TSA Computer Lab 2

Joshua Hudson, Carles Sans

26 September 2017

Assignment 1: Computations with simulated data

a)

In this question, we first used `arima.sim()` to generate 1000 observations from the AR(3) process given below:

$$x_t = 0.8x_{t-1} - 0.2x_{t-2} + 0.1x_{t-3} + w_t$$

We then implemented our own calculation of the partial autocorrelation at lag h=3 using the following formulae:

$$\phi_{33} = corr(x'_{t+3}, x''_t)$$

where

$$x'_{t+3} = x_{t+3} - \sum_{j=1}^{3-1} \hat{\beta}_j x_{t+3-j} = x_{t+3} - (\hat{\beta}_1 x_{t+2} + \hat{\beta}_2 x_{t+1})$$

and

$$x''_t = x_t - \sum_{j=1}^{3-1} \hat{\beta}_j x_{t+j} = x_{t+3} - (\hat{\beta}_1 x_{t+1} + \hat{\beta}_2 x_{t+2})$$

The $\hat{\beta}_j$s are the same in both equations and were found using the linear regression function `lm()`. We also computed the partial autocorrelation at lag 3 with the in-built R function `pacf()`. Additionally, we compare these two values to the theoretical value of 0.1, seeing as for an AR(3) process, $\phi_{33} = \phi_3$. The 3 values found are shown in the table below.

```r
ar3_model <- list(ar = c(0.8, -0.2, 0.1))
set.seed(12345)
ar3_ts <- arima.sim(ar3_model, n = 1000)

ar3_df <- ts.intersect(xt = ar3_ts, xtlag1 = lag(ar3_ts, -1), xtlag2 = lag(ar3_ts, -2))
#autoregression on xt
lm_model <- lm(xt ~ xtlag1 + xtlag2, ar3_df)
Bhat <- lm_model$coefficients[-1]
xtprpr <- ar3_ts - (Bhat[1]*lag(ar3_ts, -1) + Bhat[2]*lag(ar3_ts, -2))
xtlag3pr <- lag(ar3_ts, -3) - (Bhat[1]*lag(ar3_ts, -2) + Bhat[2]*lag(ar3_ts, -1))
phi33 <- cor(xtlag3pr, xtprpr)
phi_R <- pacf(ar3_ts, lag.max = 3, plot = FALSE)$acf
phi33_R <- phi_R[3,1,1]
phi33_df <- data.frame("My_estimate" = phi33, "R_estimate" = phi33_R, "Theoretical" = ar3_model$ar[3])

library(knitr)
kable(phi33_df, caption = "Comparison of the estimates and theoretical PACF for lag 3")
```

Table 1: Comparison of the estimates and theoretical PACF for lag 3

My_estimate	R_estimate	Theoretical
0.1248488	0.1170643	0.1

We can see that our estimate of ϕ_33 was less accurate than the estimate obtained using `pacf()`.

b)

In this question, We generated 100 observations from the AR(2) process given by:

$$x_t = 0.8x_{t-1} + 0.1x_{t-2} + w_t$$

We then used the `ar()` function to fit the parameters based on the sample, using the Yule-Walker method. We also used the `arima()` function to fit the parameters using the conditional least squares and maximum likelihood methods. The 3 sets of parameter estimates are shown below:

```r
set.seed(12345)
Ar2<-arima.sim(n = 100, list(ar = c(0.8, 0.1)))

Ar2Yule<-ar(Ar2, aic = TRUE, method = c("yule-walker"), order = 2)
AR2CSS<-arima0(Ar2, order = c(2,0,0), method = "CSS", optim.control = list(maxit = 1000))
AR2MLE<-arima0(Ar2, order = c(2,0,0), method = "ML", optim.control = list(maxit = 1000))

Yule<-cbind(Ar2Yule$ar, sqrt(Ar2Yule$var.pred))
CSS<- cbind(AR2CSS$coef, sqrt(diag(AR2CSS$var.coef)))
MLE<-cbind(AR2MLE$coef, sqrt(diag(AR2MLE$var.coef)))
colnames(Yule)<- c("estimates", "se")
colnames(CSS)<- c("estimates", "se")
colnames(MLE)<- c("estimates", "se")

list(Yule = Yule, CSS= CSS, MLE = MLE)
```

```
## $Yule
##      estimates       se
## [1,] 0.8958155 1.125804
##
## $CSS
##           estimates        se
## ar1       0.7997956 0.1007325
## ar2       0.1315714 0.1004847
## intercept 1.5490937 0.9038945
##
## $MLE
##           estimates         se
## ar1       0.7966775 0.09919866
## ar2       0.1187893 0.10001431
## intercept 0.8420766 1.13594903
```

```r
ci<- c(MLE[2,1]-1.96*MLE[2,2], MLE[2,1]+1.96*MLE[2,2])
```

Given that the true values for ϕ are 0.8 and 0.1, the one that is nearer to the truth is the MLE method, with a slightly better ϕ_2 estimate than the CSS method. The Yule Walker method is just able to find an AR(1) model with the data provided. Nevertheless, I have not restricted my Yule Walker to look for models of AR(2) whereas we did it in the CSS and MLE ones. It is possible to just find an AR(1) model given how small theϕ_2 is and that when AIC is used, simpler models are preferred for similar AIC than those that are more complex (e.g.AR(2). As shown in the table below, the value for ϕ_2 fall within confidence interval for ML estimate for the 95% confidence interval (at the very edge of it but still inside), being:

```r
print(ci)
```

```
## [1] -0.07723877  0.31481731
```

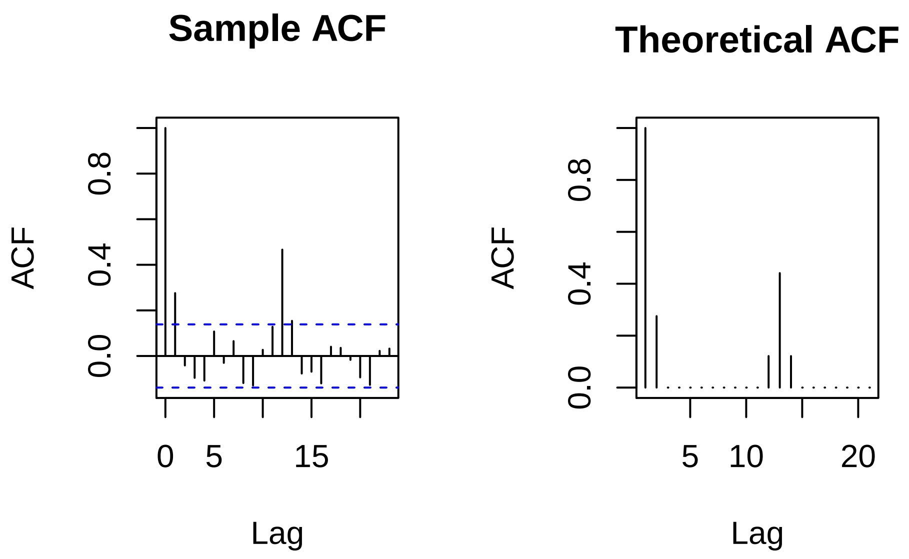
c)

