Time Series - Computer lab B

group 6

9/23/2019

Contents

Assignment 1. Computations with simulated data

(a) AR(3) series

The observations are from an AR(3) process

$$x_t = 0.8x_{t-1} - 0.2x_{t-2} + 0.1x_{t-3} + \epsilon_t,$$

so there might exist some linear relationships among the nearest 2 observations. We assume two potential linear relationships

$$x_0 \sim x_1 + x_2 \quad and \quad x_3 \sim x_2 + x_1,$$

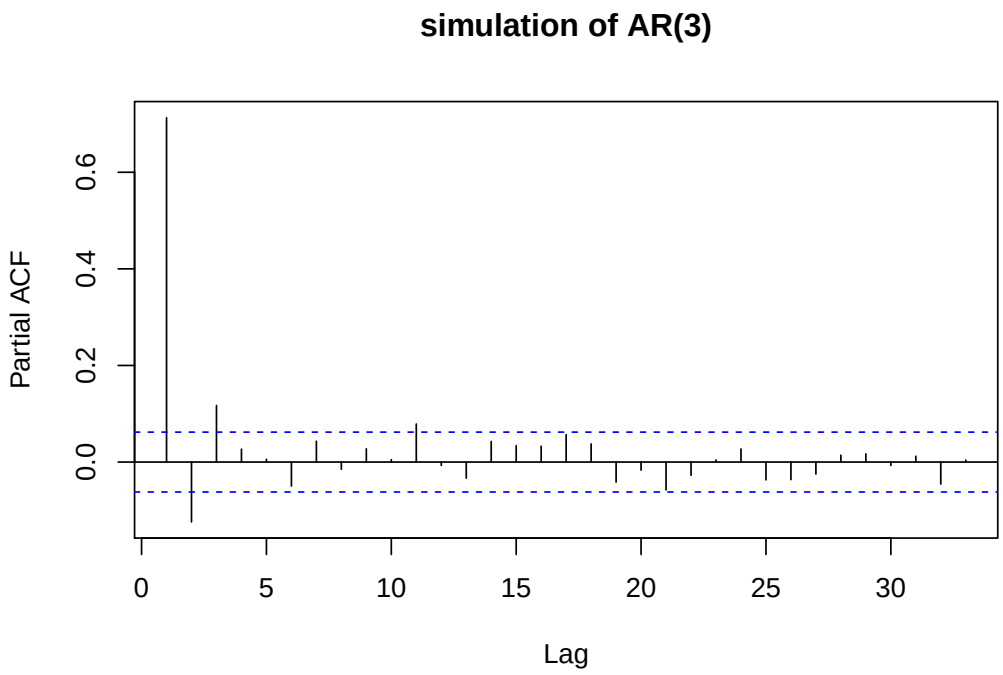
and then calculate the correlation between the residuals of such two linear models.

```
rm(list=ls())
set.seed(12345)
# the simulation of ar(3) process
sim <- arima.sim(list(ar=c(0.8,-0.2,0.1)), n=1000)
# create lag data.frame
dat <-ts.intersect(x0=sim, x1=lag(sim,-1), x2=lag(sim,-2),
                   x3=lag(sim,-3), dframe = T)
# create the potiential linear relation among nearby observations
res1 <- lm(x0~x1+x2, data=dat)
res2 <- lm(x3~x2+x1, data=dat)
# the estimation of noise
r1 <- residuals(res1)
r2 <- residuals(res2)
cor(r1, r2)
```

```
## [1] 0.1146076
```

The correlation between such residuals of linear regressions (or noise of estimations) is quite low and close to 0, which means such two residuals are nearly independent with each other. This must be true since the original time series *sim* is from an AR(3) process, whose ϵ_t are i.i.d Gaussion distributed for any t.

```
pacf(sim, lag.max = 33, main = "simulation of AR(3)")
```



The partial ACF decays (cuts off) around $lag = 3$, which is the same as the definition with AR(3) process. It means that there is a positive autocorrelation among our simulation until $lag = 3$ given by 3 lagged series. In acccordance with the partial ACF plot, AR parameter ϕ_{33} is invisble (close to 0), which means that series x_{t-33} has no effect to x_t if we remove all the relation of variables between x_{t-33} and x_t.

```
ARMAtoAR(c(0.8,-0.2,0.1), lag.max = 33)[33]
```

```
## [1] 0
```

The parameter ϕ_{33} should equal 0 theoretically, which is the same as our conclusion from partial ACF plot.

(b) AR(2) series

```
# Simulate an AR(2) process
set.seed(12345)
ar_sim2 = arima.sim(model = list(ar = c(0.8, 0.1)), n = 100)

# method of moments (Yule-Walker equations)
yw = ar(ar_sim2, order = 2,
        method = c("yule-walker"),
        aic = FALSE)

# - conditional least squares
cls = ar(ar_sim2, order = 2,
        method = c("ols"),
        aic = FALSE)
# - maximum likelihood (MLE)
ml = ar(ar_sim2, order = 2,
        method = c("mle"),
        aic = FALSE)
```

```
## Warning in arima0(x, order = c(order, 0L, 0L), include.mean = demean):
## possible convergence problem: optim gave code = 1
```

```
# compare results to the true values
# - true values are given
matrix_a1_b = matrix(c(0.8,yw$ar[1],cls$ar[1],ml$ar[1],
                       0.1,yw$ar[2],cls$ar[2],ml$ar[2]),
                       ncol = 2, nrow = 4)

df_a1_b = as.data.frame(matrix_a1_b,
                          row.names = c("True",
                                        "Yule-Walker",
                                        "conditional least squares",
                                        "maximum likelihood "),
                          col.names = c("phi1", "phi2"))
df_a1_b = round(df_a1_b,3)
```

	V1	V2
True	0.800	0.100
Yule-Walker	0.803	0.104
conditional least squares	0.807	0.121
maximum likelihood	0.797	0.119

The best method is Yule-Walker method. It estiamtes the closest parameters to the true values. Afterward, to check whether the real value ϕ_2 falls in the confidence interval for ML estimate, we need to compute the interval firstly. The asymptotic-theory variance matrix of the coefficient ML estimates is

	V1	V2
phi1	0.0090727	-0.0081274
phi2	-0.0081274	0.0090727

We have the variance σ^2 of ϕ_2 is

```
## [1] 0.009072681
```

so its standard deviation is

```
## [1] 0.09525062
```

If we choose the confidence level is $a = 95$, the confidence interval would be

$$Z_{a/2} * \frac{\sigma}{\sqrt{n}} \approx 1.96 * \frac{0.09525062}{\sqrt{100}} = 0.01866912,$$

and the confidence interval of ϕ_2 from MLE is

```
## 0.1002677 0.137606
```

Thus, the theoretical value for $\phi_2 = 0.1$ does not fall within confidence interval for ML estimate.

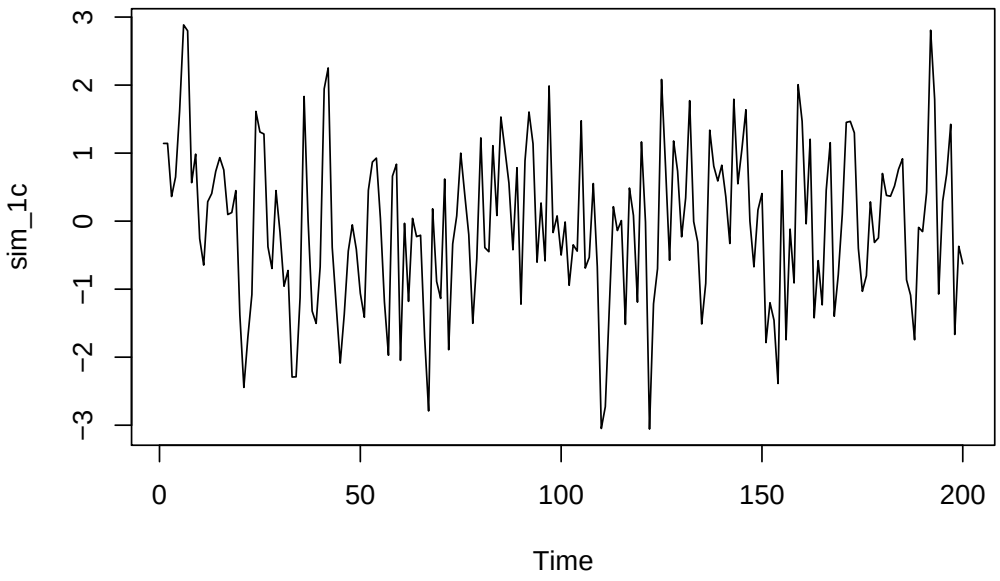
(c) Seasonal $ARIMA(0,0,1) \times (0,0,1)_1 2$

The time series we need follows the seasonal MA model $ARIMA(0,0,1) \times (0,0,1)_{12}$. We can be rewritten it as a normal MA model $MA(13)$ as

$$(1-\theta_1 B)(1-\Theta_1 B^{12})w_t = (1-\theta_1 B - \Theta_1 B^{12} + \theta_1\Theta_1 B^{13})w_t \quad ,$$

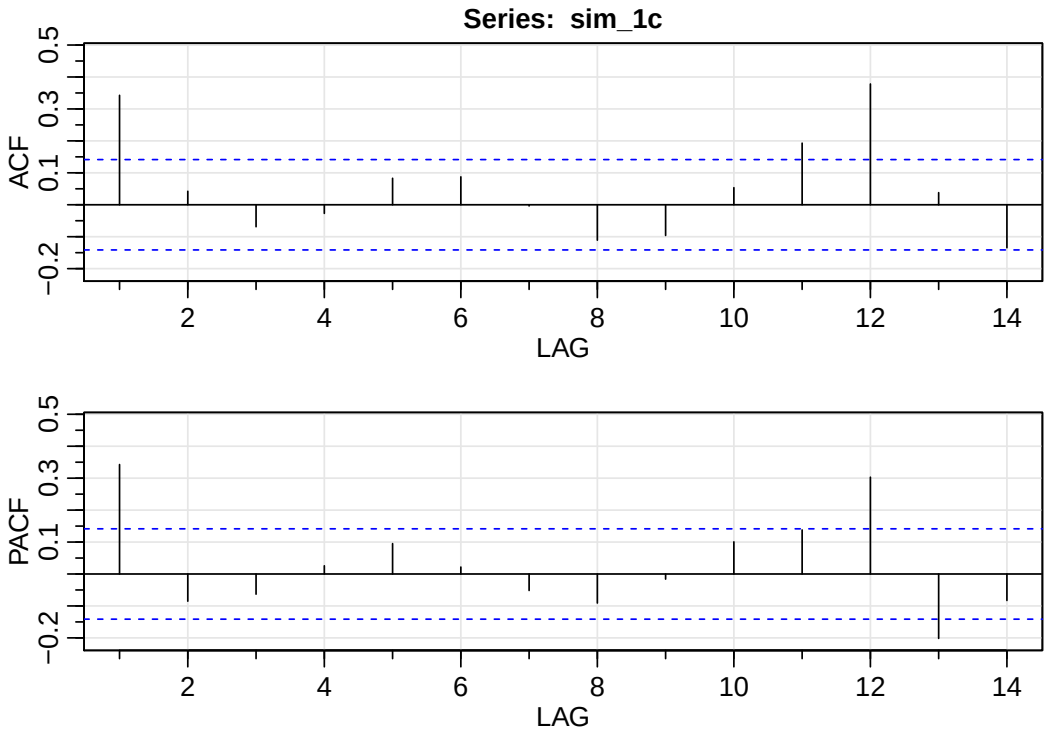
and then simulate it by `arima.sim` function.

```r
set.seed(123456)
# generate seasonal ARIMA(0,0,1)*(0,0,1)_12
sim_1c <- arima.sim(list(order = c(0,0,13), ma = c(0.3,rep(0,10),0.6, 0.3*0.6)), n = 200)
plot(sim_1c, type="l")
```

