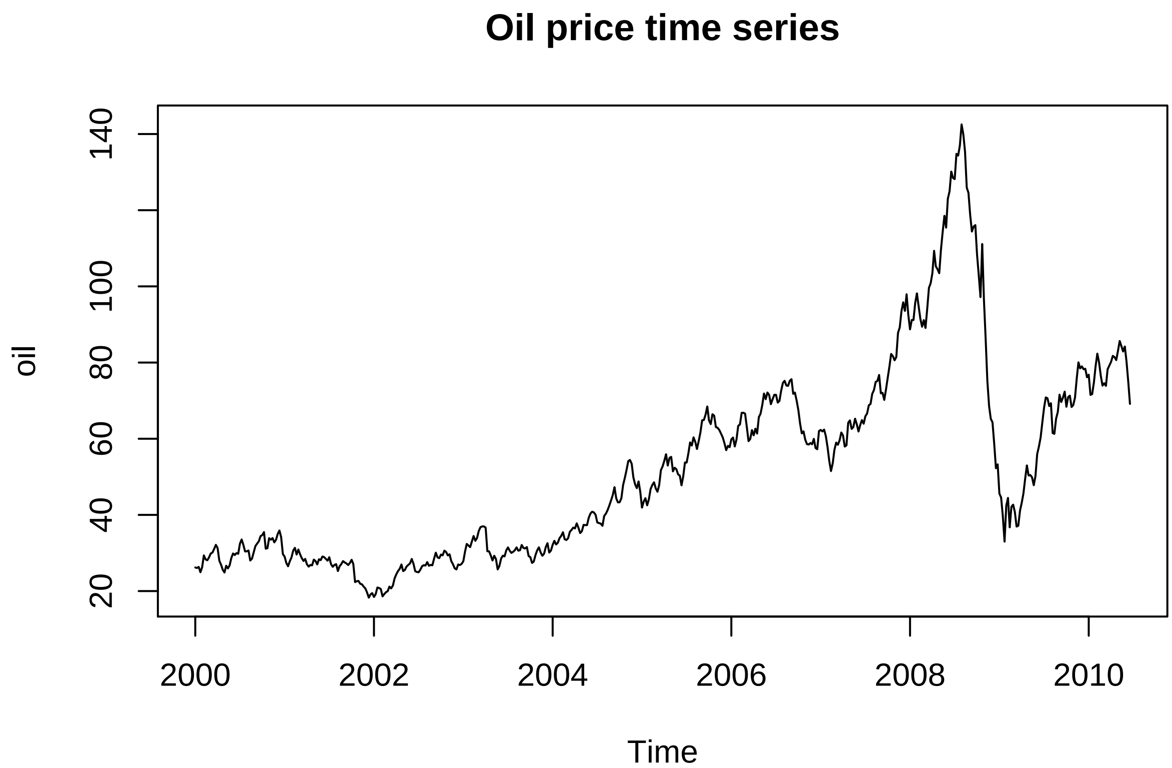

```
# EQCount
par(mfrow=c(2,2), oma=c(0,0,2,0))
acf(x_t)
pacf(x_t)
acf(diff(x_t, lag = 1))
pacf(diff(x_t, lag = 1))
title(main = "ACF and PACF for HCT", outer = TRUE)
```



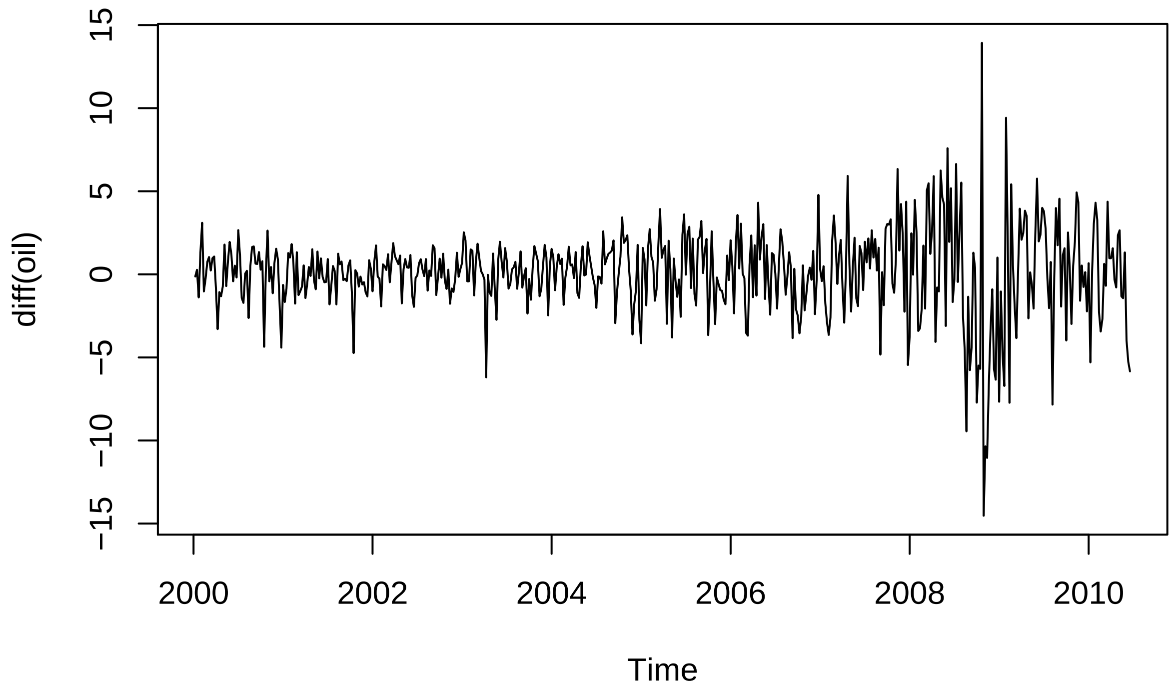
Assignment 3

a

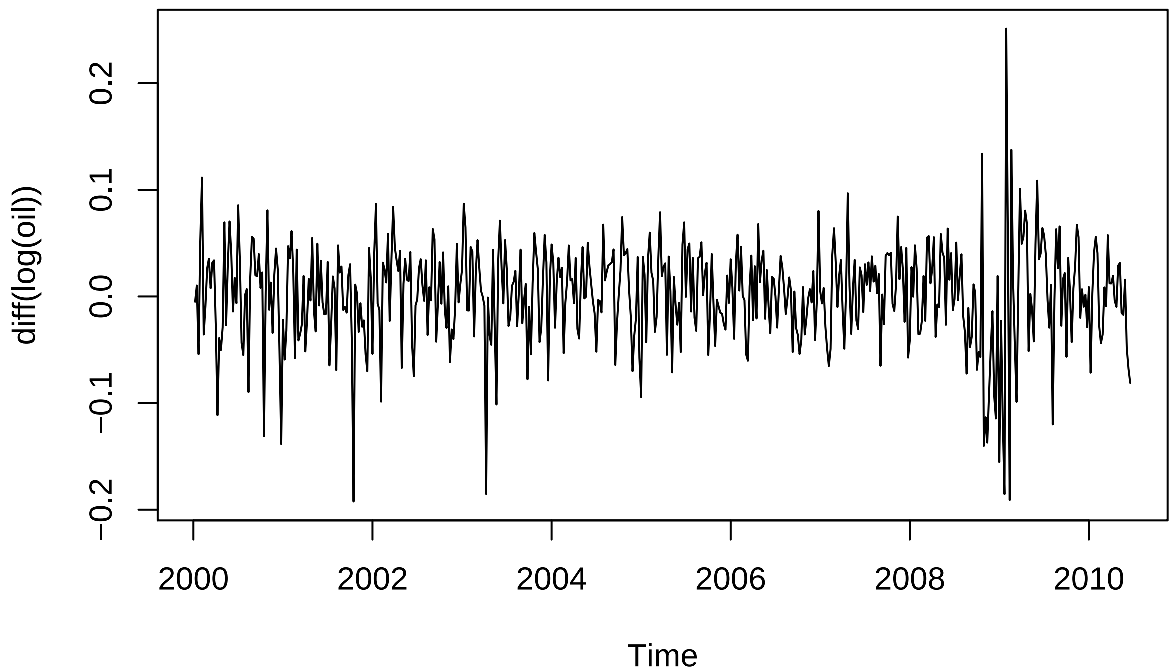
```r
ts.plot(oil)
title(main = "Oil price time series")
```




```r
# Non-stationary, so use differencing:
ts.plot(diff(oil))
```




```
# Variance is not constant everywhere, so:
ts.plot(diff(log(oil)))
```



Variance is much more equal now across the entire time series. Look at the y-scale of plot above.

```r
doil <- diff(log(oil))
```