In this question, we generated 200 obervations of a seasonal ARIMA $(0, 0, 1) \times (0, 0, 1)_{12}$ with coefficients $\Theta = 0.6$ and $\theta = 0.3$. In order to input the coefficient vector into `arima.sim()`, we first developped the polynomial in the model as follows:

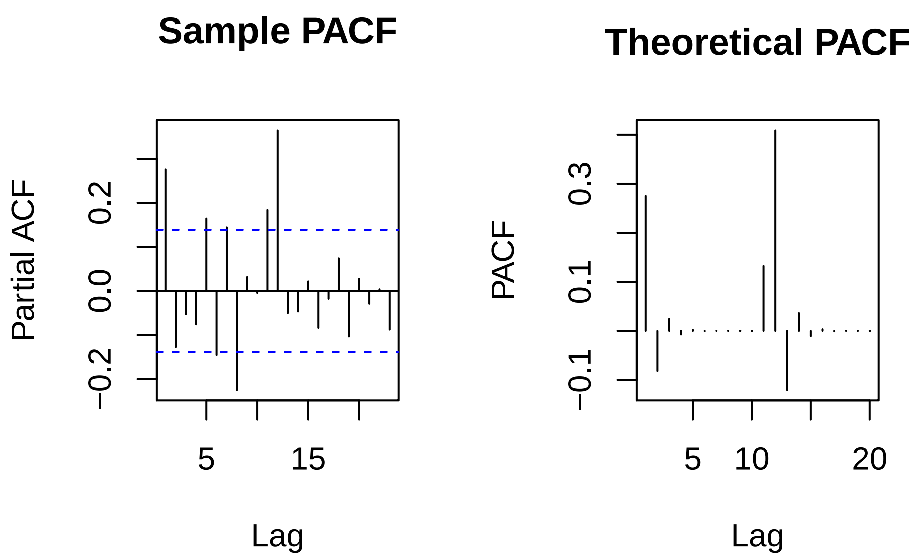
$$x_t = (1 + 0.3B)(1 + 0.6B^{12})w_t = (1 + 0.3B + 0.6B^{12} + 0.18B^{13})w_t$$

After simulating, we then plotted the sample and theoretical ACF and PACF, as shown below:

```r
THETA <- 0.6
theta <- 0.3
sma1_model <- list(ma = c(theta, rep(0, 10), THETA, theta*THETA))
set.seed(12345)
sma1_ts <- arima.sim(sma1_model, 200)

#sample/theoretical acf
par(mfrow = c(1,2))
acf(sma1_ts, main = "Sample ACF")
sma1_thACF <- ARMAacf(ma = sma1_model$ma, lag.max = 20)
plot(sma1_thACF, type = "h", main = "Theoretical ACF", xlab = "Lag", ylab = "ACF")
```



```r
#sample/theoretical pacf
pacf(sma1_ts, main = "Sample PACF")
sma1_thPACF <- ARMAacf(ma = sma1_model$ma, lag.max = 20, pacf = TRUE)
plot(sma1_thPACF, type = "h", main = "Theoretical PACF", xlab = "Lag", ylab = "PACF")
```

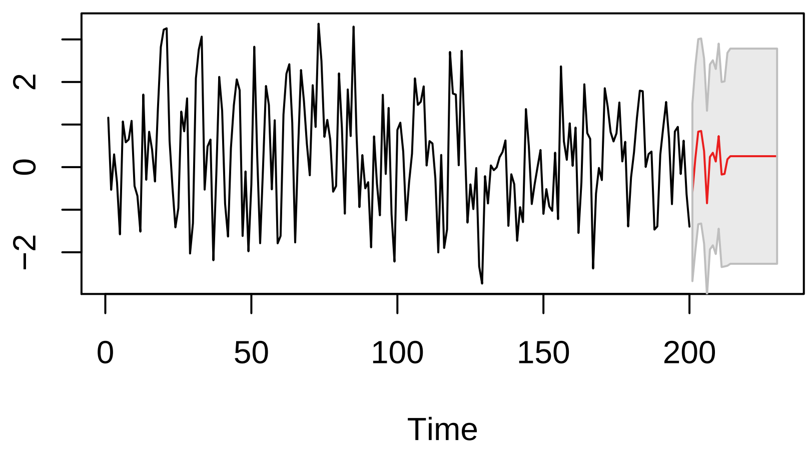


Given the graphs displayed, we can see in the theoretical ACF that the autocorrelation cuts off after lag 13. This is of course what we would expect from a MA(13) process, which the seasonal model is equivalent to as shown previously. The PACF shows seasonal peaks every approximately every 12 lags. The sample ACF shows the same autocorrelation peaks as the theoretical ACF with some white noise in between. The sample PACF shows also the same but with more significant noise in between. This makes sense since the sampling process was bound to generate random noise.

d)

Next we used the `arima()` function to fit the parameters based on the simulated seasonal MA timeseries from c), computed forecasts for up to 30 steps ahead. The forecasts are shown in the plot below as a continuation of the original data, along with the 95% prediction band.

```
sma1_fit <- arima(sma1_ts, order = c(0, 0, 1), seasonal = list(order=c(0, 0, 1), period=12))
sma1_pred <- predict(sma1_fit, n.ahead = 30)
ts.plot(sma1_ts, sma1_pred$pred, col = 1:2, xlim = c(1, 230))

#prediction band (from code5.R)
U <- sma1_pred$pr + 1.96*sma1_pred$se
L <- sma1_pred$pr - 1.96*sma1_pred$se
xx=c(time(U),rev(time(U)))
yy=c(L,rev(U))
polygon(xx,yy, border=8, col=gray(0.6, alpha=0.2))
```

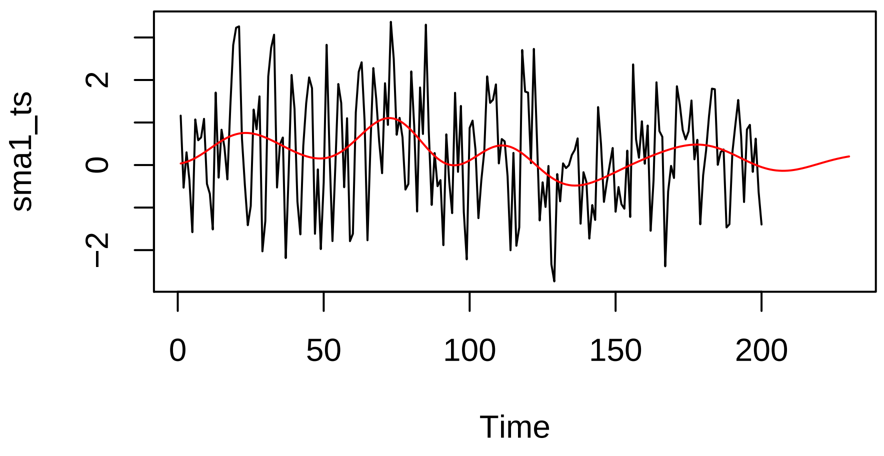


we also fitted the original time series with a gaussian process model using `gausspr()` from the `kernlab` package. The fit, as well as the forecasts for 30 steps ahead, are shown in the plot below:

```r
library(kernlab)
sma1_df <- data.frame(t = 1:200, xt = as.vector(sma1_ts))
sma1_fit2 <- gausspr(x=sma1_df$t, y=sma1_df$xt)
```

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

```r
sma1_pred2 <- predict(sma1_fit2, newdata = data.frame(1:230))
plot(sma1_ts, xlim = c(1, 230))
lines(sma1_pred2, col="red")
```



We can see that the 2 models give very different predictions. The ARIMA forecasts seem appropriate for the first 10 steps, with a similar pattern to the training time series; however after that the prediction breaks down and converges to 0. The GP model however, never takes the varying nature of the data into account, rather modelling the overall trend. Therefore the forecasting follows in this vein, giving an extremely simplified and smooth prediction curve; yet it does not "break down" after 10 steps like the other model. Given these results, it may be wise to use the ARIMA prediction for precise, short term forecasting and use GP prediction for more generalised but longer term forecasting.