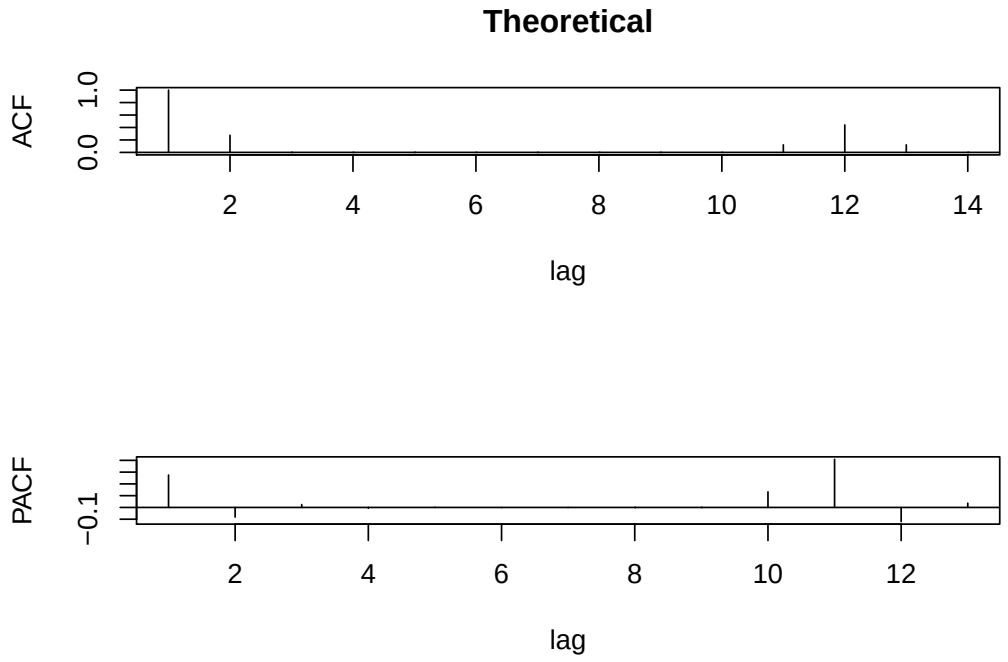


The sample ACF and PACF are plotted as below, the seasonality of $S = 12$ is quite evident from both plots.

```r
p <- acf2(sim_1c, max.lag=14)
```



then we plot the theoretical ACF and PACF based on the paramters the question provides.



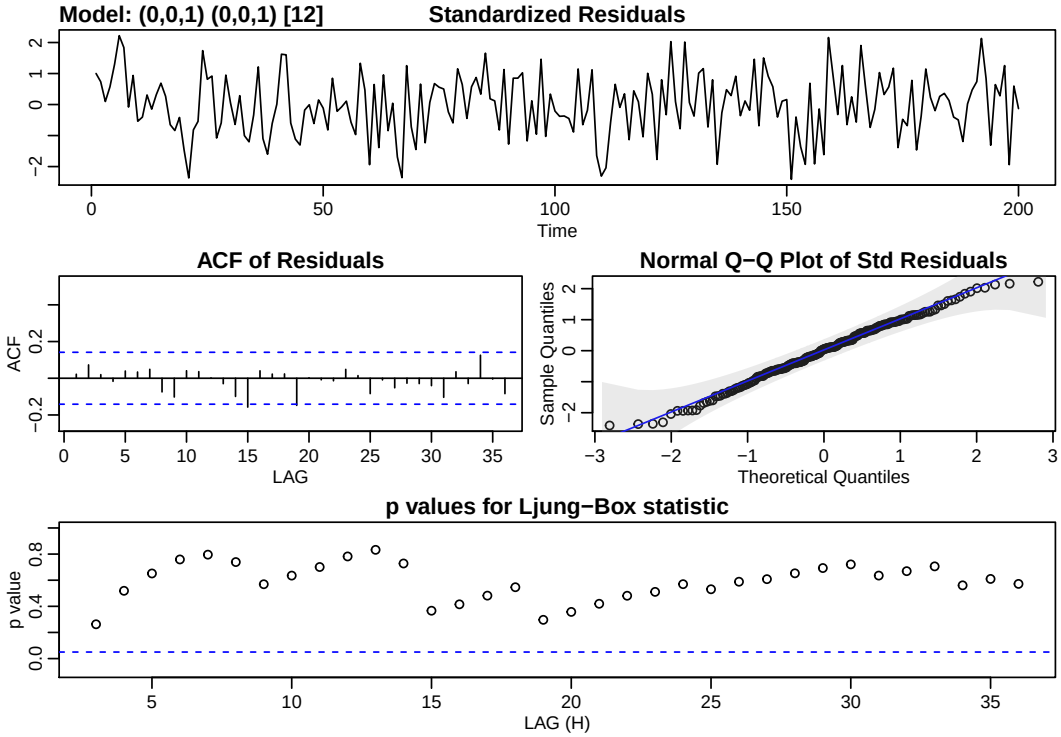
In sample plots, the ACF cut off after both $lag = 1$ and $lag = 12$. The theoretical plots are kindly repeated at the sample ACF and PACF, but the paths are much smoother and the PACF has a positive relation at $lag = 11$.

(d) Comparison with gausspr

To simplify, we use the series from question (c) here. Firstly, we fit such series by `sarima` function.

```
fit <- sarima(sim_1c, p=0, d=0, q=1, Q=1, S=12)
```

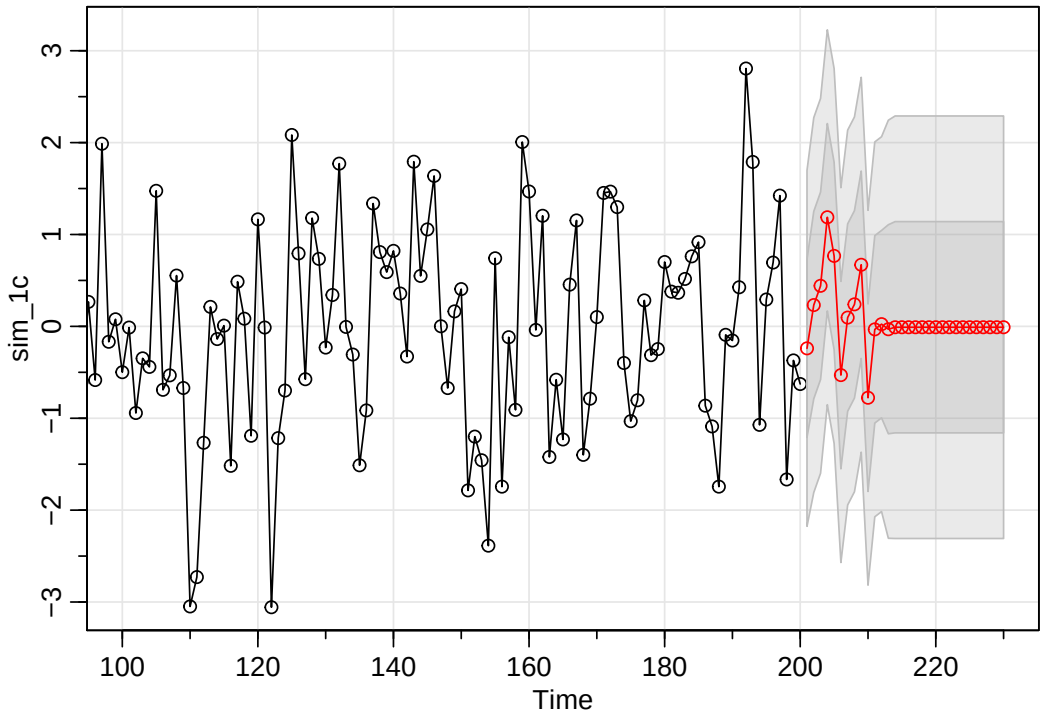
```
## initial  value 0.151267
## iter   2 value -0.010214
## iter   3 value -0.016167
## iter   4 value -0.017677
## iter   5 value -0.017824
## iter   6 value -0.018068
## iter   7 value -0.018076
## iter   8 value -0.018076
## iter   9 value -0.018076
## iter   9 value -0.018076
## iter   9 value -0.018076
## final  value -0.018076
## converged
## initial  value -0.021922
## iter   2 value -0.022404
## iter   3 value -0.022431
## iter   4 value -0.022477
## iter   5 value -0.022479
## iter   6 value -0.022480
## iter   7 value -0.022480
## iter   7 value -0.022480
## iter   7 value -0.022480
## final  value -0.022480
## converged
```



The fitting model works very well. All the p-values are large and points on Q-Q plot are close to the stright

line. Then we use `sarima.for` to forecast 30 new points and its predicted band.

```
sarima.for(sim_1c, 30, 0, 0, 1, Q=1, S=12)
```

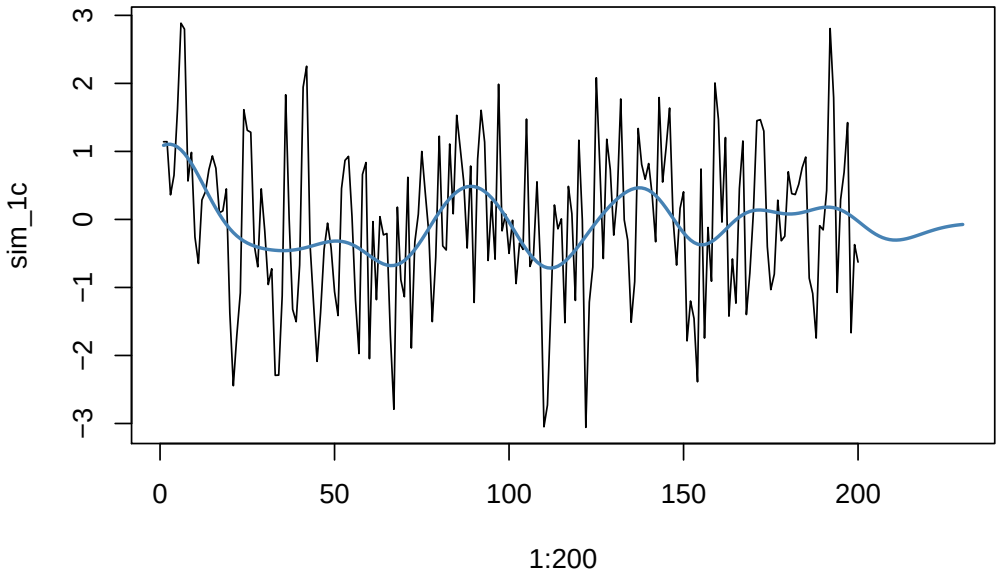



```
## $pred
## Time Series:
## Start = 201
## End = 230
## Frequency = 1
##  [1] -0.239333410  0.231639658  0.441320573  1.187379244  0.767009748
##  [6] -0.530286589  0.095861335  0.239632240  0.670403063 -0.776598473
## [11] -0.032611510  0.024144330 -0.031256352 -0.009492425 -0.009492425
## [16] -0.009492425 -0.009492425 -0.009492425 -0.009492425 -0.009492425
## [21] -0.009492425 -0.009492425 -0.009492425 -0.009492425 -0.009492425
## [26] -0.009492425 -0.009492425 -0.009492425 -0.009492425 -0.009492425
##
## $se
## Time Series:
## Start = 201
## End = 230
## Frequency = 1
##  [1] 0.9682769 1.0199562 1.0199562 1.0199562 1.0199562 1.0199562 1.0199562
##  [8] 1.0199562 1.0199562 1.0199562 1.0199562 1.0199562 1.0199562 1.1375610 1.1497185
## [15] 1.1497185 1.1497185 1.1497185 1.1497185 1.1497185 1.1497185 1.1497185
## [22] 1.1497185 1.1497185 1.1497185 1.1497185 1.1497185 1.1497185 1.1497185
## [29] 1.1497185 1.1497185
```