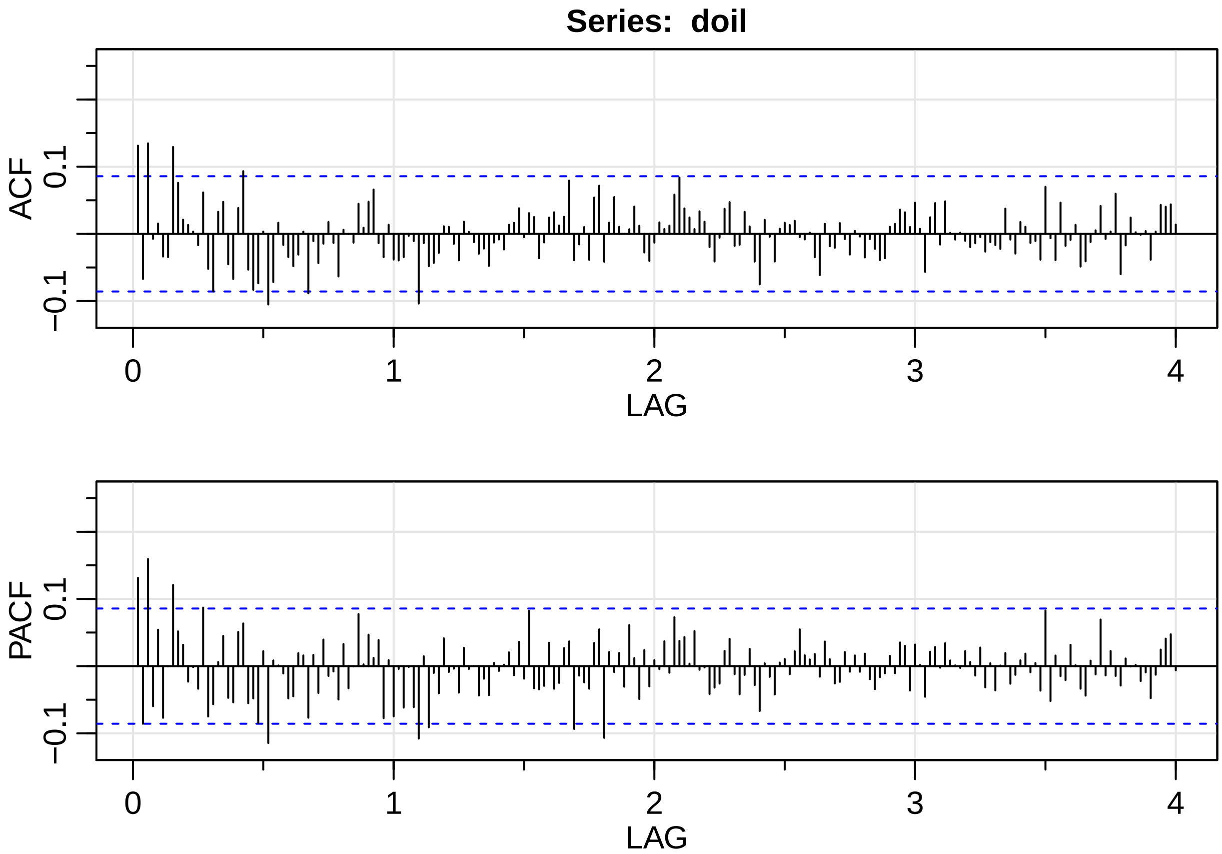
```r
library(tseries)
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
adf.test(doil)
```

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

##
##  Augmented Dickey-Fuller Test
##
## data:  doil
## Dickey-Fuller = -6.3708, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```r
acf2(doil)
```




```
##          ACF  PACF
##   [1,]  0.13  0.13
##   [2,] -0.07 -0.09
##   [3,]  0.13  0.16
##   [4,] -0.01 -0.06
##   [5,]  0.02  0.05
##   [6,] -0.03 -0.08
##   [7,] -0.03  0.00
##   [8,]  0.13  0.12
##   [9,]  0.08  0.05
##  [10,]  0.02  0.03
##  [11,]  0.01 -0.02
##  [12,]  0.00  0.00
##  [13,] -0.02 -0.03
##  [14,]  0.06  0.09
##  [15,] -0.05 -0.07
##  [16,] -0.09 -0.06
##  [17,]  0.03  0.01
##  [18,]  0.05  0.04
##  [19,] -0.05 -0.05
##  [20,] -0.07 -0.05
##  [21,]  0.04  0.05
##  [22,]  0.09  0.06
```

```
##  [23,] -0.05 -0.06
##  [24,] -0.08 -0.05
##  [25,] -0.07 -0.08
##  [26,]  0.00  0.02
##  [27,] -0.11 -0.11
##  [28,] -0.07  0.01
##  [29,]  0.02  0.00
##  [30,] -0.02 -0.01
##  [31,] -0.03 -0.05
##  [32,] -0.05 -0.04
##  [33,] -0.03  0.02
##  [34,]  0.00  0.02
##  [35,] -0.09 -0.08
##  [36,] -0.01  0.02
##  [37,] -0.04 -0.04
##  [38,] -0.01  0.04
##  [39,]  0.02 -0.01
##  [40,] -0.01 -0.01
##  [41,] -0.06 -0.05
##  [42,]  0.01  0.03
##  [43,]  0.00 -0.03
##  [44,] -0.01  0.00
##  [45,]  0.04  0.08
##  [46,]  0.01  0.00
##  [47,]  0.05  0.05
##  [48,]  0.07  0.01
##  [49,] -0.01  0.04
##  [50,] -0.03 -0.08
##  [51,]  0.01  0.01
##  [52,] -0.04 -0.07
##  [53,] -0.04  0.00
##  [54,] -0.03 -0.06
##  [55,]  0.00  0.00
##  [56,] -0.01 -0.06
##  [57,] -0.10 -0.11
##  [58,] -0.01  0.01
##  [59,] -0.05 -0.09
##  [60,] -0.04 -0.01
##  [61,] -0.03 -0.04
##  [62,]  0.01  0.04
##  [63,]  0.01 -0.01
##  [64,] -0.01  0.00
##  [65,] -0.04 -0.04
##  [66,]  0.02  0.03
##  [67,]  0.00  0.00
##  [68,] -0.01  0.00
##  [69,] -0.03 -0.04
##  [70,] -0.02 -0.02
##  [71,] -0.05 -0.04
##  [72,] -0.01  0.00
##  [73,] -0.01 -0.01
##  [74,] -0.02  0.00
##  [75,]  0.01  0.02
##  [76,]  0.02 -0.01
```

```
##  [77,]  0.04  0.04
##  [78,] -0.01 -0.02
##  [79,]  0.03  0.08
##  [80,]  0.02 -0.03
##  [81,] -0.04 -0.03
##  [82,] -0.01 -0.03
##  [83,]  0.02  0.03
##  [84,]  0.03 -0.03
##  [85,]  0.01 -0.02
##  [86,]  0.03  0.03
##  [87,]  0.08  0.04
##  [88,] -0.04 -0.09
##  [89,] -0.02 -0.01
##  [90,]  0.01 -0.02
##  [91,] -0.04 -0.03
##  [92,]  0.05  0.03
##  [93,]  0.07  0.05
##  [94,] -0.04 -0.11
##  [95,]  0.02  0.02
##  [96,]  0.05 -0.01
##  [97,]  0.01  0.02
##  [98,]  0.00 -0.03
##  [99,]  0.01  0.06
## [100,]  0.04  0.01
## [101,]  0.01 -0.05
## [102,] -0.03  0.02
## [103,] -0.04 -0.03
## [104,] -0.01  0.01
## [105,]  0.02  0.00
## [106,]  0.01  0.04
## [107,]  0.01 -0.01
## [108,]  0.06  0.07
## [109,]  0.08  0.04
## [110,]  0.04  0.04
## [111,]  0.02  0.00
## [112,]  0.01  0.05
## [113,]  0.03 -0.01
## [114,]  0.02  0.00
## [115,] -0.02 -0.04
## [116,] -0.04 -0.03
## [117,] -0.01 -0.03
## [118,]  0.04  0.02
## [119,]  0.05  0.04
## [120,] -0.02 -0.01
## [121,] -0.02 -0.04
## [122,]  0.03 -0.01
## [123,]  0.01  0.03
## [124,] -0.04 -0.03
## [125,] -0.08 -0.07
## [126,]  0.02  0.00
## [127,]  0.00 -0.02
## [128,] -0.04 -0.04
## [129,]  0.01  0.01
## [130,]  0.02  0.01
```

```
## [131,]  0.01 -0.01
## [132,]  0.02  0.02
## [133,]  0.00  0.05
## [134,] -0.01  0.02
## [135,]  0.00  0.01
## [136,] -0.03  0.02
## [137,] -0.06 -0.02
## [138,]  0.01  0.04
## [139,] -0.02  0.01
## [140,] -0.02 -0.03
## [141,]  0.02 -0.02
## [142,] -0.01  0.02
## [143,] -0.03 -0.01
## [144,]  0.00  0.02
## [145,]  0.00 -0.01
## [146,] -0.04  0.02
## [147,] -0.01 -0.02
## [148,] -0.02 -0.03
## [149,] -0.04 -0.02
## [150,] -0.04 -0.01
## [151,]  0.01  0.02
## [152,]  0.01 -0.01
## [153,]  0.04  0.04
## [154,]  0.03  0.03
## [155,]  0.01 -0.04
## [156,]  0.05  0.03
## [157,]  0.01  0.00
## [158,] -0.06 -0.05
## [159,]  0.02  0.02
## [160,]  0.05  0.03
## [161,] -0.02  0.00
## [162,]  0.05  0.03
## [163,]  0.00  0.01
## [164,] -0.01  0.00
## [165,]  0.00  0.00
## [166,] -0.01  0.02
## [167,] -0.02  0.01
## [168,] -0.01 -0.01
## [169,]  0.00  0.03
## [170,] -0.03 -0.03
## [171,] -0.01  0.00
## [172,] -0.02 -0.04
## [173,] -0.02  0.00
## [174,]  0.04  0.02
## [175,] -0.01 -0.03
## [176,] -0.03 -0.01
## [177,]  0.02  0.01
## [178,]  0.01  0.02
## [179,] -0.01 -0.01
## [180,] -0.01  0.00
## [181,] -0.04 -0.04
## [182,]  0.07  0.08
## [183,] -0.01 -0.05
## [184,] -0.04  0.02
```

```
## [185,]  0.05 -0.01
## [186,] -0.02 -0.02
## [187,] -0.01  0.03
## [188,]  0.01  0.00
## [189,] -0.05 -0.03
## [190,] -0.04 -0.04
## [191,] -0.01  0.01
## [192,]  0.01 -0.01
## [193,]  0.04  0.07
## [194,] -0.01 -0.01
## [195,]  0.00  0.02
## [196,]  0.06 -0.01
## [197,] -0.06 -0.03
## [198,] -0.02  0.01
## [199,]  0.02  0.00
## [200,]  0.00  0.00
## [201,]  0.00 -0.02
## [202,]  0.00 -0.01
## [203,] -0.04 -0.05
## [204,]  0.00 -0.01
## [205,]  0.04  0.02
## [206,]  0.04  0.04
## [207,]  0.04  0.05
## [208,]  0.01 -0.01
```