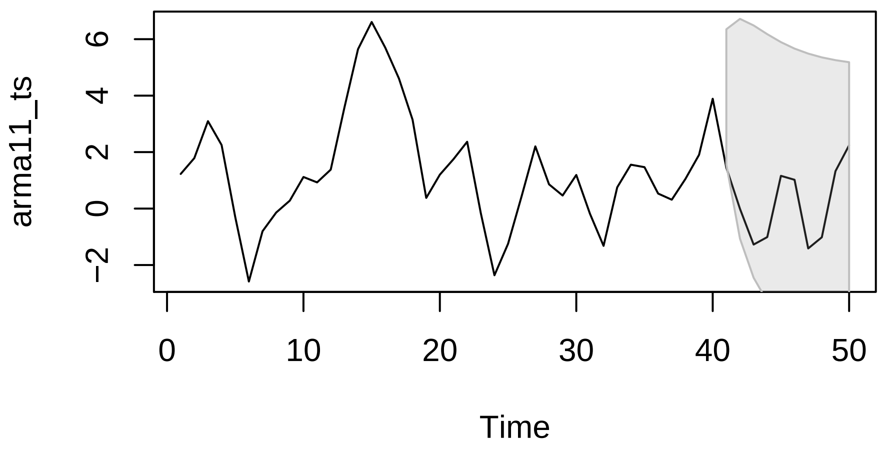
e)

In this question, we generated 50 observations from the ARMA(1,1) process shown below:

$$x_t = 0.7x_{t-1} + w_t + 0.5w_{t-1}$$

We then used the first 40 observations to fit the parameters of this model, with mean 0. Finally we used this model to forecast steps 41-50 and compared to the true observations. The totality of the origianl data is plotted below, with the 95% prediction band obtained with this prediction process:

```r
arma11_model <- list(ar = c(0.7), ma = c(0.5))
set.seed(12345)
arma11_ts <- arima.sim(arma11_model, n = 50)

arma11_fit <- arima(arma11_ts[1:40], order = c(1, 0, 1), include.mean = FALSE)

arma11_pred <- predict(arma11_fit, n.ahead = 10)
ts.plot(arma11_ts)

#prediction band (from code5.R)
U <-arma11_pred$pr + 2*arma11_pred$se
L <- arma11_pred$pr - 2*arma11_pred$se
xx=c(time(U),rev(time(U)))
yy=c(L,rev(U))
polygon(xx,yy, border=8, col=gray(0.6, alpha=0.2))
```



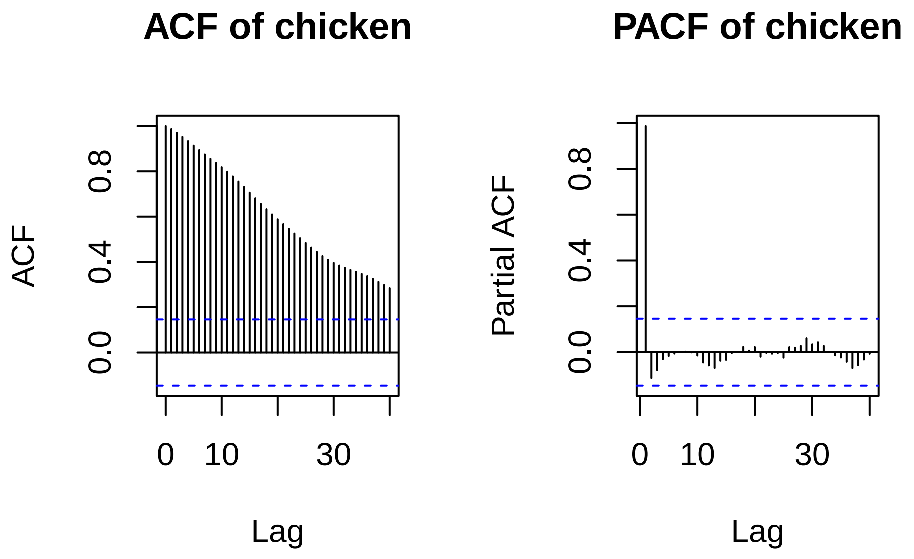
We can see that none of the true observations 41-50 fall outside the prediction band obtained using ARIMA forecasting, despite the high variance in this part of the data. Therefore the modelling was relatively successful at capturing the variation present in the data.

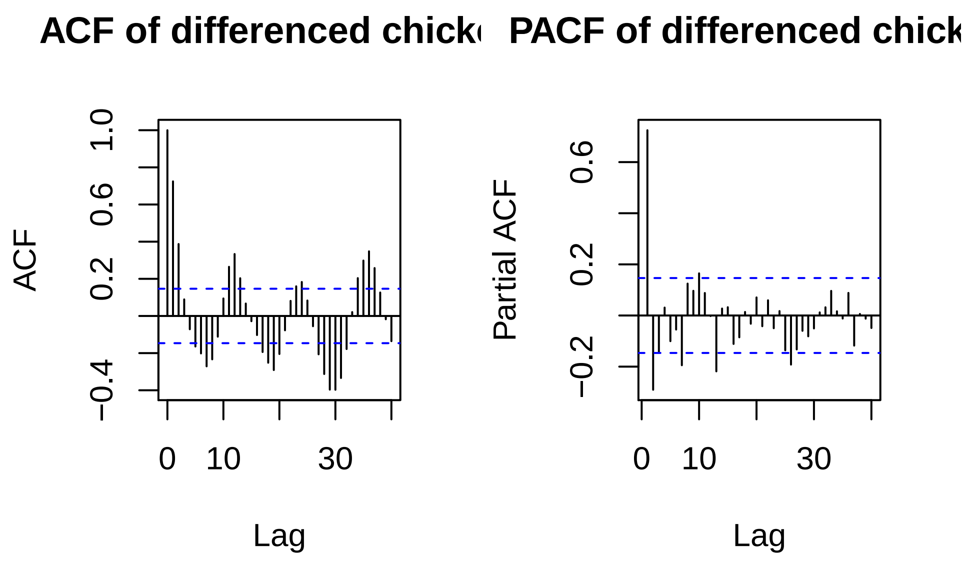
Assignment 2: ACF anf PACF diagnostics

In this assignment, we considered 4 different time series data sets in turn, all from the `astsa` package, and plotted the following quantities: $ACF(x_t)$, $PACF(x_t)$, $ACF(\nabla x_t)$ and $PACF(\nabla x_t)$, for up to 40 lags. The aim was to gauge the order of the possible ARIMA/SARIMA models from these plots.