As we can see, the firstly 12 predicted points seem to be good and follow the previous pattern. But the remaining 18 points become 0 very quickly. We use another function `gausspr` with default parameters to fit and forecast data.

```
fit2 <- gausspr(x=1:200, y=sim_1c)

## Using automatic sigma estimation (sigest) for RBF or laplace kernel
plot(1:200, sim_1c, type="l", xlim=c(1,230))
lines(x=1:230, y=predict(fit2, 1:230)[,1], col = "steelblue", lwd=2)
```

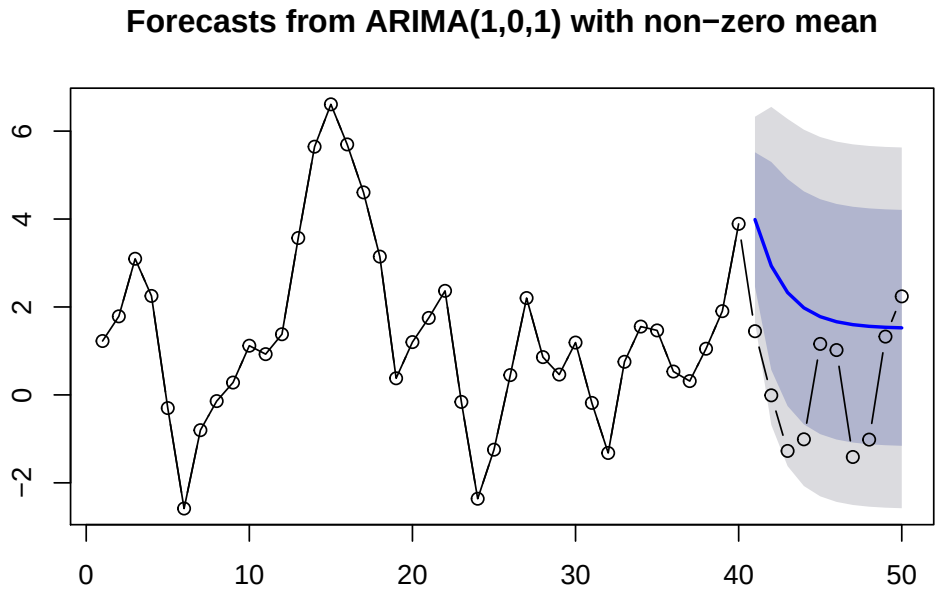


It seems that such model can predict the trend of data well, but it almost ignores the seasonality of such series.

(e) ARMA(1,1)

```
set.seed(12345)
sim_1e <- arima.sim(list(ar=0.7, ma=0.5), 50)

sample <- sim_1e[1:40]
fit40  <- auto.arima(sample, max.p = 1, max.q = 1,
                     allowmean = TRUE)      # zero-mean
fore <- forecast(fit40, 10)
plot(fore)
lines(sim_1e,type="b")
```




```
## Number of points outsides the 95% prediction interval: 1
```

Although the locations of 10 predicted points are very wired, but the 95% prediction band (the grey area) follows the pattern of previous data well. In additionally, there is only one actual point outsides such prediction interval.

Assignment 2. ACF and PACF diagnostics.

```
set.seed(12345)
rm(list = ls())
```

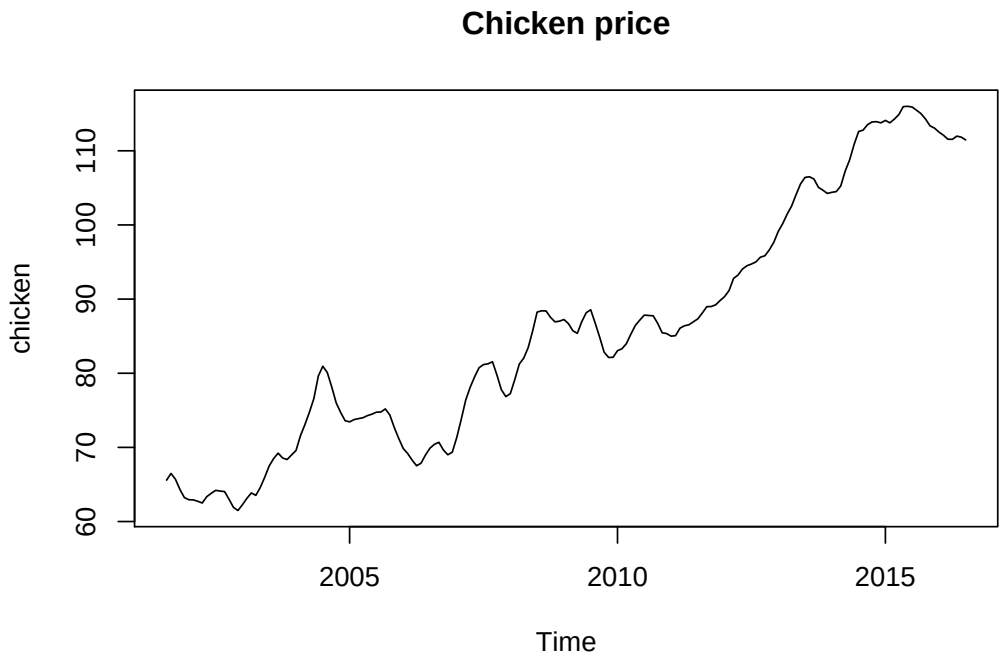
a) Model selection - chicken data

For data series chicken in package astsa (denote it by (x_t) plot 4 following graphs up to 40 lags: ACF(x_t), PACF(x_t) ACF(∇x_t), PACF(∇x_t), (group them in one graph). Which $ARIMA(p,d,q)$ or $ARIMA(p,d,q) \times (P,D,Q)_S$ models can be suggested based on this information only? Motivate your choice.

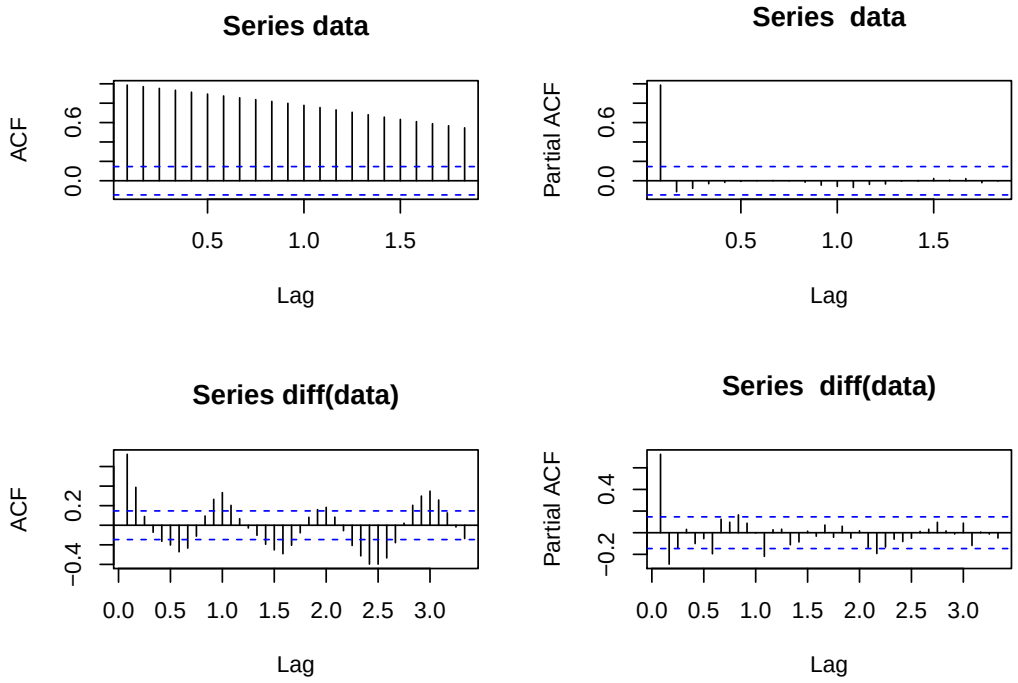
In this part of the task I created a function that creates the different visualizations. The function can be seen in the appendix. The output of this function are 2 ACF and 2 PACF plots. The visualization in the upper left part is the ACF plot in the lower left lagged ACF. In the upper right part is the PACF and in the lower right part the lagged PACF plot. In the task it is given to use up to 40.

- Data: **chicken**

```
plot(chicken, main = "Chicken price")
```




```
plot_function(chicken, 40)
```



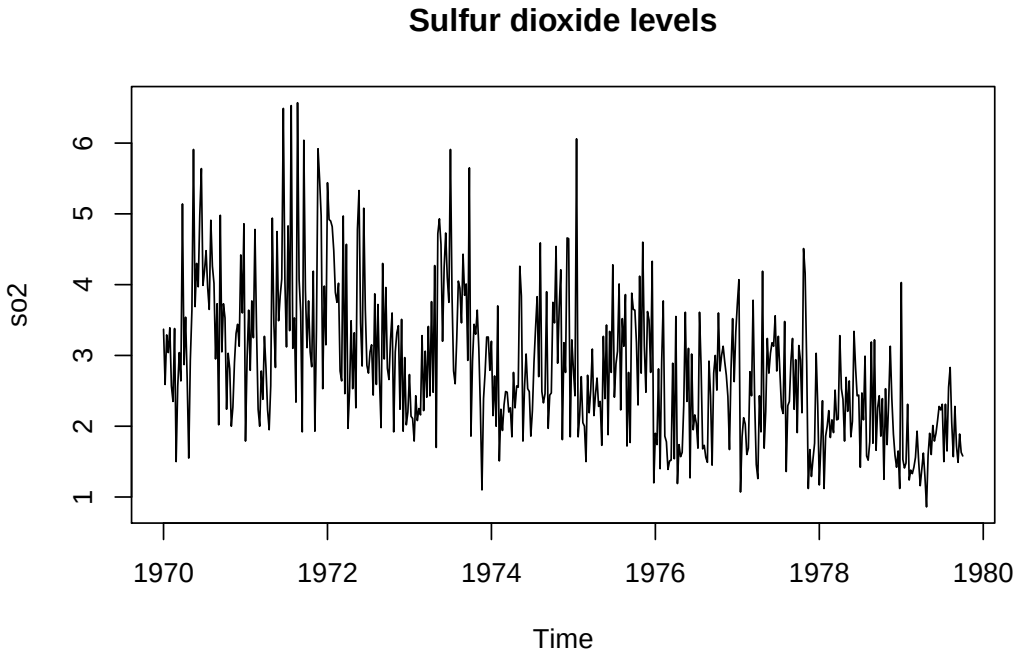
In the upper right and left part are the not lagged plots. The ACF value decreases only slightly, whereas the PACF after the first lay falls into the area with the blue line. From this it can be concluded that an AR(1) or an ARIMA(1,0,0) model could be appropriate.