```r
library(TSA)
```

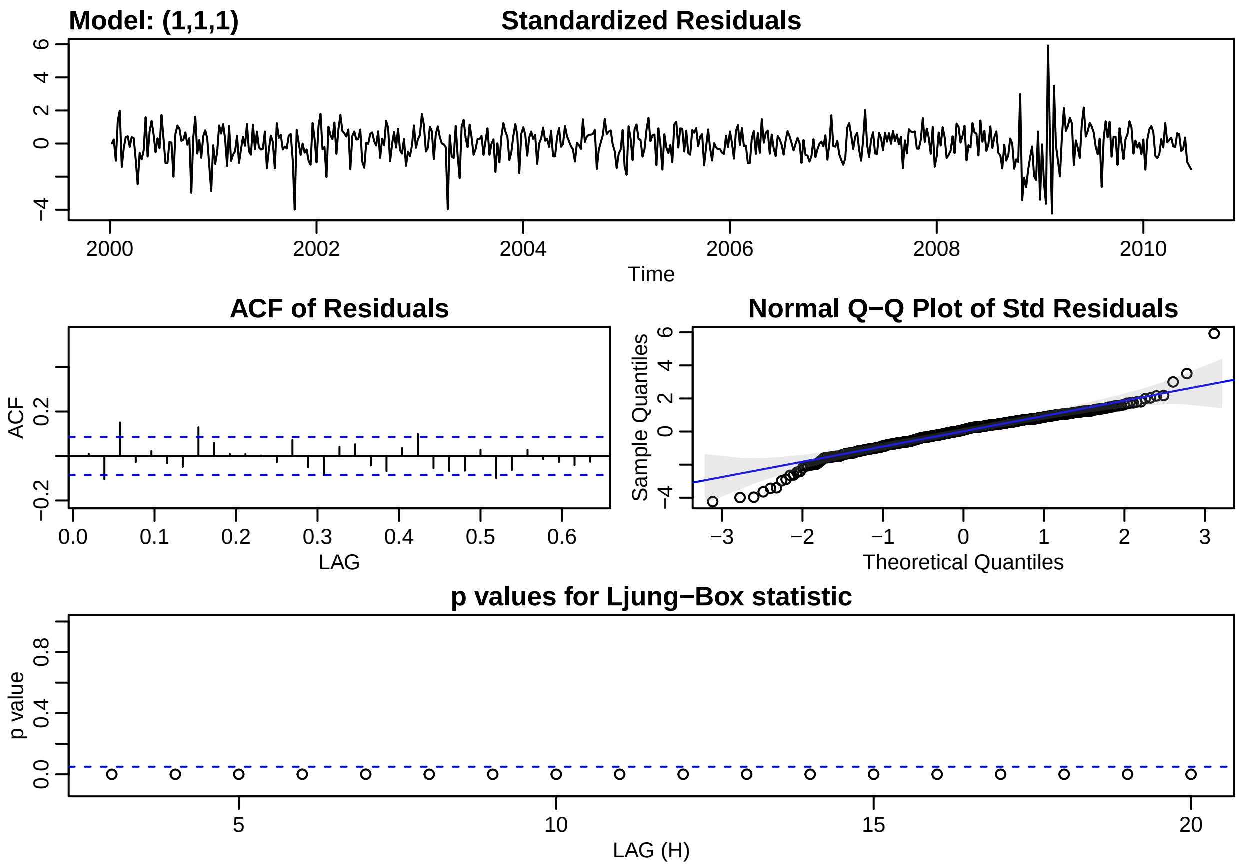
```
##
## Attaching package: 'TSA'

## The following objects are masked from 'package:stats':
##
##     acf, arima

## The following object is masked from 'package:utils':
##
##     tar
```

```r
eacf(doil)
```

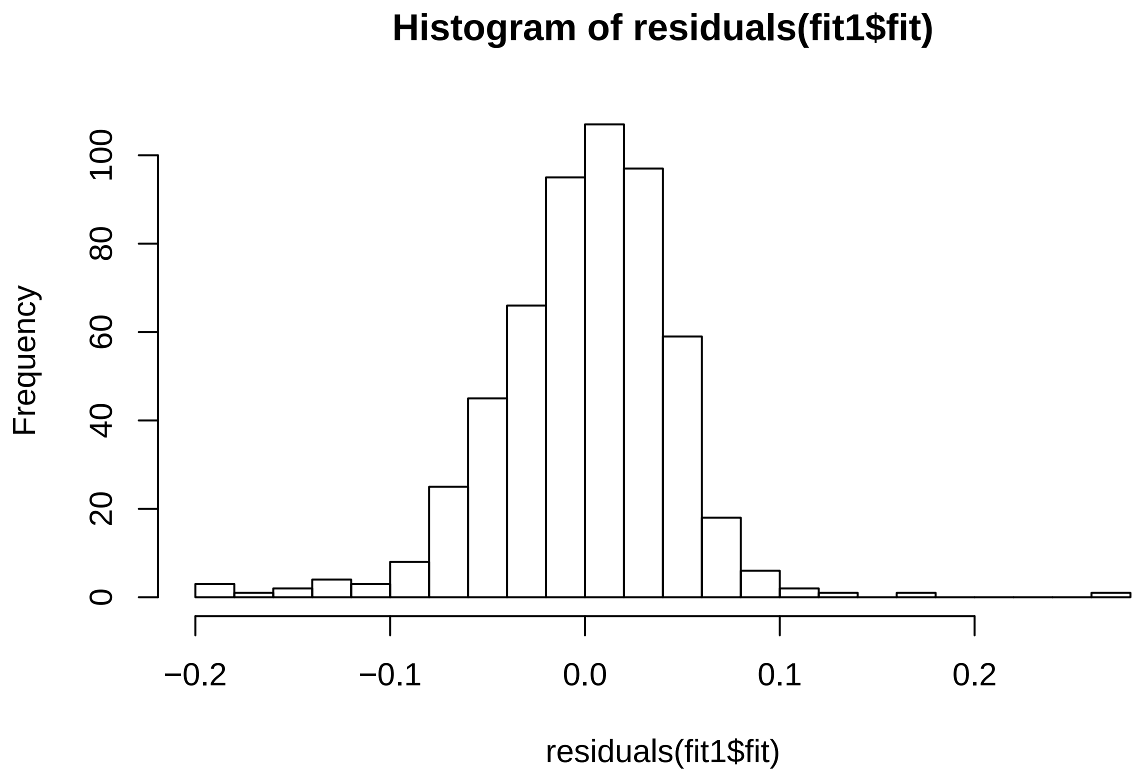
```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x o x o o o o x o o o  o  o  o
## 1 x o x o o o o x o o o  o  o  o
## 2 x x x o o o o x o o o  o  o  o
## 3 x x x o o o o x o o o  o  o  o
## 4 x o x o o o o x o o o  o  o  o
## 5 x x x o x o o x o o o  o  o  o
## 6 o x x o x x o x o o o  o  o  x
## 7 o x x x x x x x o x o  o  o  o
```

```r
# ARMA(1,1)
fit1 <- sarima(doil, p = 1, d = 1, q = 1)
```

```
## initial  value -2.782182
## iter   2 value -2.918634
## iter   3 value -2.957668
## iter   4 value -2.980235
## iter   5 value -3.018742
## iter   6 value -3.031249
## iter   7 value -3.054383
## iter   8 value -3.054928
## iter   9 value -3.057379
## iter  10 value -3.060053
## iter  11 value -3.061128
## iter  12 value -3.061129
## iter  13 value -3.061252
## iter  14 value -3.061366
## iter  15 value -3.061366
## iter  15 value -3.061366
## iter  16 value -3.061368
## iter  16 value -3.061368
## iter  16 value -3.061368
## final  value -3.061368
## converged
## initial  value -3.058586
## iter   2 value -3.059184
## iter   3 value -3.059381
## iter   4 value -3.059818
## iter   5 value -3.059990
## iter   6 value -3.060281
## iter   7 value -3.060537
## iter   8 value -3.060797
## iter   9 value -3.060823
## iter  10 value -3.060826
## iter  11 value -3.060826
## iter  11 value -3.060826
## final  value -3.060826
## converged
```