Chicken data

```
library(astsa)
data("chicken")

ass2func <- function(x_t) {
  x_ts <- ts(x_t)
  #differencing
  dx_ts <- diff(x_ts, 1)
  #plots
  par(mfrow= c(1, 2))
  acf(x_ts, lag.max = 40, main = paste0("ACF of ", substitute(x_t)))
  pacf(x_ts, lag.max = 40, main = paste0("PACF of ", substitute(x_t)))
  acf(dx_ts, lag.max = 40, main = paste0("ACF of differenced ", substitute(x_t)))
  pacf(dx_ts, lag.max = 40, main = paste0("PACF of differenced ", substitute(x_t)))



}

ass2func(chicken)
```

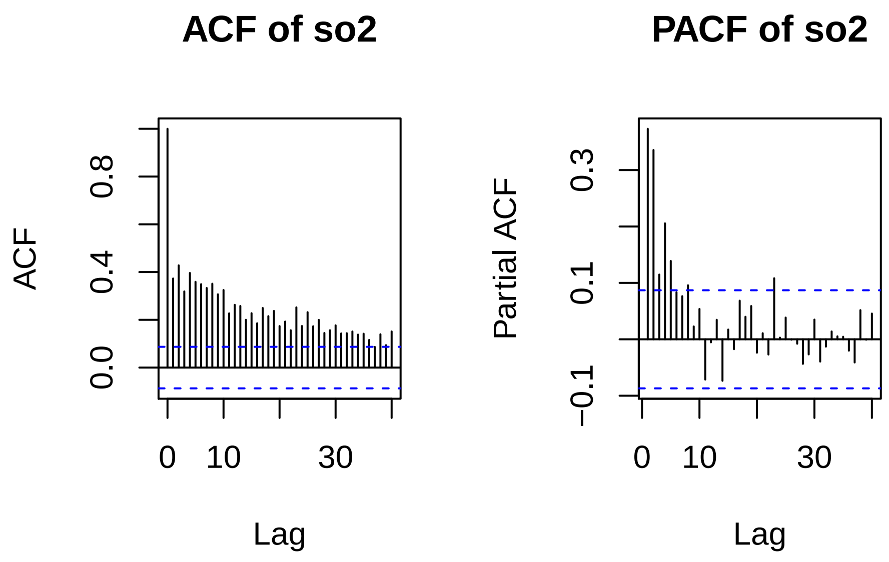


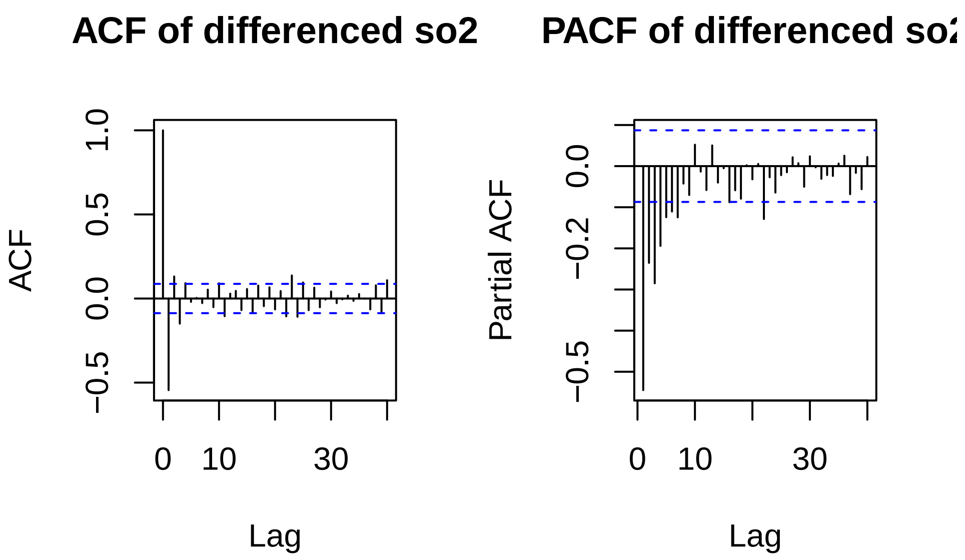


For the `Chicken` time series, we can see that the ACF decreases steadily while the PACF on the other hand, cuts off after lag 1. These are signs that the data could be modeled by a AR(1)/ARIMA(1, 0, 0) process. From the differenced plots however, neither the ACF nor the PACF seem to drop off, so the order of the ARIMA model is unclear. In addition, there is a pronounced seasonal pattern with s=12 present in the ACF.