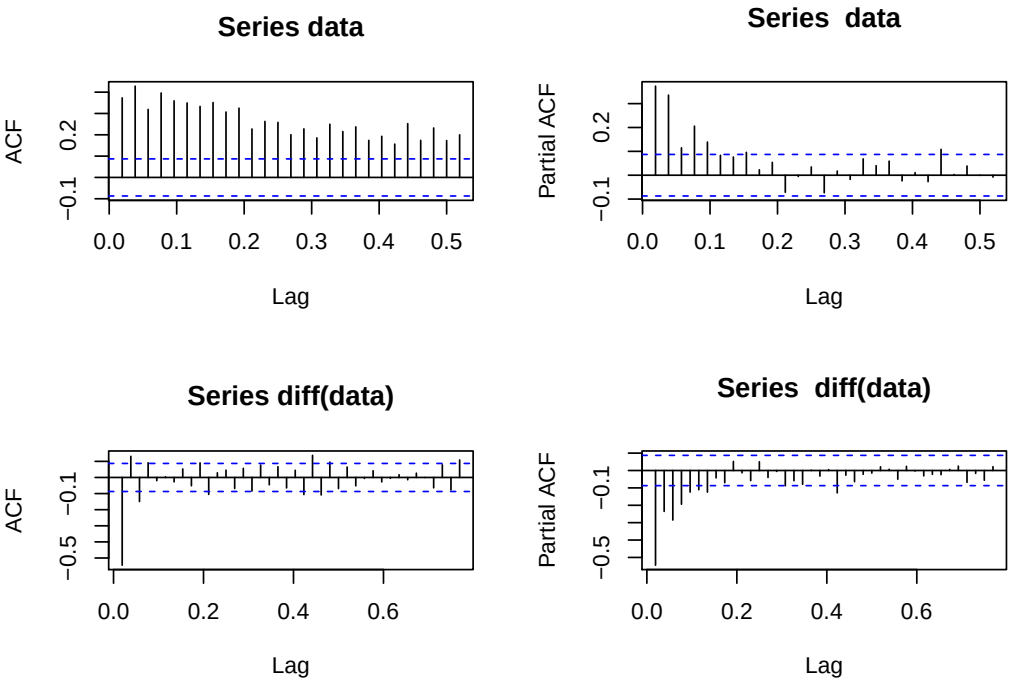
In the lower part of the plot a season patter can be seen in the ACF, where s=12.

b) Model selection - different data

Repeat step 1 for the following datasets: so2, EQcount, HCT in package `astsa`.

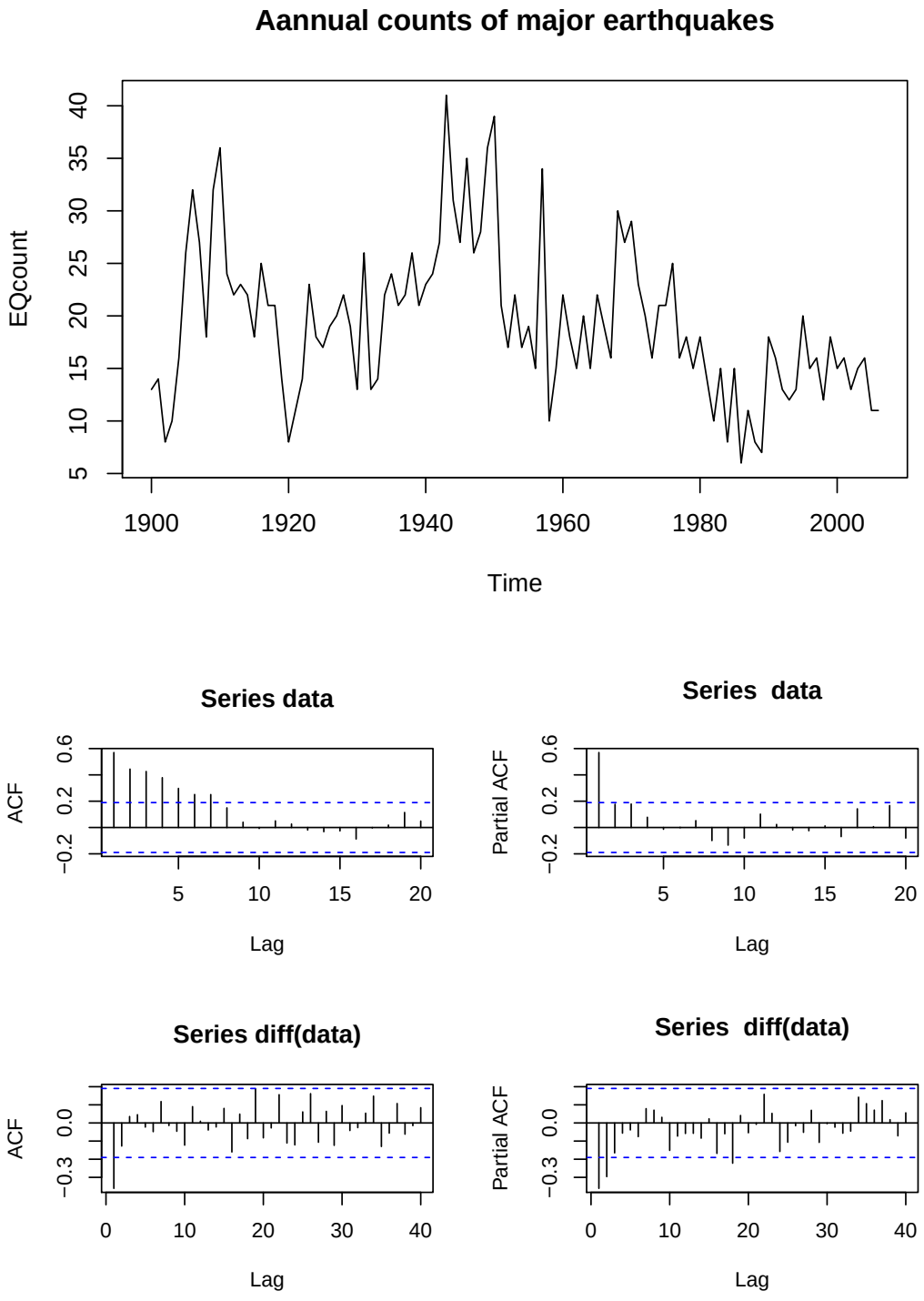
- Data: **so2**





If you look at the time series, as well as the ACF and PACF, it's not easy to say what a model could look like. However, ACF and PACF plots show tailing off rather than any cut-off point. However, if we look at the differencing term, i.e. the lower part of the plot, we see that ACF cuts off after lagged 1. The differencing PACF tails of, so I would suggest an ARMA(0,1,1) model.

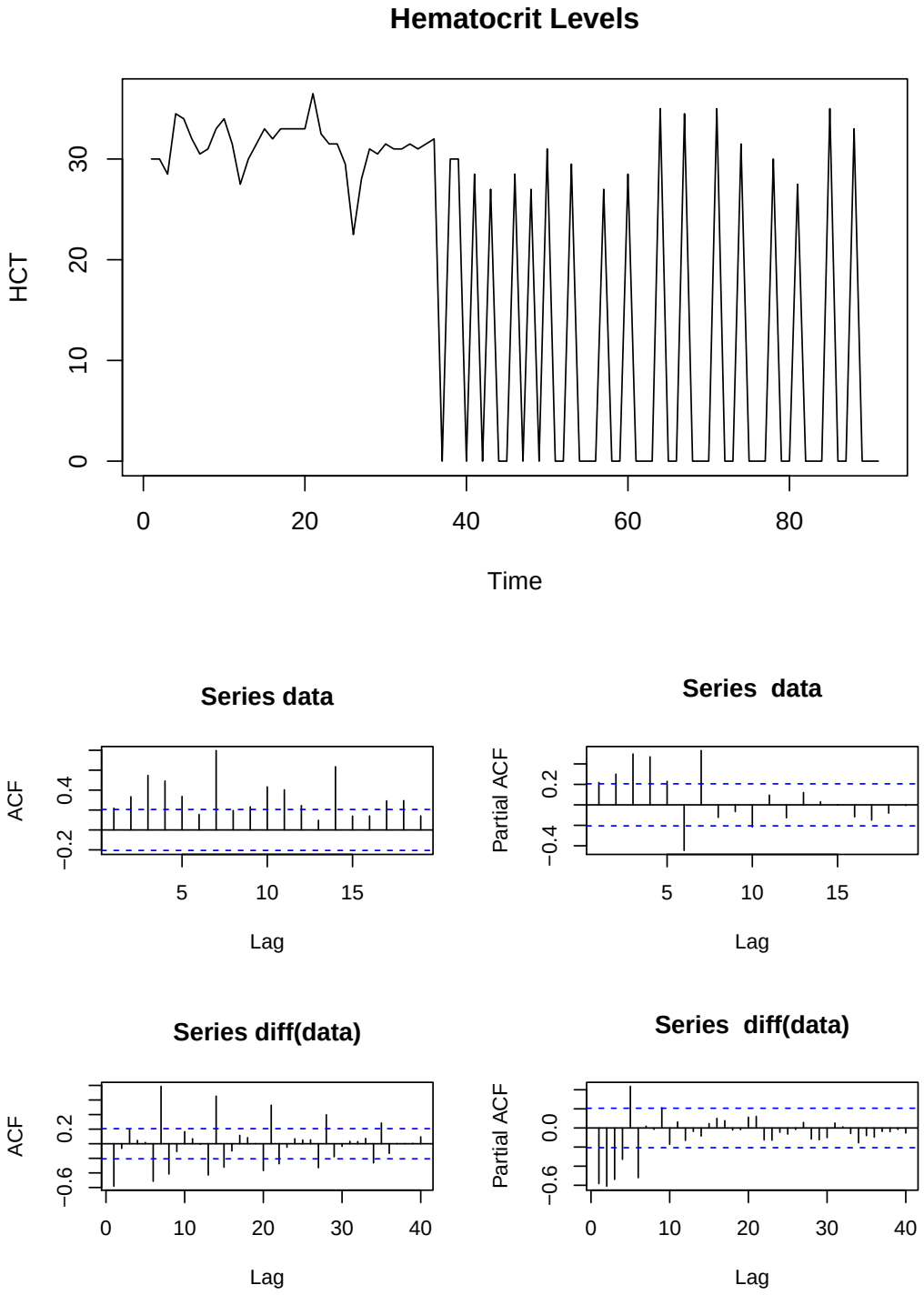
- Data: **EQcount**



In the upper part of the visualization it can be seen that the ACF value is in the blue range after 8 lags, a slow reduction can be observed. With the PACF, however, the value after the first dropped was in the blue range. Accordingly I would recommend here again for an AR(1) or ARMA(1,0,0) as model.