```r
hist(residuals(fit1$fit), breaks = 25)
```



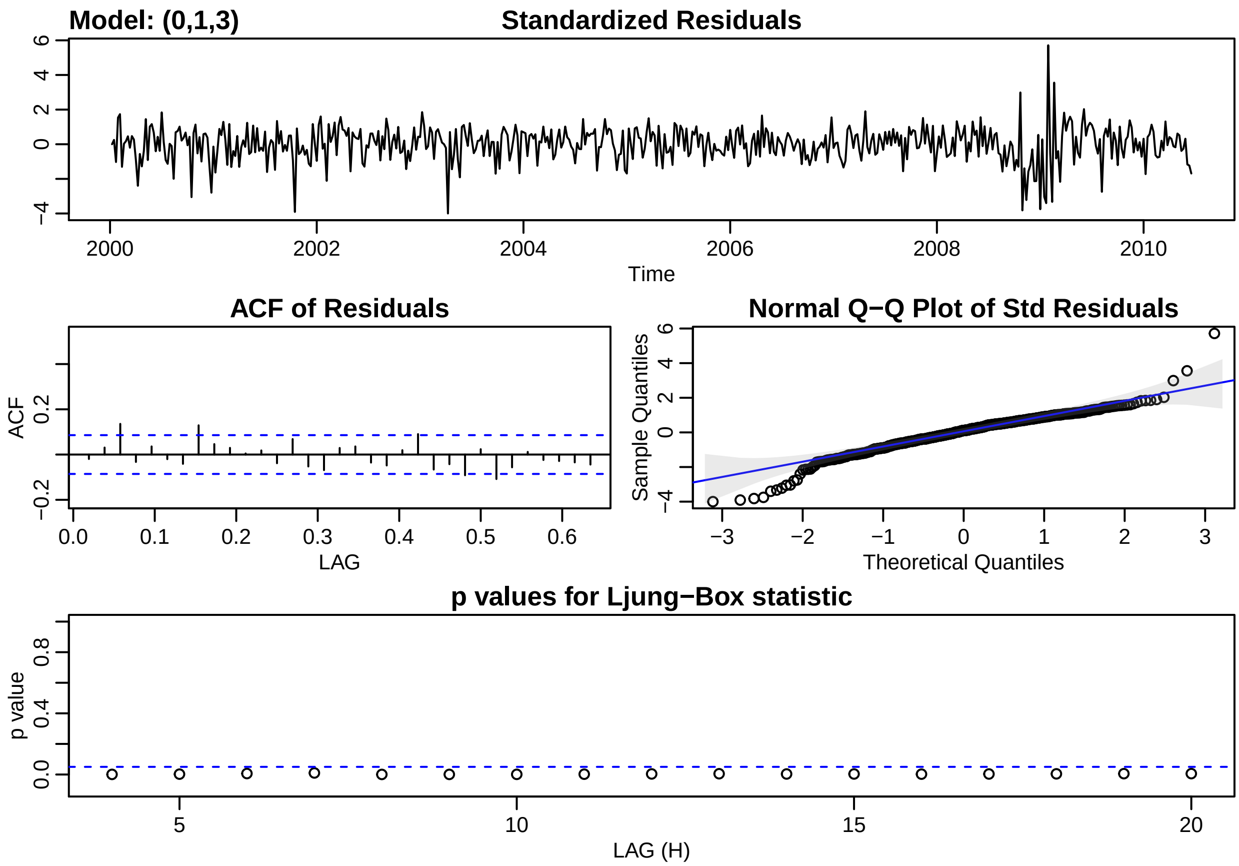

```r
runs(residuals(fit1$fit))
```

```
## $pvalue
## [1] 0.728
##
## $observed.runs
## [1] 267
##
## $expected.runs
## [1] 271.5294
##
## $n1
## [1] 252
##
## $n2
## [1] 292
##
## $k
## [1] 0
```

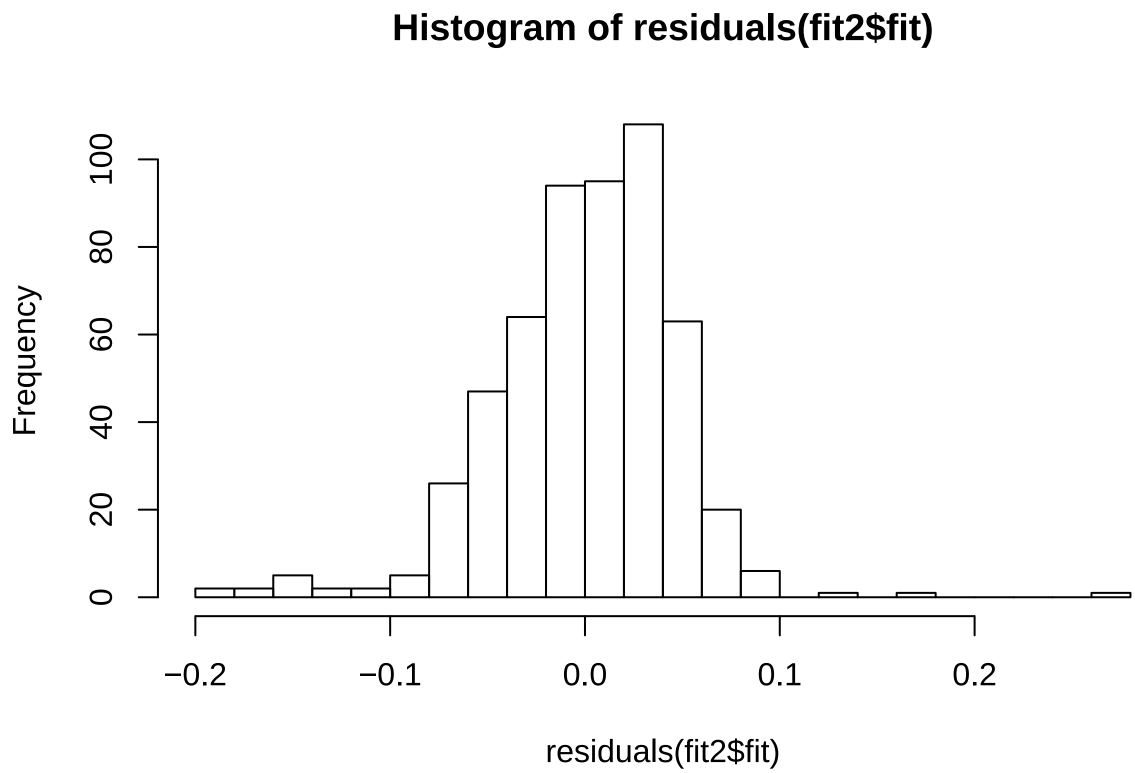
```r
# ARMA(0,3)
fit2 <- sarima(doil, p = 0, d = 1, q = 3)
```

```
## initial  value -2.783046
## iter   2 value -2.960447
```

```
## iter   3 value -3.015376
## iter   4 value -3.041409
## iter   5 value -3.048865
## iter   6 value -3.052819
## iter   7 value -3.063057
## iter   8 value -3.069428
## iter   9 value -3.069559
## iter  10 value -3.070886
## iter  11 value -3.071194
## iter  12 value -3.071534
## iter  13 value -3.072032
## iter  14 value -3.072150
## iter  15 value -3.073159
## iter  16 value -3.073173
## iter  17 value -3.073185
## iter  17 value -3.073185
## iter  17 value -3.073185
## final  value -3.073185
## converged
## initial  value -3.067962
## iter   2 value -3.069658
## iter   3 value -3.069697
## iter   4 value -3.070292
## iter   5 value -3.070402
## iter   6 value -3.070424
## iter   7 value -3.070441
## iter   8 value -3.070442
## iter   8 value -3.070442
## iter   8 value -3.070442
## final  value -3.070442
## converged
```




```
hist(residuals(fit2$fit), breaks = 25)
```

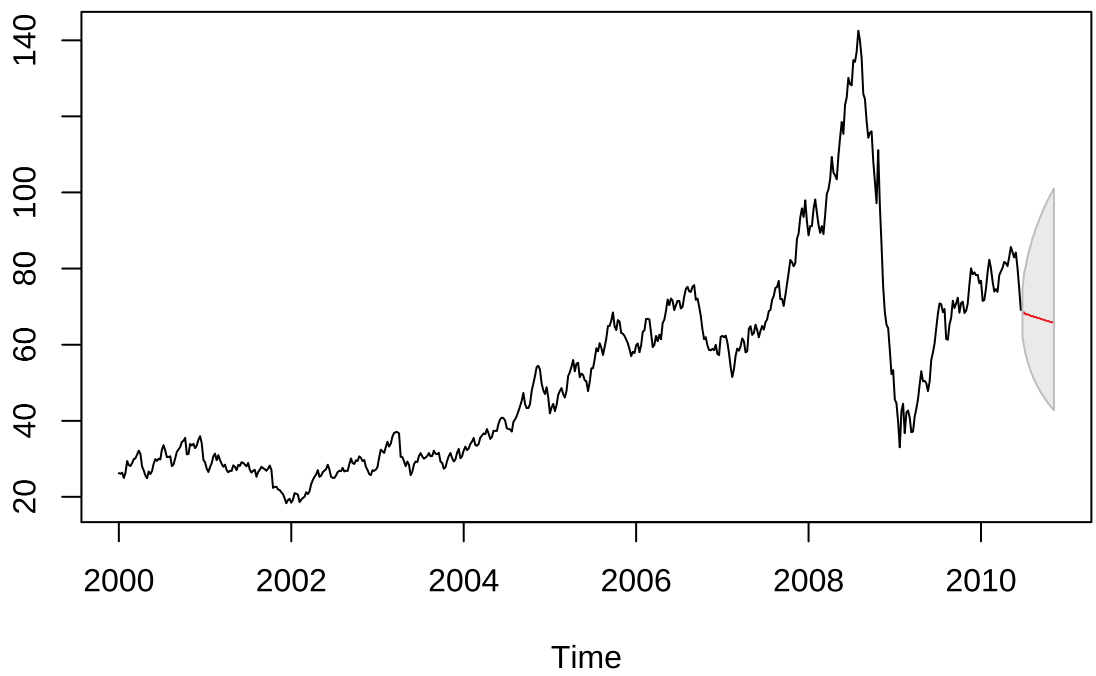