SO2 data

```r
data(so2)
ass2func(so2)
```

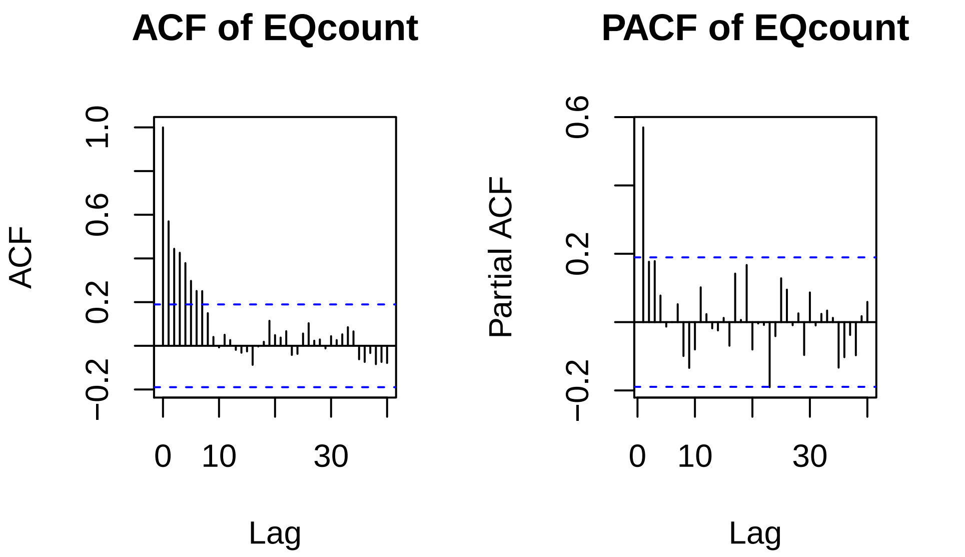


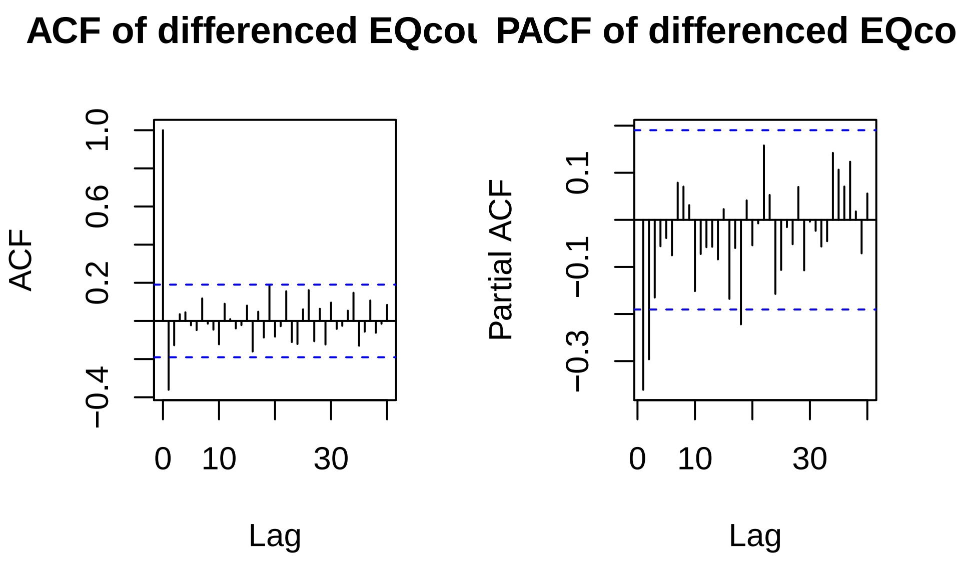


For the `so2` time series, both the ACF and PACF plots show tailing off rather than any cut-off point, making it difficult to gauge the order of the ARIMA model. The differenced time series however, has an ACF which cuts off after lag 1 and a PACF which tails off progressively: we would therefore suggest an ARIMA (0, 1, 1) model.

EQcount data

```r
data(EQcount)
ass2func(EQcount)
```

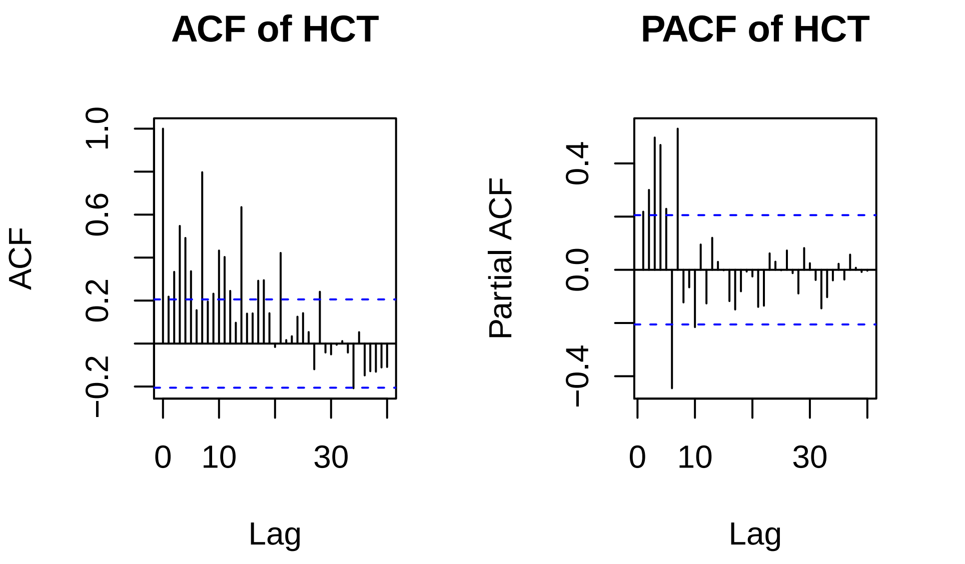


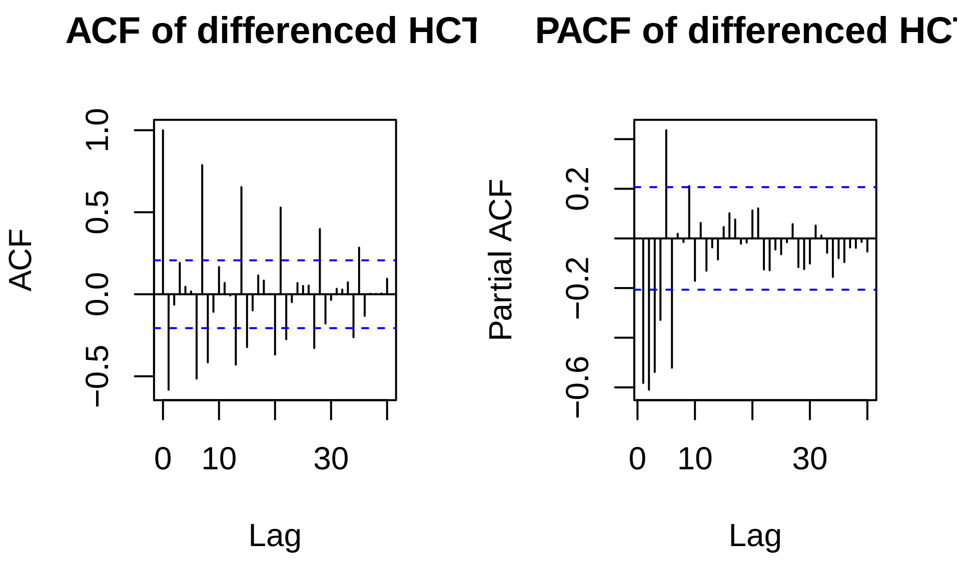


For the `EQcount` time series, the ACF decreases progressively to negligeable correlation values (within the blue band) while the PACF drops off immediately after lag 1. This would suggest that an AR(1)/ARIMA(1,0,0) model would be appropriate. The differenced time series however, shows the reverse: the ACF cuts off after lag 1, while the PACF tails off slowly. This suggests that an ARIMA(0,1,1) model would also be worth considering.

HCT data

```r
data(HCT)
ass2func(HCT)
```





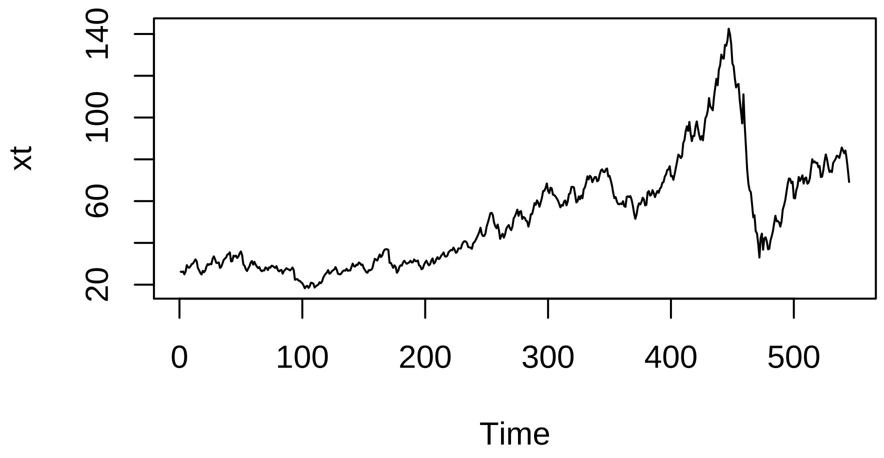
For the `HCT` time series, the ACF seems to indicate the presence of a seasonal pattern with s=7. However, within this period the ACF goes up and down, making it difficult to determine the order of the seasonal part of such a model. The PACF however cuts off after lag 7, indicating that the non-seasonal part may be AR(7)/ARIMA(7,0,0). The differenced time series data is a little easier to analyse. The ACF shows a clear seasonal pattern with s=6, with the ACF values cutting off after the 3rd peak in each cycle. The overall trend of the large peaks however, seems to steadily tail off. The PACF tails off steadily, without displaying any particular seasonal pattern. All this leads to the possibility that an appropriate model could be $ARIMA(6,1,0) \times (0,0,3)_6$.