In the lower part of the visualization you can almost see the opposite of the upper part. The ACF value drops off after one lag and the PACF value is in the blue area after some lag. Accordingly, an ARIMA model(0,0,1) could also be considered.

- Data: **HCT**

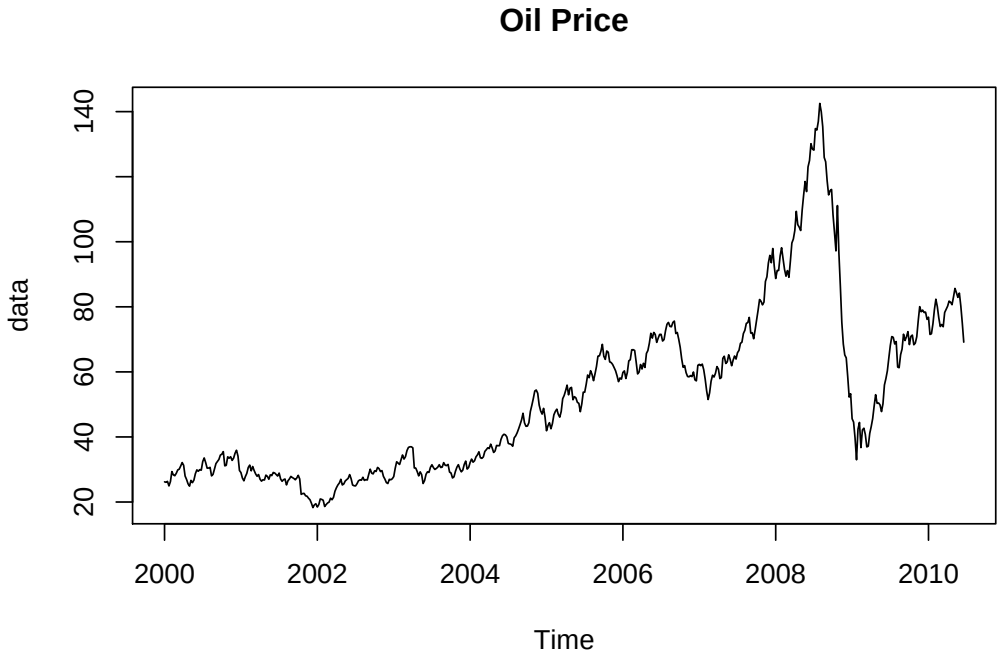


In the upper part of the plot we can say that the ACF plot shows a season with s=7. The PACF part cuts of with lag 7, therefor we could say that the non-seasonal part may be AR(7) or ARIMA(7,0,0).

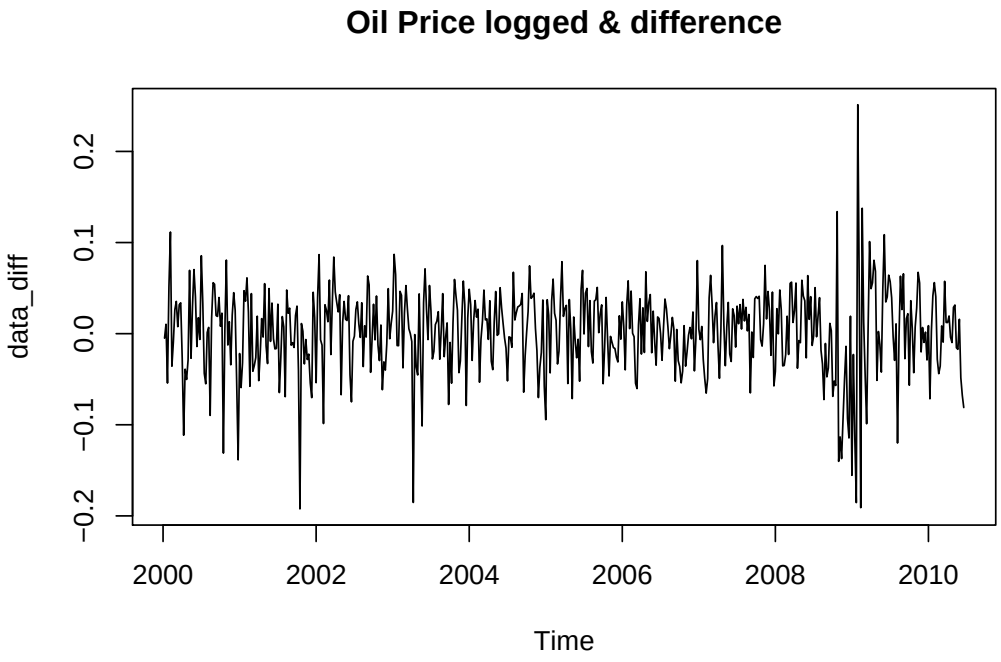
Assignment 3. ARIMA modeling cycle

(a)

Visualization




```
### transform
# log for narrow the variance
# diff for stationary removal
data_diff = diff(log(data))
plot(data_diff,  main = "Oil Price logged & difference")
```




```
# looks stationary
```

There is an obvious trend in the original oil data, we use logarithm transformation firstly to narrow the variance of data. Then we apply one-order difference to the data and get a stationary-like time series.

Hypothesis test

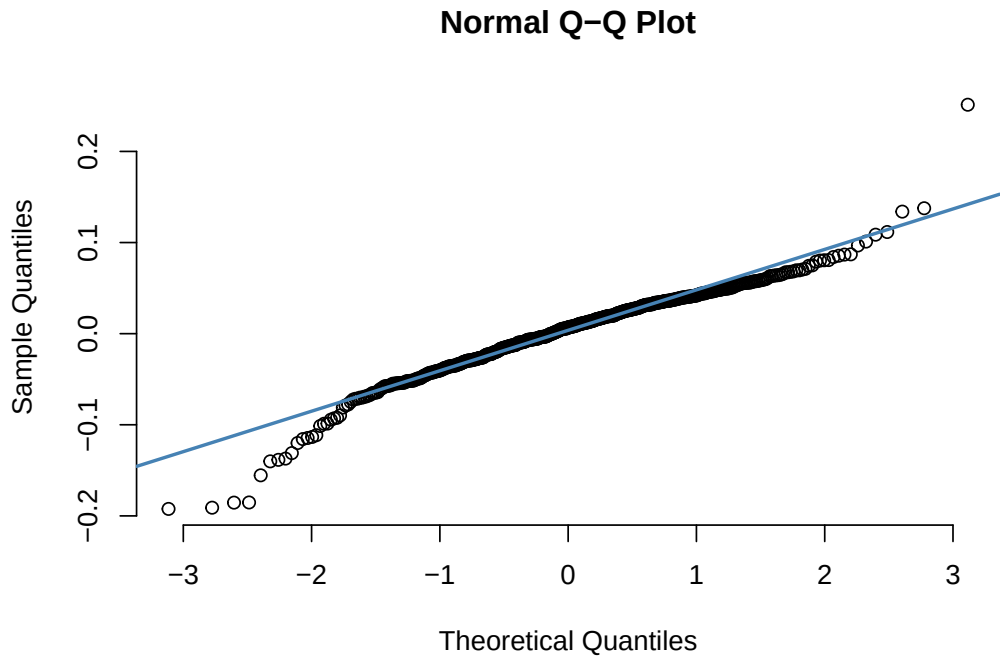
```
## Warning in adf.test(data_diff): p-value smaller than printed p-value
```

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

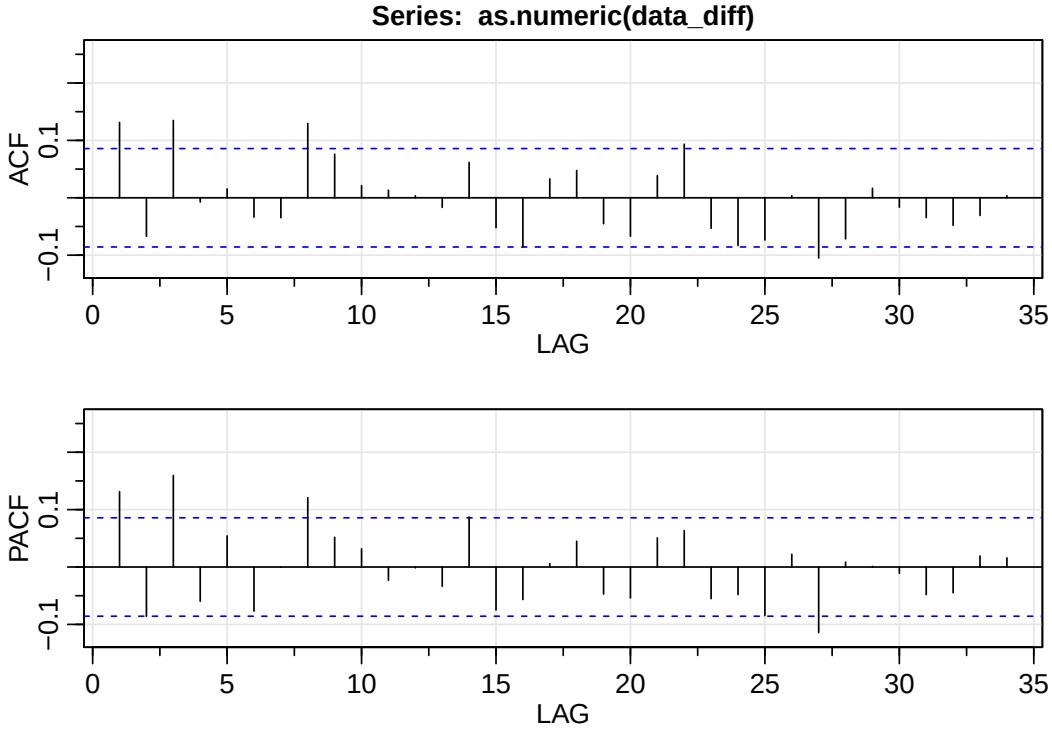
```
##
##  Box-Ljung test
##
## data:  data_diff
## X-squared = 9.4307, df = 1, p-value = 0.002134
```

In *Box-Ljung* test, we reject the null hyperthsis, which means that the series is not independence. In *unit root* test, it asks us to accept its alternative hypothesis, which says the series is stationary.

Plot analysis



The Q-Q plot shows that the sample p-value is significantly lower than the expected p-value in the beginning, but higer in the end. In general, sample and theoretical quantiles follow similar distributions, since most of the points are on the stright line. But the sample quantiles seem more dispersed because of such "S-shape".