```r
runs(residuals(fit2$fit))
```

```
## $pvalue
## [1] 0.414
##
## $observed.runs
## [1] 281
##
## $expected.runs
## [1] 271.0551
##
## $n1
## [1] 249
##
## $n2
## [1] 295
##
## $k
## [1] 0
```

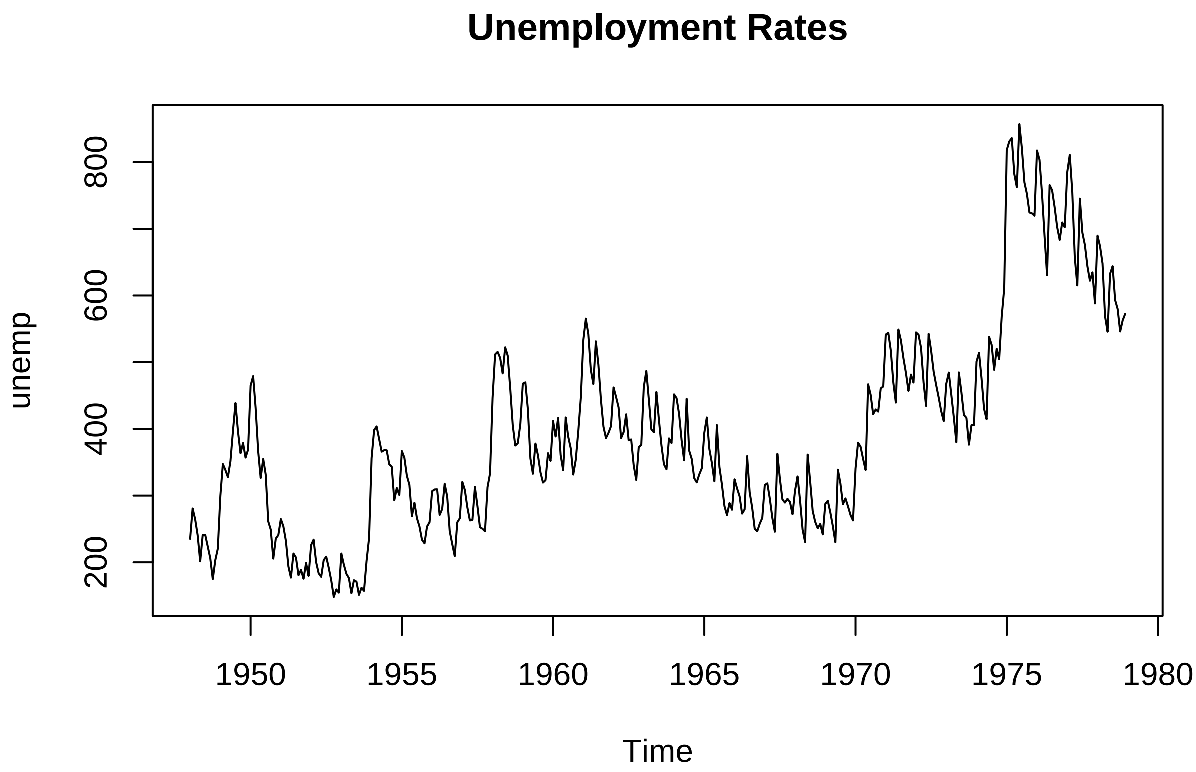
```r
logoil <- arima(log(oil), order = c(3,0,3))
prediction <- predict(logoil, n.ahead=20)
```

```r
ts.plot(oil, exp(prediction$pred), col=c(1:2))

U <- exp(prediction$pred + 2*prediction$se)
L <- exp(prediction$pred - 2*prediction$se)

xx = c(time(U), rev(time(U)));  yy = c(L, rev(U))
polygon(xx, yy, border = 8, col = gray(.6, alpha = .2))
lines(prediction$pred, type="p", col=2)
```

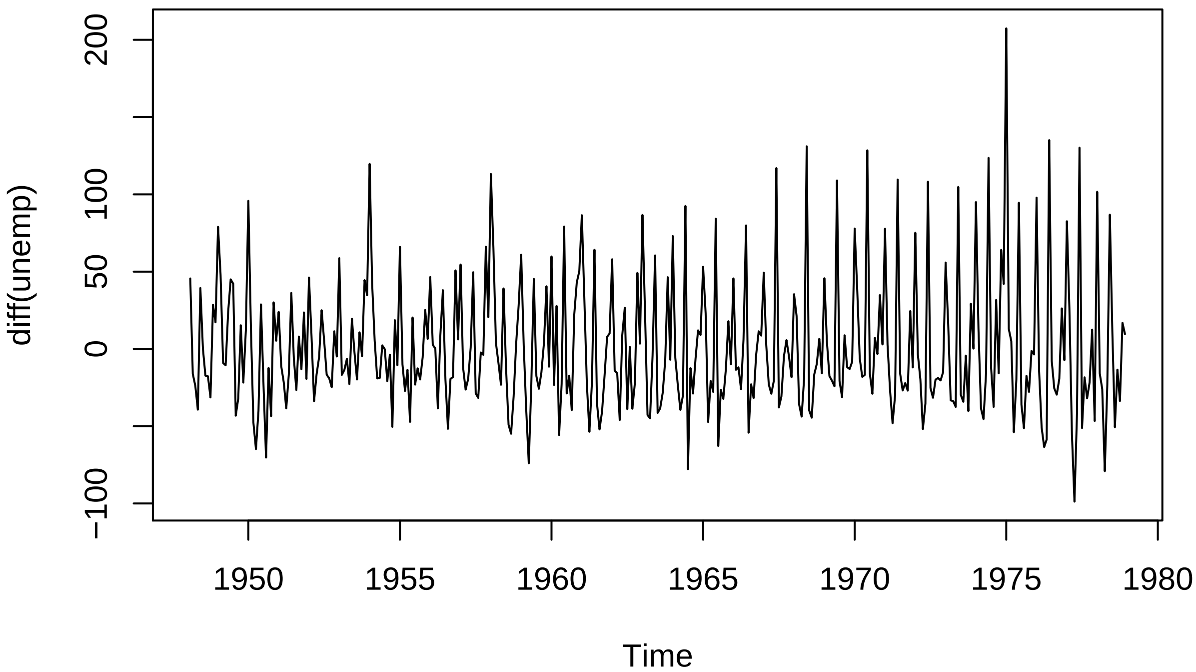


b

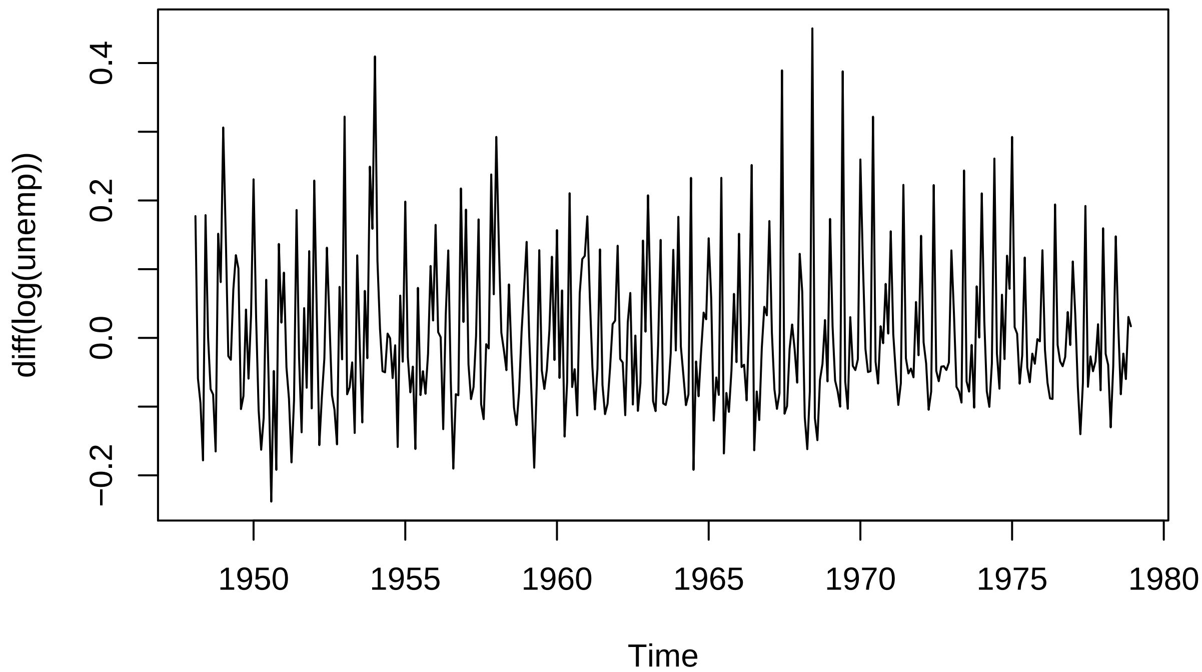
```r
ts.plot(unemp)
title(main = "Unemployment Rates")
```




```r
# Non-stationary, so use differencing:
ts.plot(diff(unemp))
```




```r
# Variance is not constant everywhere, so:
ts.plot(diff(log(unemp)))
```




```r
dunemp <- diff(log(unemp))
```

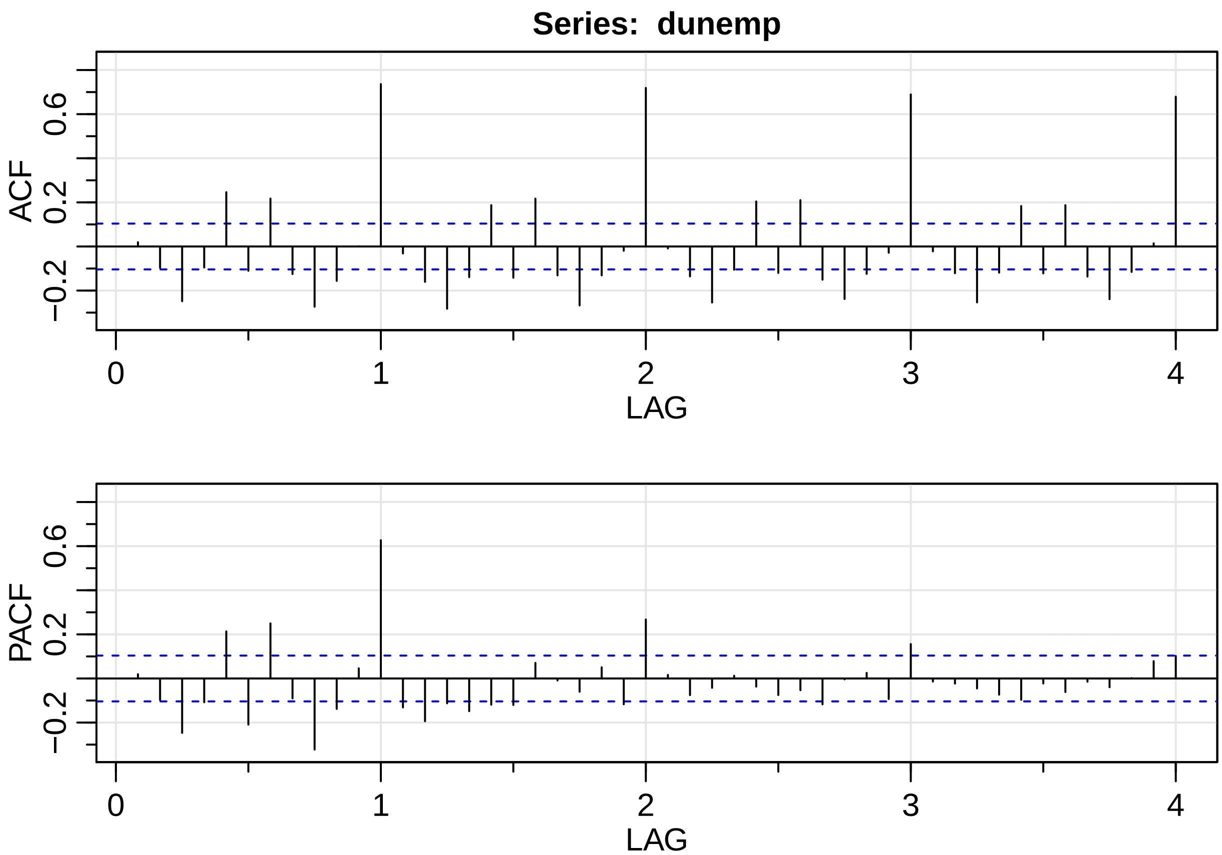
```r
library(tseries)
```