Assignment 3:

1.

In this question, the aim was to find a suitable ARIMA(p,d,q) model for the `oil` data set. The first step was to simply plot the time series, as shown below:

```r
data(oil)
xt <- ts(oil)
plot(xt)
```



Immediately we can see that a transformation may be required to stabilise the variance in the data: we took the log transformation. We next studied the stationarity of our transformed data, using the Augmented Dickey-Fuller Test implementation in `adf.test()` of the `tseries` package. The output shown below would indicate that the transformed data is NOT stationary, and that differencing is required. We started with d=1.

```
logxt <- log(xt)
tseries::adf.test(logxt)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  logxt
## Dickey-Fuller = -2.775, Lag order = 8, p-value = 0.2503
## alternative hypothesis: stationary
```

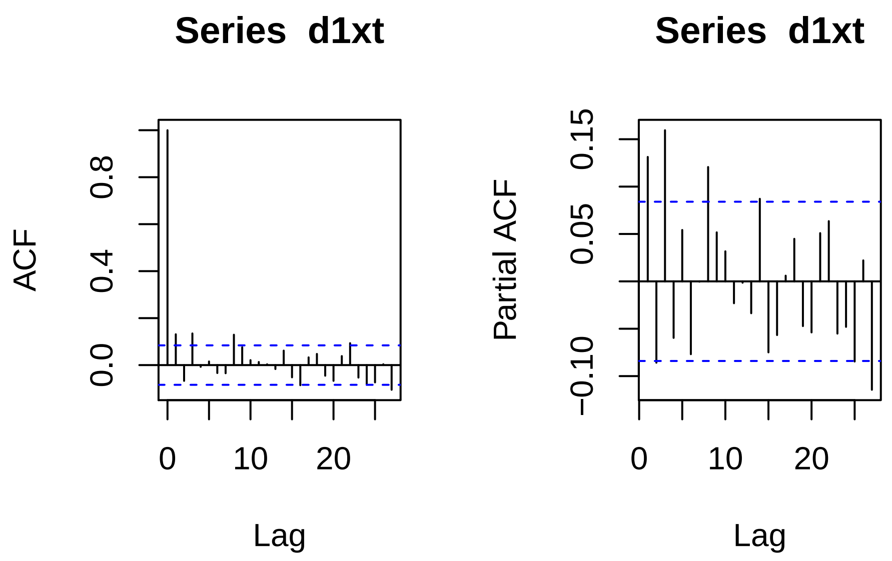
Below is the ADF test output for the differenced log data, which is shown to be stationary.

```
d1xt <- diff(logxt)
tseries::adf.test(d1xt)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  d1xt
## Dickey-Fuller = -6.3708, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

The next step was to examine the sample ACF and PACF plots of the transformed data, as shown below:

```
par(mfrow=c(1,2))
acf(d1xt)
pacf(d1xt)
```



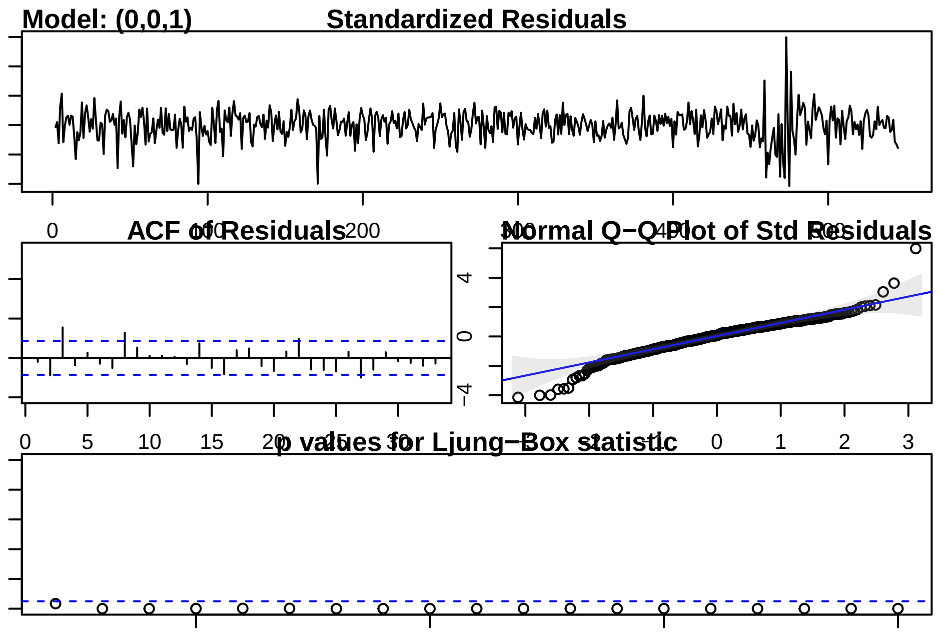
Neither the ACF nor the PACF show any cut-off points, we can say that the differenced log or growth rate of oil is ARMA(p, q) with p and q unknown. To find these orders, we next examined the EACF of this growth rate, as shown below:

```
library(TSA)
eacf(d1xt)
```

```
## 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
```

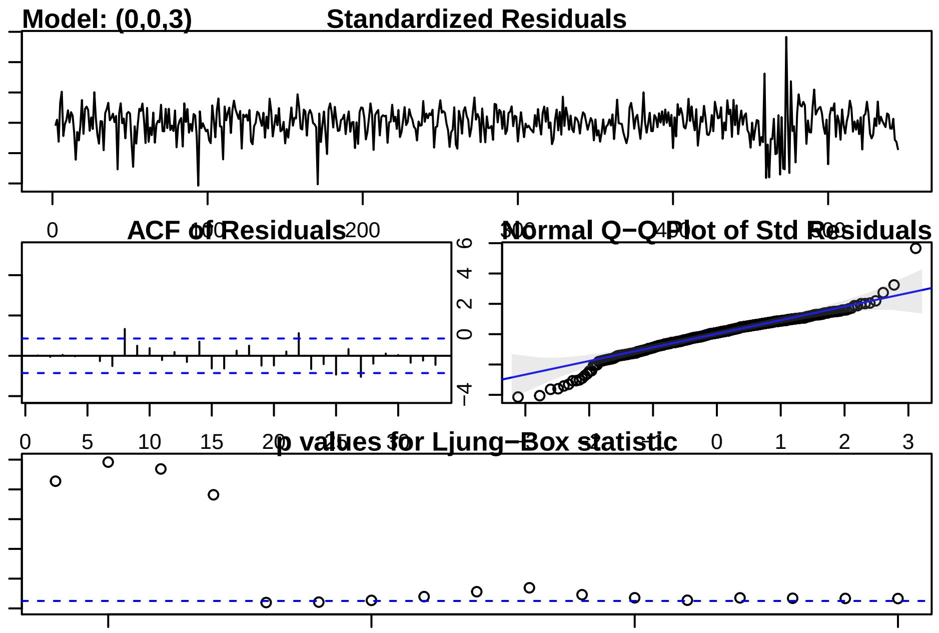
From this EACF, we could consider MA(1) and MA(3) as possible models. Now we have our 2 tentative models, the next part was to fit the parameters using `sarima()`. This function outputs several plots to aid residual analysis, including the Ljung-Box Test and QQ plots. Below is the output for the AR(1) model:

```
par(mar=c(1,1,1,1))
ma1_fit <- sarima(d1xt, 0, 0, 1, details = FALSE)
```



ACF of residuals: Few values are not in the blue band, which confirms our model assumptions. QQ plot: The normality of the residuals seems to be confirmed, bar some outliers at either end of the QQ line. Ljung-Box: The null hypothesis of residuals being independent is rejected for every lag, so this model residuals are not independent, indicating that MA(1) is not an appropriate model. Next for the MA(3) model:

```
 par(mar=c(1,1,1,1))
ma3_fit <- sarima(d1xt, 0, 0, 3, details = FALSE)
```



Here we can see that this MA(3) model. ACF of residuals: Few values are not in the blue band, which confirms our model assumptions. QQ plot: The normality of the residuals seems to be confirmed, bar some outliers at either end of the QQ line. Ljung-Box: The null hypothesis of residuals being independent is not rejected for lags 1-5, so this model residuals are independent to some degree. This MA(3) model seems to be a good fit for the differenced log oil data. Finally we looked the AIC and BIC values for both considered models:

```
ma1_fit$AIC
```

```
## [1] -5.131601
```

```
ma1_fit$BIC
```

```
## [1] -6.115796
```

```
ma3_fit$AIC
```

```
## [1] -5.160442
```

```
ma3_fit$BIC
```

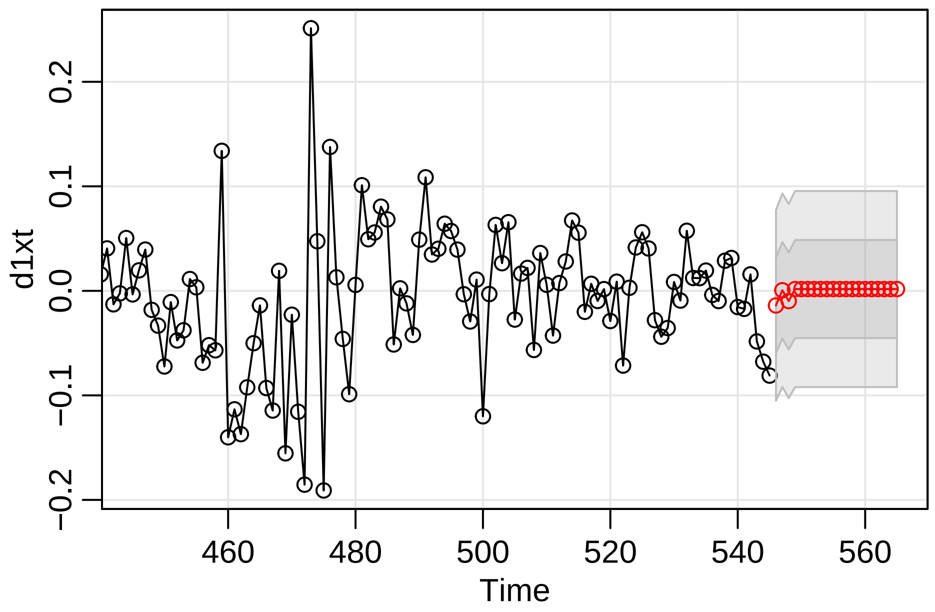
```
## [1] -6.128832
```

For both criterion, the MA(1) model outperform but only marginally. Given the Ljung-Box Test results, we must still pick so we pick the MA(3). The final model is therefore:

$$\nabla x_t = w_t + 0.1688 w_{t-1} + -0.09 w_{t-2} + 0.1447 w_{t-3}$$

Finally we used this model to forecast 20 steps ahead, as shown in the plot below:

```
ma3_pred <- sarima.for(d1xt, n.ahead = 20, 0, 0, 3)
```

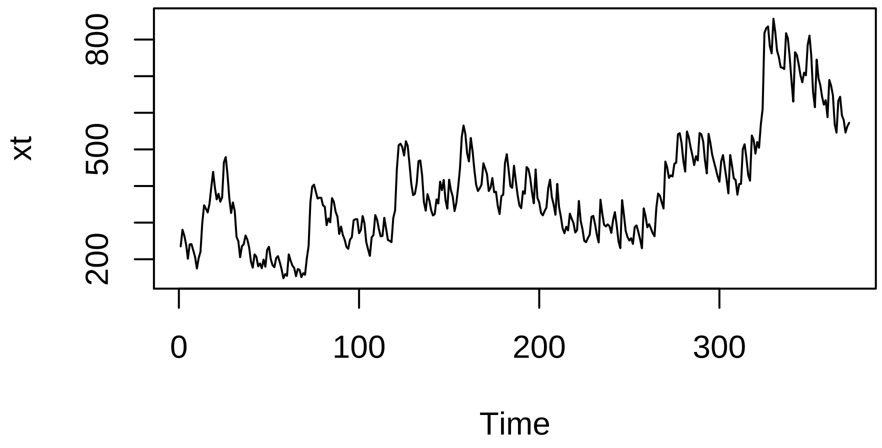


The prediction doesn't seem to capture the variance of the data, it is possible the model underfits the differenced log oil data.

2.

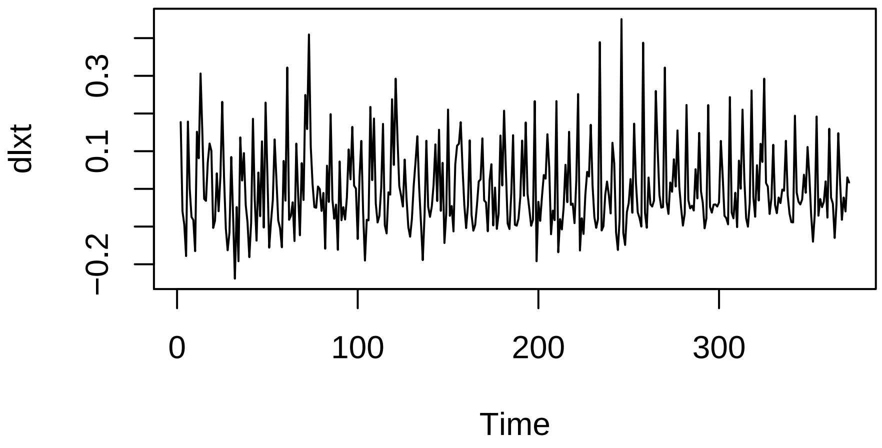
In this question, the aim was to find a suitable ARIMA(p,d,q) model for the `oil` data set. The first step was to simply plot the time series, as shown below:

```
data(unemp)
xt <- ts(unemp)
plot(xt)
```



Again we can see that a transformation may be required to stabilise the variance in the data and remove trend: we took the differenced log/growth rate transformation. As shown in the plot below the trend seems to have been successfully removed. We next studied the stationarity of our transformed data, using the Augmented Dickey-Fuller Test. The output shown below would indicate that the transformed data is stationary.