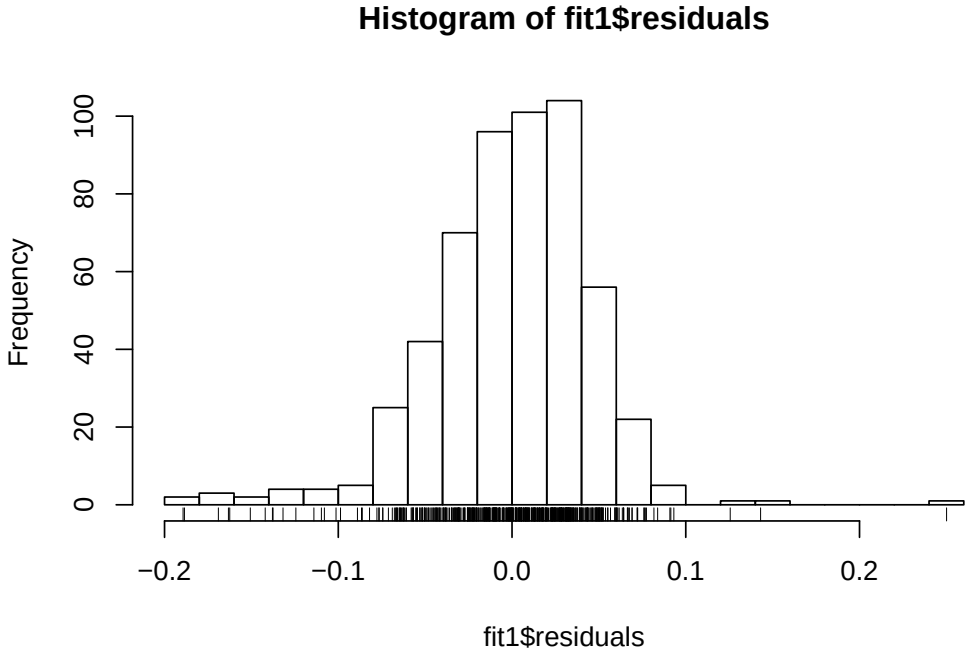


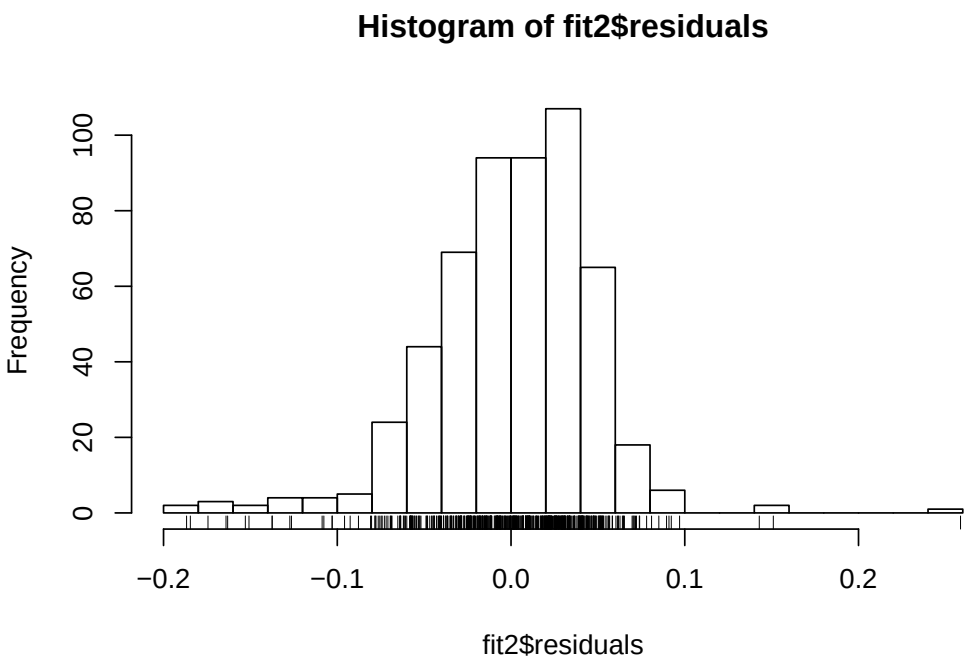

```
## AR/MA
##   0 1 2 3 4 5 6
## 0 x o x o o o o
```

```
## 1 x o x o o o o
## 2 x x x o o o o
## 3 x x x o o o o
## 4 x o x o o o o
## 5 x x x o x o o
## 6 o x x o x x o
## 7 o x x x x x x
```

Both ACF and PACF plots tail off when lag is larger, which means the series might follow an ARMA model. By EACF, we can assume our logged and diffed series follow either ARMA(1,1) or ARMA(3,3).

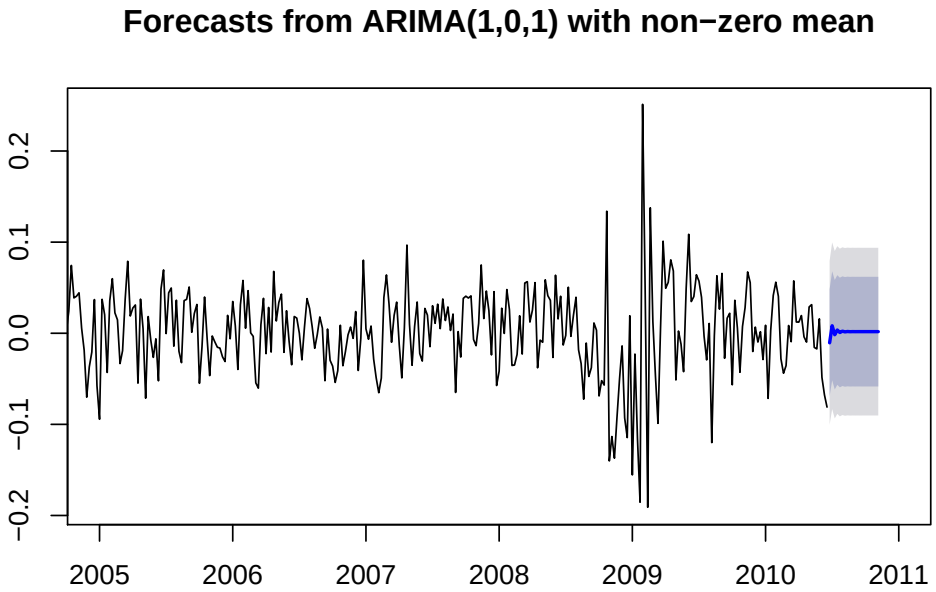
`Fitting`



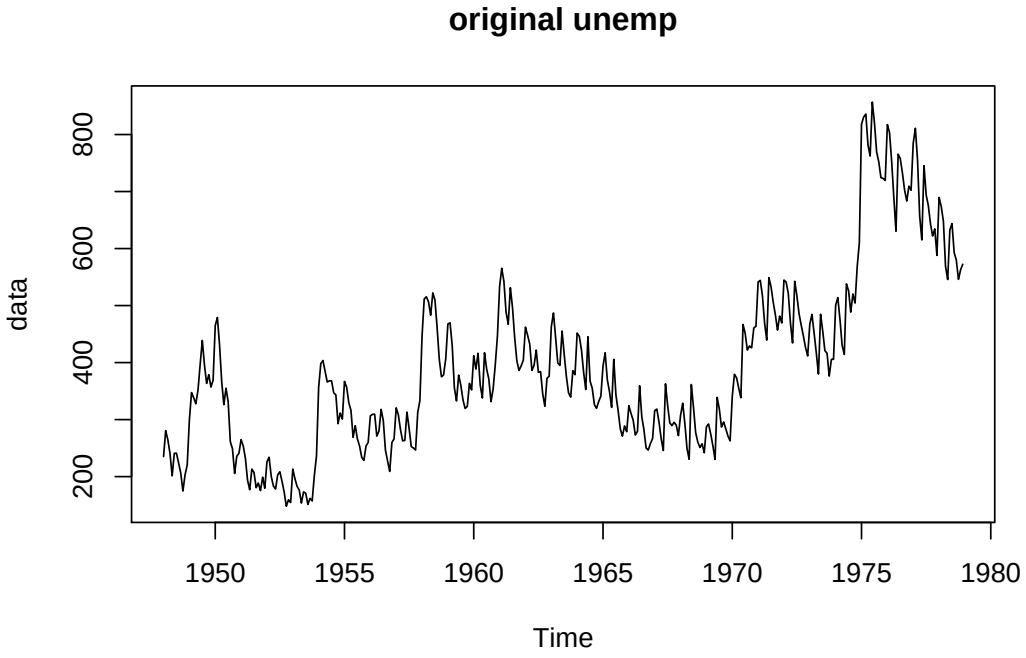


Since the results from such two fitting models are quite similar, we choose the simpler model as the best fitting, which means that our logged series follow ARIMA(1,1,1).