```r
adf.test(dunemp)
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  dunemp
## Dickey-Fuller = -6.9076, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```r
acf2(dunemp)
```




```
##         ACF  PACF
##  [1,]  0.02  0.02
##  [2,] -0.10 -0.10
##  [3,] -0.25 -0.25
##  [4,] -0.10 -0.11
##  [5,]  0.25  0.21
##  [6,] -0.11 -0.21
##  [7,]  0.22  0.25
##  [8,] -0.12 -0.09
##  [9,] -0.27 -0.32
## [10,] -0.16 -0.14
## [11,]  0.00  0.05
## [12,]  0.74  0.63
## [13,] -0.03 -0.13
## [14,] -0.16 -0.19
## [15,] -0.28 -0.11
## [16,] -0.14 -0.15
## [17,]  0.19 -0.12
## [18,] -0.14 -0.12
## [19,]  0.22  0.07
## [20,] -0.13 -0.01
## [21,] -0.27 -0.06
## [22,] -0.13  0.05
```

```
## [23,] -0.02 -0.12
## [24,]  0.72  0.27
## [25,] -0.01  0.02
## [26,] -0.14 -0.08
## [27,] -0.25 -0.04
## [28,] -0.11  0.01
## [29,]  0.20 -0.04
## [30,] -0.12 -0.08
## [31,]  0.21 -0.05
## [32,] -0.15 -0.12
## [33,] -0.24  0.00
## [34,] -0.12  0.03
## [35,] -0.03 -0.09
## [36,]  0.69  0.16
## [37,] -0.02 -0.01
## [38,] -0.12 -0.02
## [39,] -0.25 -0.05
## [40,] -0.12 -0.07
## [41,]  0.18 -0.10
## [42,] -0.12 -0.02
## [43,]  0.19 -0.06
## [44,] -0.14 -0.01
## [45,] -0.24 -0.04
## [46,] -0.12  0.00
## [47,]  0.01  0.08
## [48,]  0.68  0.10
```

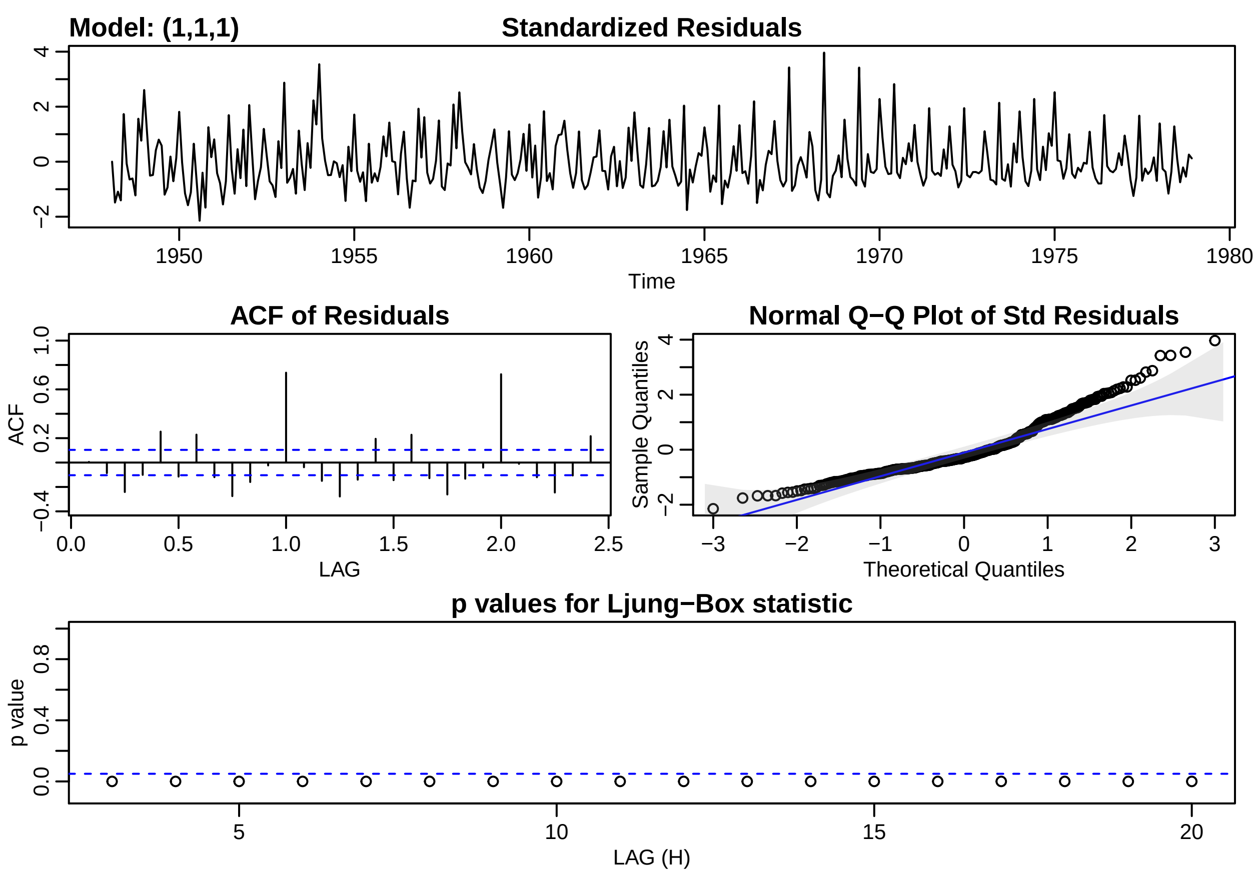
```r
library(TSA)
eacf(dunemp)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 o o x o x x x x x x o  x  o  x
## 1 x o x o x o x x x x o  x  x  o
## 2 x x o x x o o x o o o  x  x  x
## 3 x x x x x o o x o o o  x  x  x
## 4 x x o x o x o o o o o  x  o  x
## 5 x x o x o x o o o o o  x  o  x
## 6 x x x o o x o o o o o  x  o  x
## 7 x x o x x o x o o o o  x  o  o
```

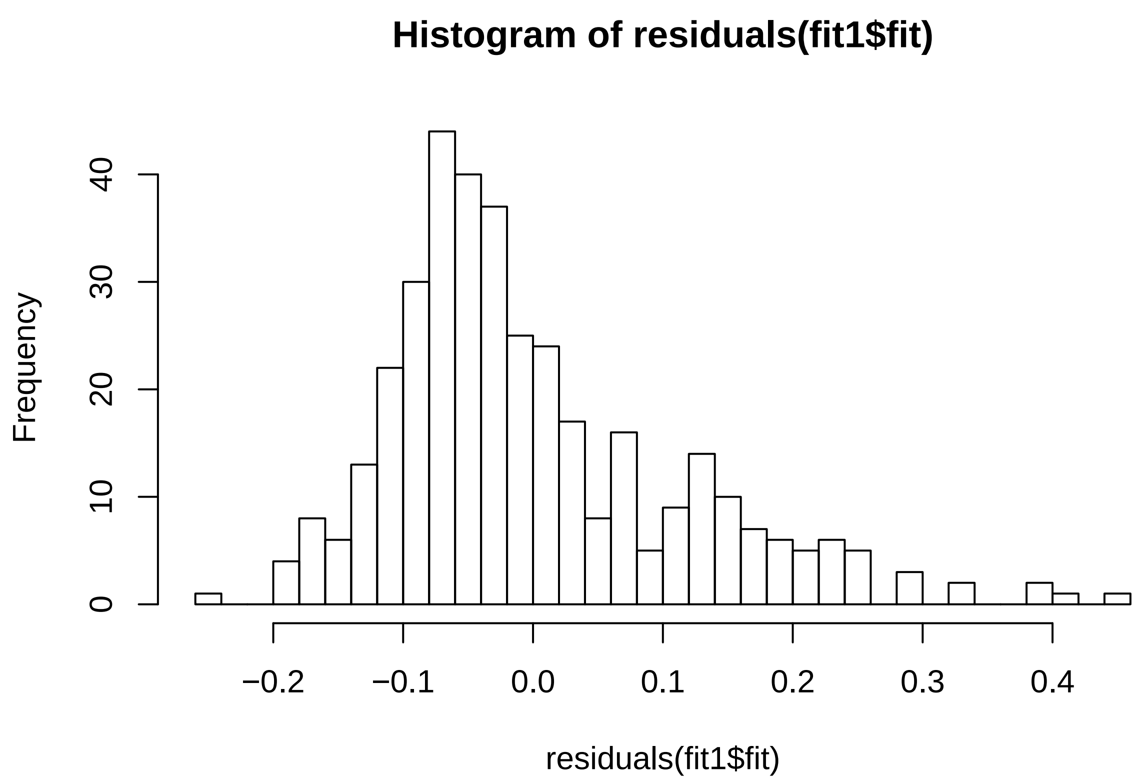
```r
# ARMA(1,1)
fit1 <- sarima(dunemp, p = 1, d = 1, q = 1)
```