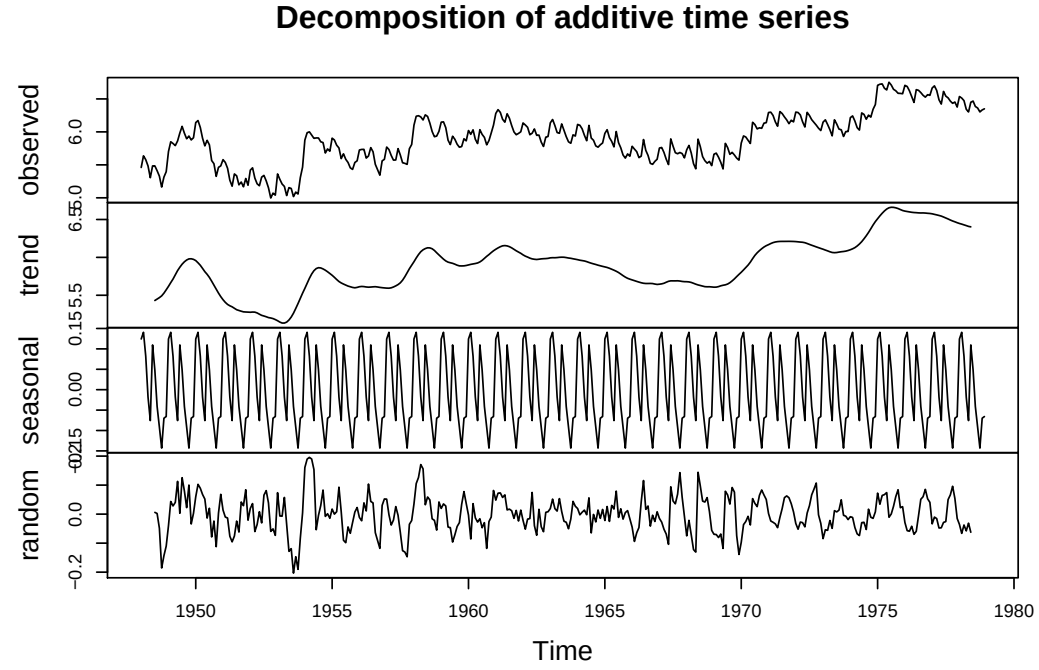
`Forecast`

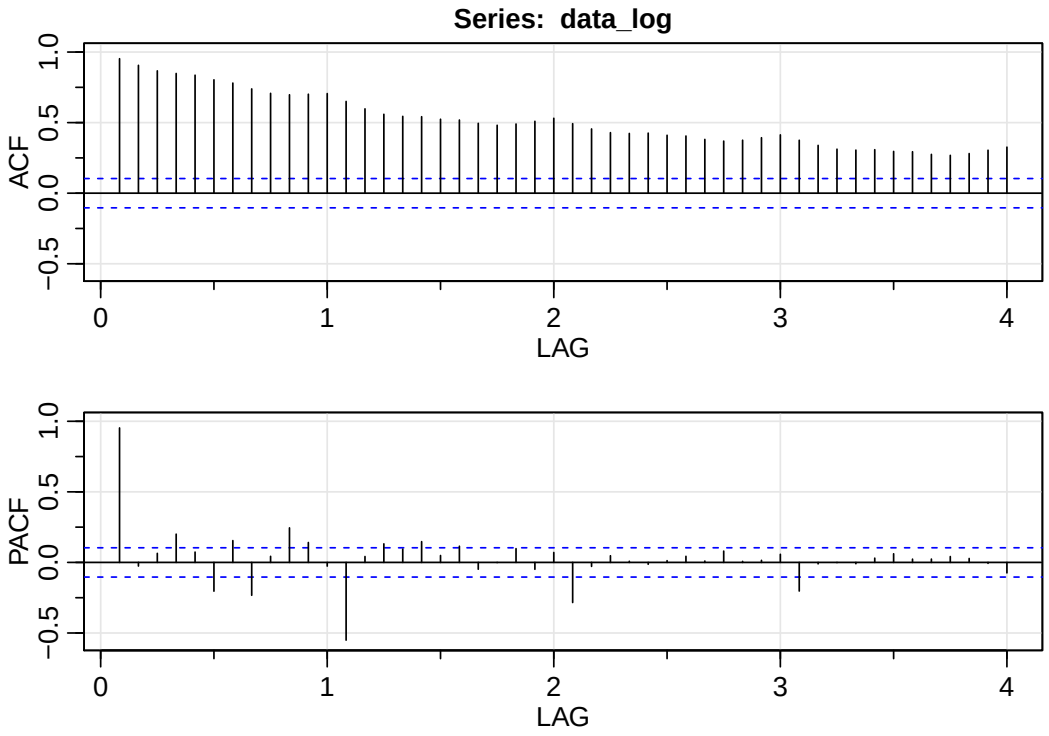


(b)

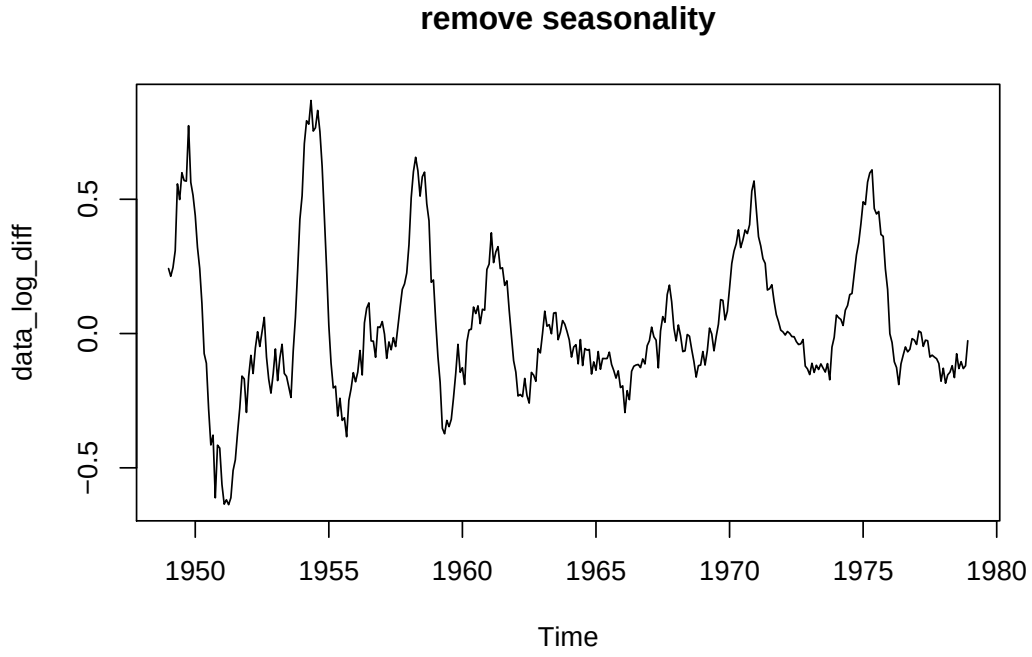


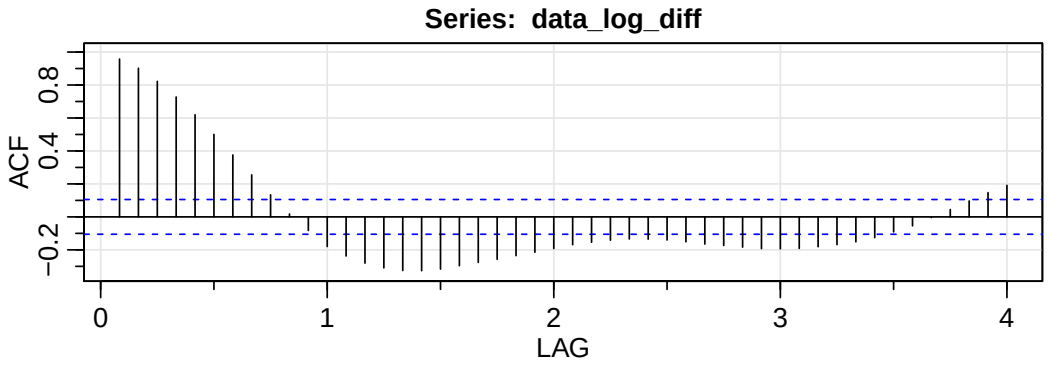
The original `unemp` data is non-stationary obviously. We use log transposition and `decompose` to check the seasonality of such series.

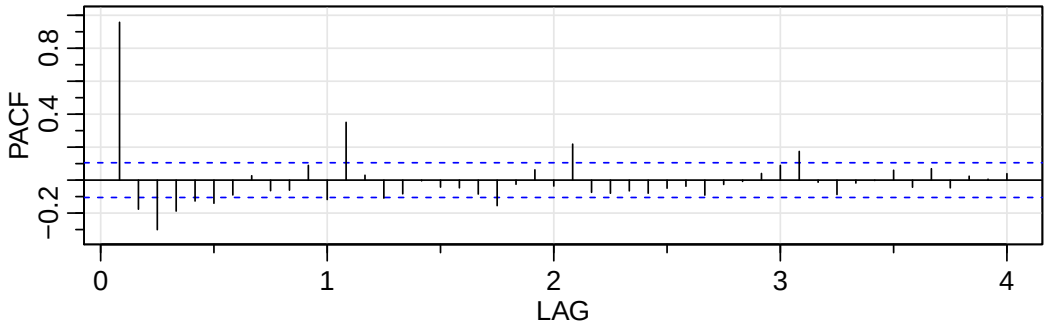


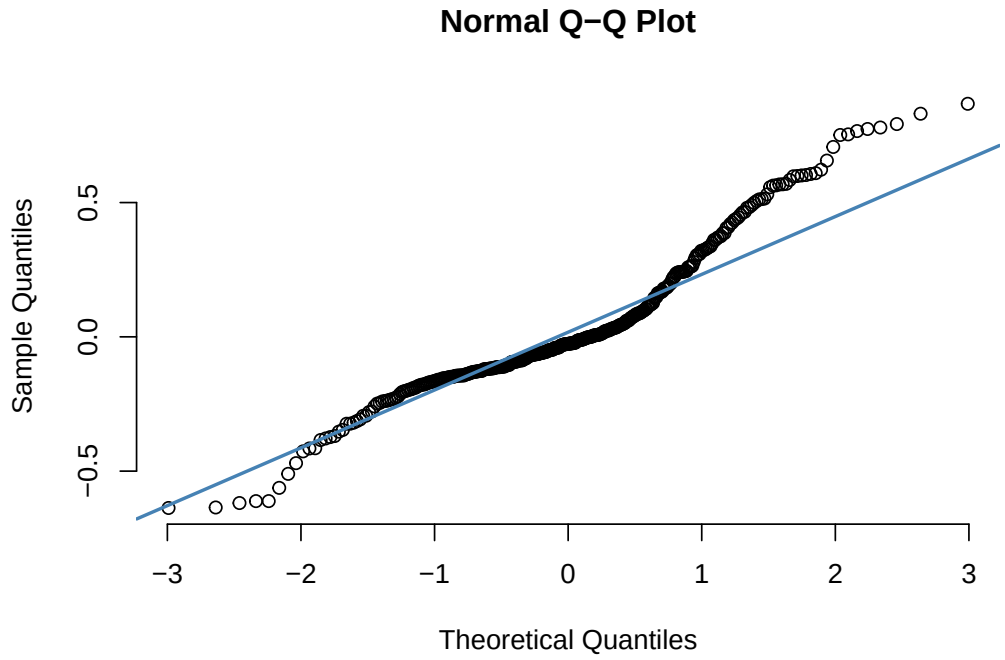


It seems the seasonality $S = 12$, and $P = 1$, $Q = 0$. thus we try the first difference with $lag = 12$ to remove its seasonal trend.










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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data_log_diff
## Dickey-Fuller = -6.2079, Lag order = 7, p-value = 0.01
```

```
## alternative hypothesis: stationary

##
##  Box-Ljung test
##
## data:  data_log_diff
## X-squared = 332.7, df = 1, p-value < 2.2e-16
```

The detrending works well and two tests shows that our logged diffed series is dependent and statioanry. Since both PACF and ACF tails off, we assume such series follows ARIMA(p,0,q) model. Thus, we apply `eacf` to our series.

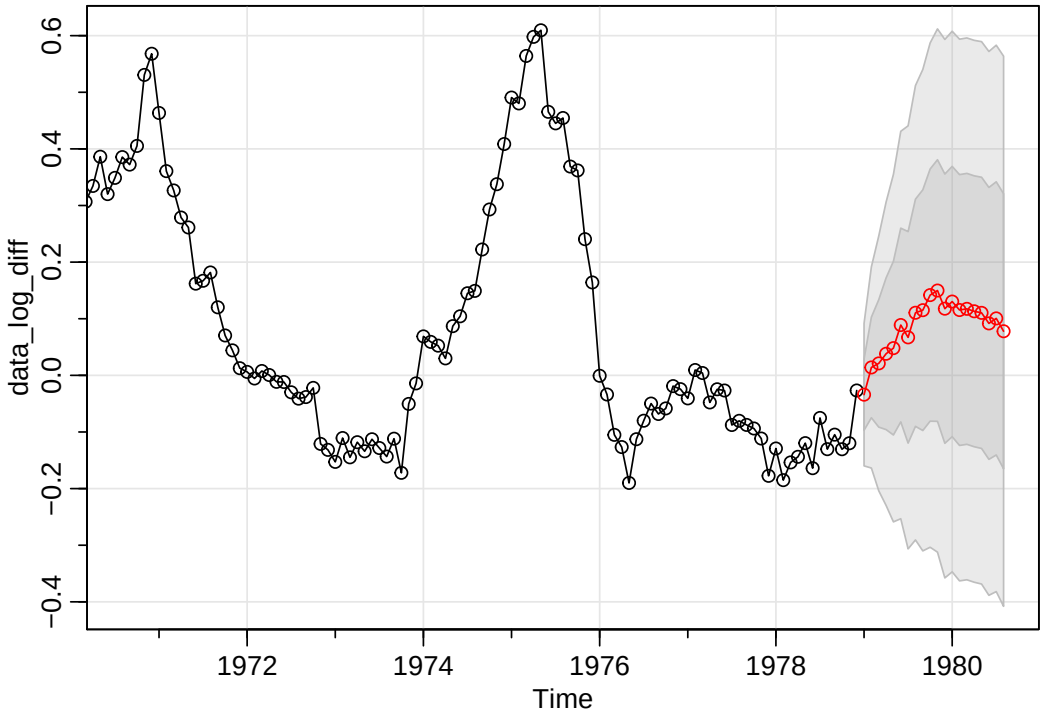
```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x x x x x x x o o  x  x  x
## 1 x x x x x x x x x o o  x  x  x
## 2 x x o o o o o o o x x  x  x  o
## 3 x o o o o o x o o o x  x  x  x
## 4 x o x o o o o o o o o  x  x  x
## 5 x x x o o o o o o o o  x  x  o
## 6 x x x x x o o o o o o  x  x  o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