```
## initial  value -1.839836
## iter   2 value -1.977326
## iter   3 value -2.026912
## iter   4 value -2.094279
## iter   5 value -2.146078
## iter   6 value -2.146575
## iter   7 value -2.147047
## iter   8 value -2.148328
## iter   9 value -2.148459
```

```
## iter  10 value -2.149790
## iter  11 value -2.156447
## iter  12 value -2.158729
## iter  13 value -2.158771
## iter  14 value -2.158776
## iter  15 value -2.158776
## iter  15 value -2.158776
## final  value -2.158776
## converged
## initial  value -2.158913
## iter   2 value -2.159636
## iter   3 value -2.162406
## iter   4 value -2.164650
## iter   5 value -2.165463
## iter   6 value -2.165518
## iter   7 value -2.165653
## iter   8 value -2.165653
## iter   8 value -2.165653
## final  value -2.165653
## converged
```



```r
hist(residuals(fit1$fit), breaks = 25)
```



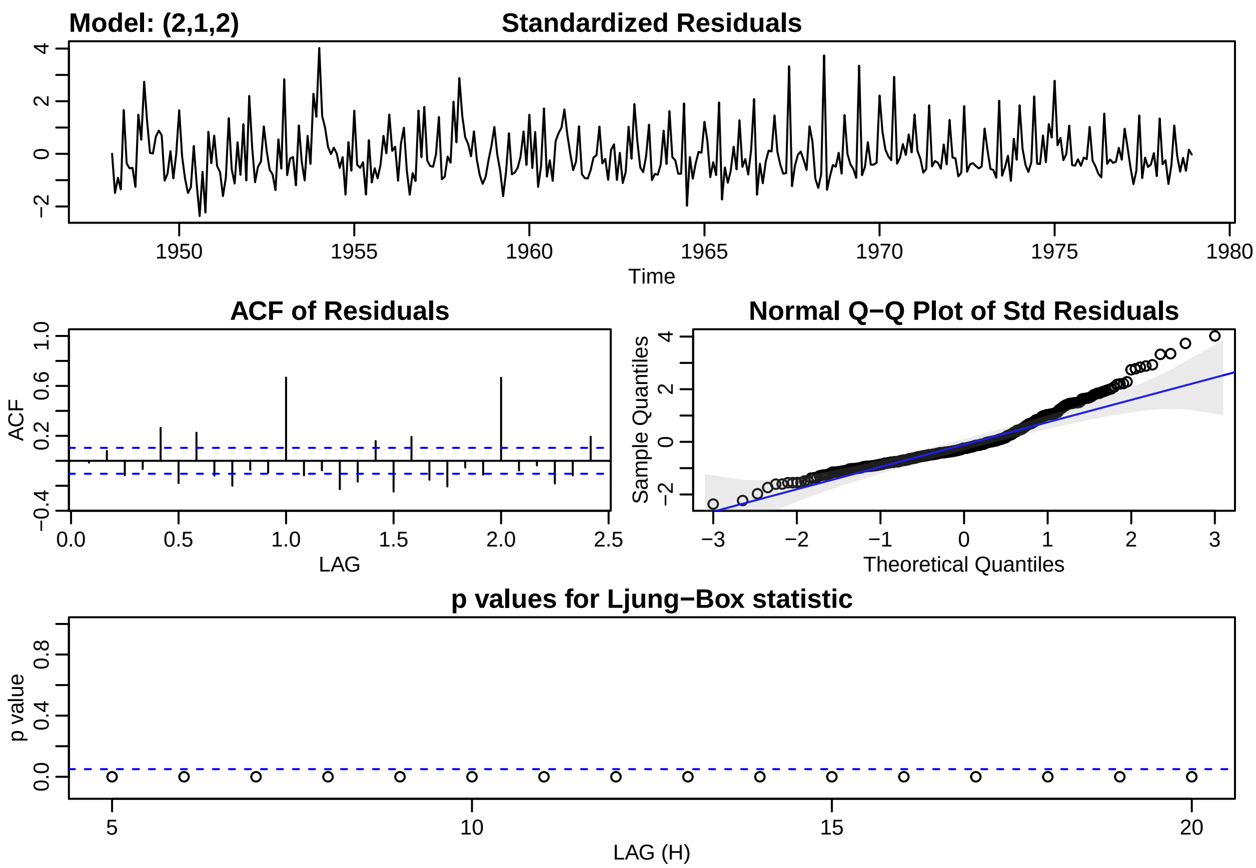

```r
runs(residuals(fit1$fit))
```

```
## $pvalue
## [1] 0.256
##
## $observed.runs
## [1] 165
##
## $expected.runs
## [1] 175.8248
##
## $n1
## [1] 230
##
## $n2
## [1] 141
##
## $k
## [1] 0
```

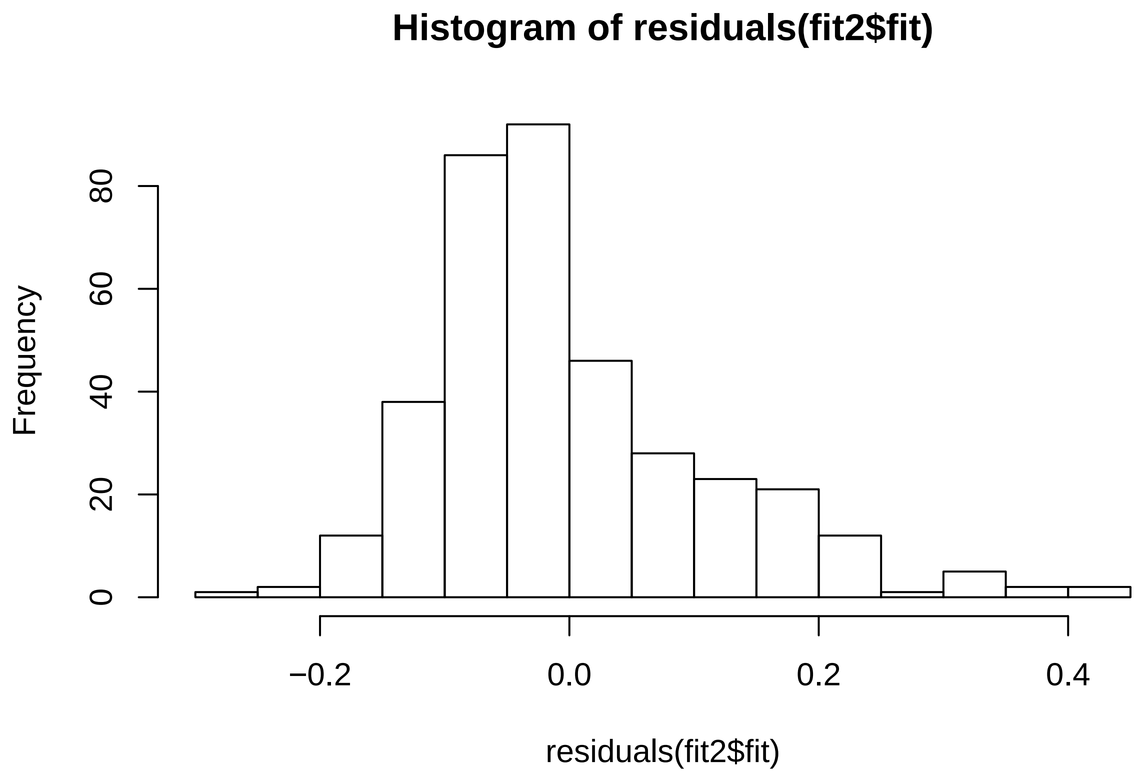
```r
# ARMA(2,2)
fit2 <- sarima(dunemp, p = 2, d = 1, q = 2)
```

```
## initial  value -1.838545
## iter   2 value -1.962865
## iter   3 value -2.019446
## iter   4 value -2.064444
## iter   5 value -2.082272
## iter   6 value -2.101401
## iter   7 value -2.117105
## iter   8 value -2.120891
## iter   9 value -2.135749
## iter  10 value -2.147752
## iter  11 value -2.150291
## iter  12 value -2.151408
## iter  13 value -2.151587
## iter  14 value -2.152259
## iter  15 value -2.152378
## iter  16 value -2.152525
## iter  17 value -2.152583
## iter  18 value -2.153335
## iter  19 value -2.154567
## iter  20 value -2.155686
## iter  21 value -2.157122
## iter  22 value -2.159875
## iter  23 value -2.165731
## iter  24 value -2.165877
## iter  25 value -2.166208
## iter  26 value -2.177620
## iter  27 value -2.180805
## iter  28 value -2.181066
## iter  29 value -2.181837
## iter  30 value -2.183668
## iter  31 value -2.183715
## iter  32 value -2.185996
## iter  33 value -2.187753
## iter  34 value -2.188297
## iter  35 value -2.188444
## iter  36 value -2.190219
## iter  37 value -2.191224
## iter  38 value -2.191229
## iter  39 value -2.192155
## iter  39 value -2.192155
## iter  40 value -2.193056
## iter  40 value -2.193056
## iter  41 value -2.193365
## iter  42 value -2.193504
## iter  42 value -2.193504
## iter  43 value -2.193566
## iter  43 value -2.193566
## iter  44 value -2.193574
## iter  44 value -2.193574
## iter  45 value -2.193581
## iter  45 value -2.193581
## iter  46 value -2.193582
## iter  46 value -2.193582
## iter  46 value -2.193582
```