Assume our model is either ARIMA(3,0,1) or ARIMA(2,0,2). The fitting results are shown as follow.

```
##
## 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:
##          ar1      ar2      ar3      ma1     sar1   xmean
##       1.7672  -0.6637  -0.1229  -0.7706  -0.4630  0.0322
## s.e.  0.0992   0.1588   0.0673   0.0882   0.0518  0.0271
##
## sigma^2 estimated as 0.003957:  log likelihood = 481.95,  aic = -949.9

##
## 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:
##          ar1      ar2      ma1     sar1   xmean
##       1.9224  -0.9365  -0.8559  -0.4790  0.0329
## s.e.  0.0283   0.0268   0.0480   0.0499  0.0236
##
## sigma^2 estimated as 0.003992:  log likelihood = 480.31,  aic = -948.62
```

The fitting with $ARIMA(3,0,1) \times (1,0,0)_{12}$ (the first fitting) has smaller AIC and larger log lokelihood. Based on these fitting, the final prediction is




```
## $pred
##              Jan         Feb         Mar         Apr         May
## 1979 -0.03411690  0.01397185  0.02110653  0.03813488  0.04809483
## 1980  0.13036306  0.11537141  0.11746581  0.11316675  0.11041392
##              Jun         Jul         Aug         Sep         Oct
## 1979  0.08891155  0.06709597  0.11056913  0.11510462  0.14177038
## 1980  0.09175893  0.10061147  0.07789249
##              Nov         Dec
## 1979  0.14981609  0.11775909
## 1980
##
## $se
##             Jan        Feb        Mar        Apr        May        Jun
## 1979 0.06290610 0.08881259 0.11249306 0.13392508 0.15344516 0.17107490
## 1980 0.23871840 0.23918809 0.23925162 0.23926805 0.23949634 0.24011986
##             Jul        Aug        Sep        Oct        Nov        Dec
## 1979 0.18680414 0.20062759 0.21256620 0.22267378 0.23103787 0.23777746
## 1980 0.24124543 0.24290816
```

Appendix

```
knitr::opts_chunk$set(echo = TRUE, out.width = "400px")
library(astsa)
library(kernlab)
library(latex2exp)
library(kableExtra)
library(forecast)
library(TSA)
library(tseries)
rm(list=ls())
set.seed(12345)
# the simulation of ar(3) process
sim <- arima.sim(list(ar=c(0.8,-0.2,0.1)), n=1000)
# create lag data.frame
dat <-ts.intersect(x0=sim, x1=lag(sim,-1), x2=lag(sim,-2),
                   x3=lag(sim,-3), dframe = T)
# create the potiential linear relation among nearby observations
res1 <- lm(x0~x1+x2, data=dat)
res2 <- lm(x3~x2+x1, data=dat)
# the estimation of noise
r1 <- residuals(res1)
r2 <- residuals(res2)
cor(r1, r2)
pacf(sim, lag.max = 33, main = "simulation of AR(3)")
ARMAtoAR(c(0.8,-0.2,0.1), lag.max = 33)[33]
# Simulate an AR(2) process
set.seed(12345)
ar_sim2 = arima.sim(model = list(ar = c(0.8, 0.1)), n = 100)

# method of moments (Yule-Walker equations)
yw = ar(ar_sim2, order = 2,
        method = c("yule-walker"),
        aic = FALSE)

# - conditional least squares
cls = ar(ar_sim2, order = 2,
        method = c("ols"),
        aic = FALSE)
# - maximum likelihood (MLE)
ml = ar(ar_sim2, order = 2,
        method = c("mle"),
        aic = FALSE)

# compare results to the true values
# - true values are given
matrix_a1_b = matrix(c(0.8,yw$ar[1],cls$ar[1],ml$ar[1],
                     0.1,yw$ar[2],cls$ar[2],ml$ar[2]),
                     ncol = 2, nrow = 4)

df_a1_b = as.data.frame(matrix_a1_b,
                        row.names = c("True",
                                      "Yule-Walker",
```

```
                                    "conditional least squares",
                                    "maximum likelihood "),
                        col.names = c("phi1", "phi2"))
df_a1_b = round(df_a1_b,3)
kable(df_a1_b) %>%
  kable_styling(latex_options="basic")
# The asymptotic-theory variance matrix of the coefficient estimates
name <- c("phi1", "phi2")
mat <- as.data.frame(ml$asy.var.coef, row.names = name, col.names = name)
kable(mat) %>%
  kable_styling(latex_options="basic")
va <- ml$asy.var.coef[2,2]
va
sd <- sqrt(va)
int <- 1.96*sd/sqrt(100)
sd
cat(c(ml$ar[2]-int, ml$ar[2]+int))
set.seed(123456)
# generate seasonal ARIMA(0,0,1)*(0,0,1)_12
sim_1c <- arima.sim(list(order = c(0,0,13), ma = c(0.3,rep(0,10),0.6, 0.3*0.6)), n = 200)
plot(sim_1c, type="l")
p <- acf2(sim_1c, max.lag=14)
par(mfrow=c(2,1))
plot(ARMAacf(ma = c(0.3,rep(0,9),0.6, 0.3*0.6), pacf=FALSE),
     type="h",xlab="lag",ylab="ACF",main="Theoretical",); abline(h=0)
plot(ARMAacf(ma = c(0.3,rep(0,9),0.6, 0.3*0.6), pacf=TRUE),
     type="h",xlab="lag",ylab="PACF"); abline(h=0)
fit <- sarima(sim_1c, p=0, d=0, q=1, Q=1, S=12)
sarima.for(sim_1c, 30, 0, 0, 1, Q=1, S=12)
fit2 <- gausspr(x=1:200, y=sim_1c)
plot(1:200, sim_1c, type="l", xlim=c(1,230))
lines(x=1:230, y=predict(fit2, 1:230)[,1], col = "steelblue", lwd=2)
set.seed(12345)
sim_1e <- arima.sim(list(ar=0.7, ma=0.5), 50)

sample <- sim_1e[1:40]
fit40  <- auto.arima(sample, max.p = 1, max.q = 1,
                     allowmean = TRUE)      # zero-mean
fore <- forecast(fit40, 10)
plot(fore)
lines(sim_1e,type="b")
dt <- cbind(sim_1e[41:50], fore$lower[,2], fore$upper[,2])
out <- 0
for(i in 1:10){
  if(!dt[i,1]>dt[i,2]&&dt[i,1]<dt[i,3]){
    out <- out+1
  }
}
cat("Number of points outsides the 95% prediction interval:",out,"\n")
set.seed(12345)
rm(list = ls())
# get all the data
# chicken, so2, EQcount, HCT
```

```r
data(chicken)
data(so2)
data(EQcount)
data(HCT)

plot_function = function(data,lag_dif){
  # data: the data - as time series object
  # lag_dif: the lag difference
  par(mfrow = c(2,2))

  # not lagged data
  acf_obj = acf(data)
  pacf_obj = pacf(data)

  # the lagged data
  acf_obj_lag = acf(diff(data), lag.max = lag_dif)
  pacf_obj_lag = pacf(diff(data), lag.max = lag_dif)

}

plot(chicken, main = "Chicken price")

plot_function(chicken, 40)
plot(so2, main = "Sulfur dioxide levels")
plot_function(so2, 40)

plot(EQcount, main = "Aannual counts of major earthquakes")
plot_function(EQcount, 40)

plot(HCT, main = "Hematocrit Levels")
plot_function(HCT, 40)
set.seed(12345)
rm(list = ls())

data(oil)
data <- oil
# original
plot(data, main = "Oil Price")
### transform
# log for narrow the variance
# diff for stationary removal
data_diff = diff(log(data))
plot(data_diff,  main = "Oil Price logged & difference")
# looks stationary
# unit root test
adf.test(data_diff)
# Box-Ljung test
Box.test(data_diff, type="Ljung-Box")
# Q-Q plots
qqnorm(data_diff, pch = 1, frame = FALSE)
qqline(data_diff, col = "steelblue", lwd = 2)
# ACF and PACF plots: acf2
p <- acf2(as.numeric(data_diff))
```

```r
# EACF analysis
# choose a circle locating on dignal
# The model should be as simple as it can be
eacf(data_diff, ma.max = 6)
# ARIMA fit analysis
# model 1 & 2
fit1 = arima(x = data_diff, c(3,0,3))
fit2 = arima(x = data_diff, c(1,0,1))

hist(fit1$residuals, breaks=30)
rug(fit1$residuals)
hist(fit2$residuals, breaks=30)
rug(fit2$residuals)
###### choose one of the model
best_fit <- arima(x=data_diff, c(1,0,1))

# predictions
fore <- forecast(data_diff, 20, model=best_fit)
plot(fore,xlim=c(2005, 2011))
rm(list=ls())
data(unemp)
data = unemp

### visualisation of original plot
plot(data, main = "original unemp")
### transform -- log
data_log <- log(data)
# evidently seasonal
plot(decompose(data_log))
p <- acf2(data_log)
data_log_diff = diff(data_log, lag = 12)
plot(data_log_diff, main="remove seasonality")
p <- acf2(data_log_diff)
# Q-Q plot
qqnorm(data_log_diff, pch = 1, frame = FALSE)
qqline(data_log_diff, col = "steelblue", lwd = 2)
# unit root test
adf.test(data_log_diff)
# Box-Ljung test
Box.test(data_log_diff, type="Ljung-Box")
eacf(data_log_diff)
# fit
fit1 <- sarima(data_log_diff,p=3,d=0,q=1,P=1,S=12,details = FALSE)
fit2 <- sarima(data_log_diff,p=2,d=0,q=1,P=1,S=12,details = FALSE)
fit1$fit
fit2$fit  # bigger log-llk, smaller aic
sarima.for(data_log_diff, 20,3,0,1,1,1,0,0,12)
```