```
## final  value -2.193582
## converged
## initial  value -2.176035
## iter   2 value -2.176227
## iter   3 value -2.177282
## iter   4 value -2.185940
## iter   5 value -2.186046
## iter   6 value -2.186121
## iter   7 value -2.186123
## iter   8 value -2.186124
## iter   9 value -2.186125
## iter  10 value -2.186128
## iter  11 value -2.186131
## iter  12 value -2.186133
## iter  12 value -2.186133
## final  value -2.186133
## converged
```




```r
hist(residuals(fit2$fit), breaks = 25)
```



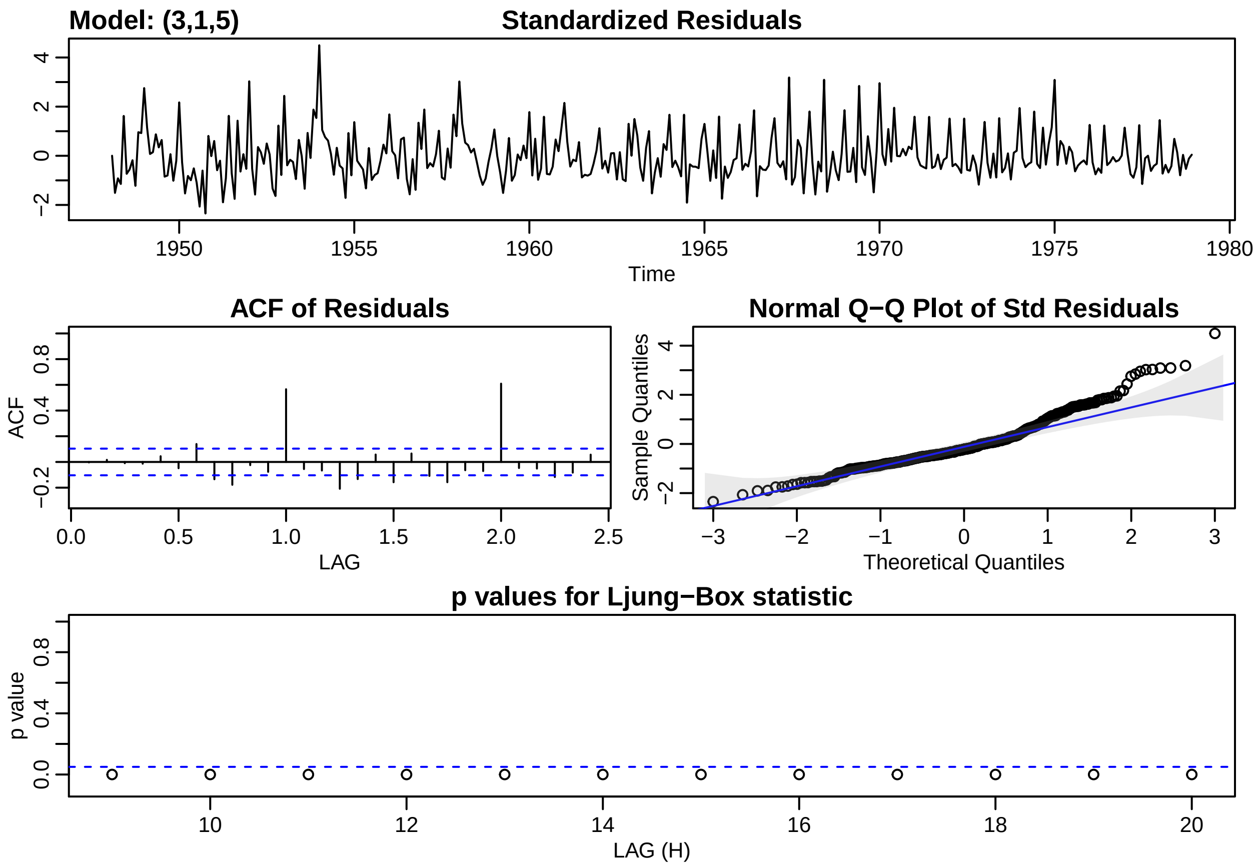

```r
runs(residuals(fit2$fit))
```

```
## $pvalue
## [1] 0.589
##
## $observed.runs
## [1] 170
##
## $expected.runs
## [1] 175.3396
##
## $n1
## [1] 231
##
## $n2
## [1] 140
##
## $k
## [1] 0
```

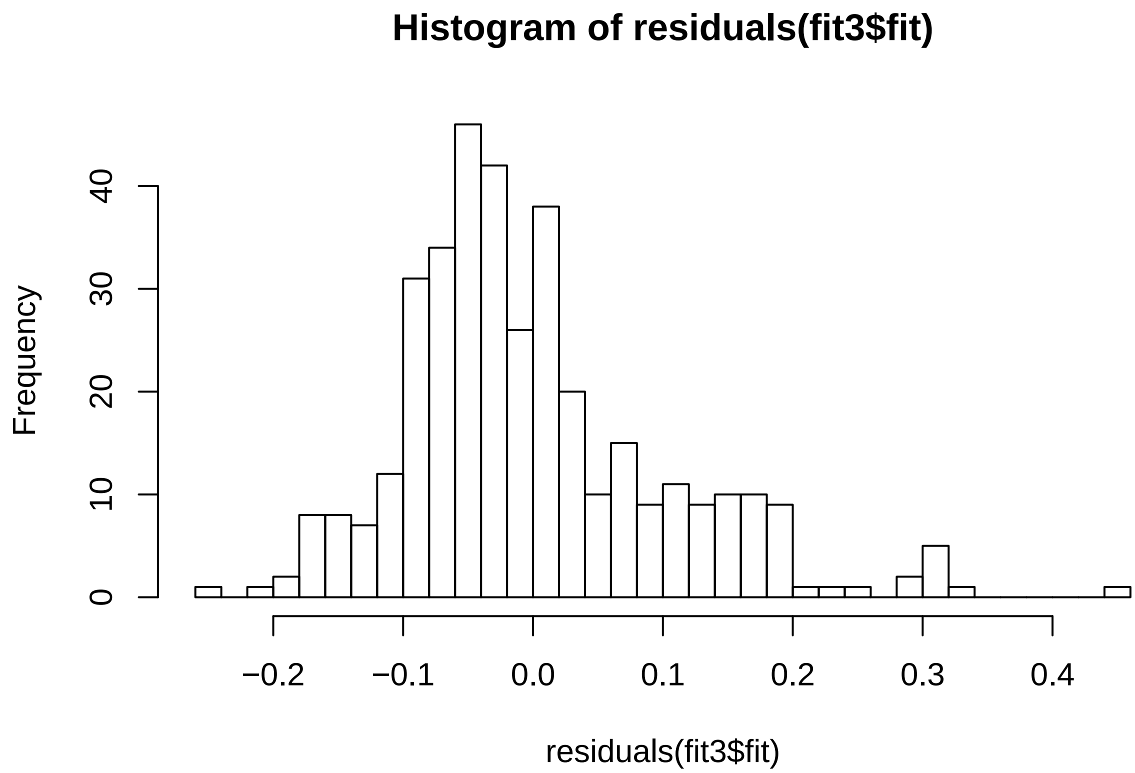
```r
# ARMA (3,5)
fit3 <- sarima(dunemp, p = 3, d = 1, q = 5)
```

```
## initial  value -1.837556
## iter   2 value -2.020464
```

```
## iter   3 value -2.089573
## iter   4 value -2.129445
## iter   5 value -2.131421
## iter   6 value -2.166812
## iter   7 value -2.185142
## iter   8 value -2.203775
## iter   9 value -2.218709
## iter  10 value -2.231746
## iter  11 value -2.235788
## iter  12 value -2.240680
## iter  13 value -2.245600
## iter  14 value -2.248597
## iter  15 value -2.249462
## iter  16 value -2.250063
## iter  17 value -2.250466
## iter  18 value -2.250656
## iter  19 value -2.250694
## iter  20 value -2.250699
## iter  20 value -2.250700
## final  value -2.250700
## converged
## initial  value -2.256625
## iter   2 value -2.257321
## iter   3 value -2.259743
## iter   4 value -2.261883
## iter   5 value -2.262252
## iter   6 value -2.266218
## iter   7 value -2.266866
## iter   8 value -2.267973
## iter   9 value -2.268396
## iter  10 value -2.269035
## iter  11 value -2.269462
## iter  12 value -2.269565
## iter  13 value -2.269620
## iter  14 value -2.269678
## iter  15 value -2.269693
## iter  16 value -2.269699
## iter  17 value -2.269702
## iter  18 value -2.269702
## iter  19 value -2.269702
## iter  19 value -2.269702
## iter  19 value -2.269702
## final  value -2.269702
## converged
```




```
hist(residuals(fit3$fit), breaks = 25)
```




```r
runs(residuals(fit3$fit))
```

```
## $pvalue
## [1] 0.434
##
## $observed.runs
## [1] 173
##
## $expected.runs
## [1] 180.8059
##
## $n1
## [1] 218
##
## $n2
## [1] 153
##
## $k
## [1] 0
```

```r
# ARMA (3,5)
logunemp <- arima(log(unemp), order = c(3,0,5))
prediction <- predict(logunemp, n.ahead=20)
```

```r
ts.plot(unemp, exp(prediction$pred), col=c(1:2))

U <- exp(prediction$pred + 2*prediction$se)
L <- exp(prediction$pred - 2*prediction$se)

xx = c(time(U), rev(time(U)));  yy = c(L, rev(U))
polygon(xx, yy, border = 8, col = gray(.6, alpha = .2))
lines(prediction$pred, type="p", col=2)
```