```r
logxt <- log(xt)
dlxt <- diff(logxt)
plot(dlxt)
```



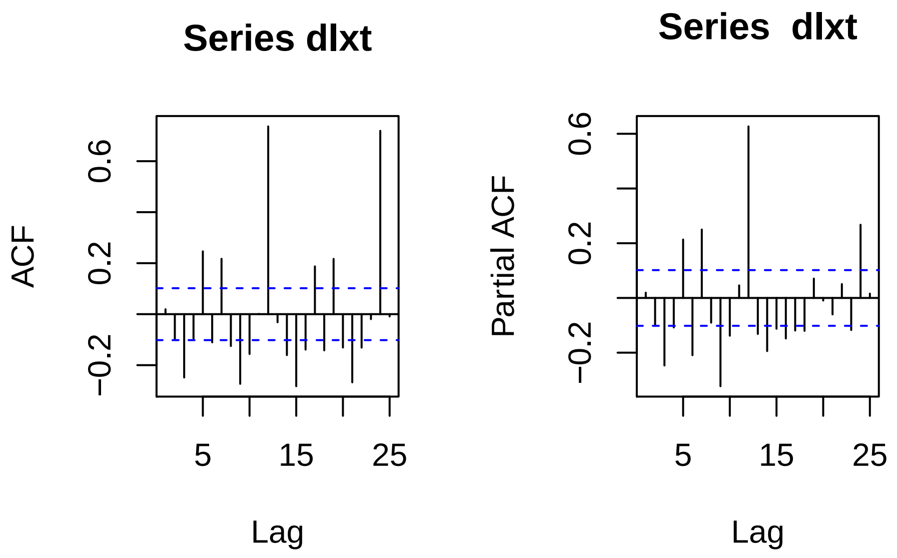
```r
tseries::adf.test(dlxt)
```

```
##
##  Augmented Dickey-Fuller Test
##
```

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

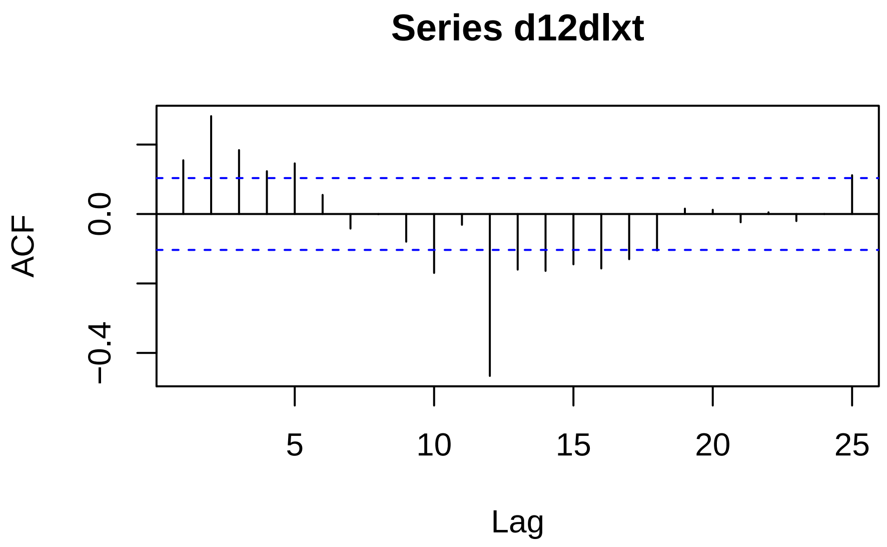
The next step was to examine the sample ACF and PACF plots of the transformed data, as shown below:

```
par(mfrow=c(1,2))
acf(dlxt)
pacf(dlxt)
```

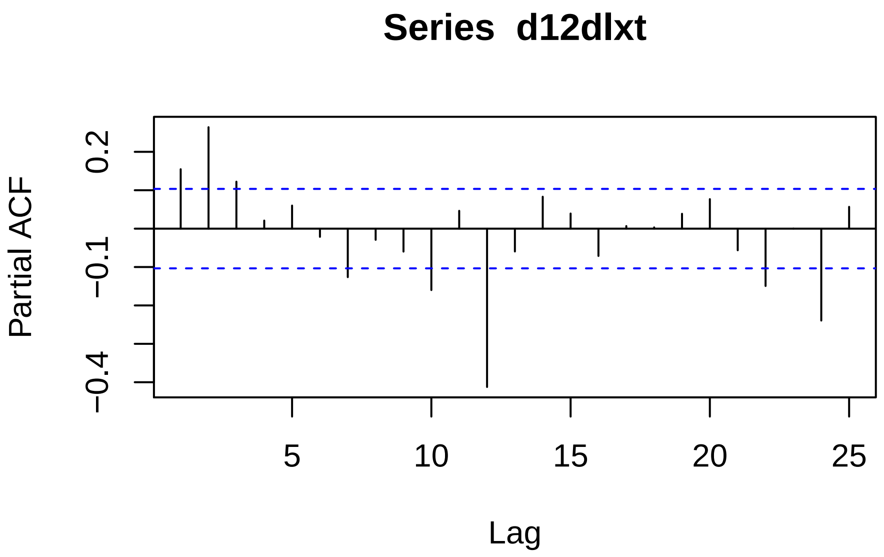


The ACF in particular shows a seasonal pattern with s=12 so we try seasonal differencing on the growth rate. Below are the corresponding ACF/PACF plots.

```
d12dlxt <- diff(dlxt, 12)
acf(d12dlxt)
```

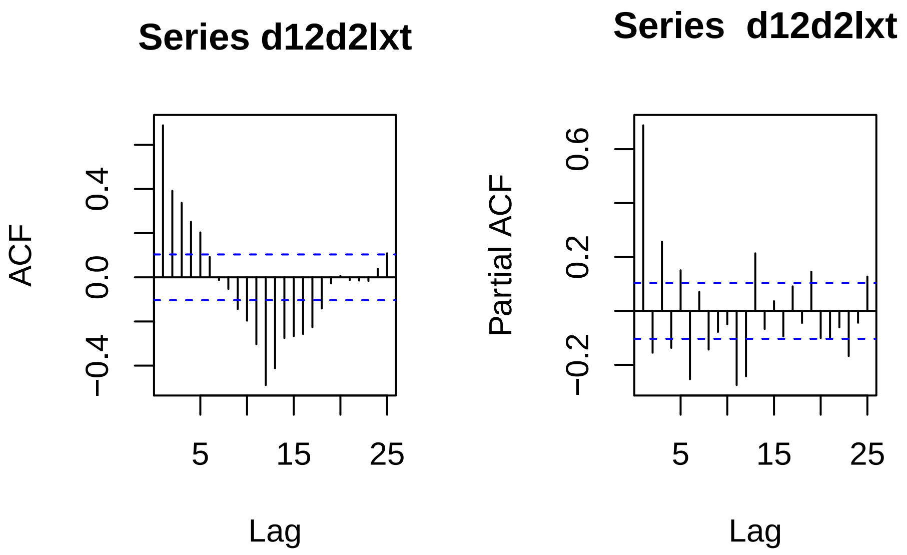


```r
pacf(d12dlxt)
```



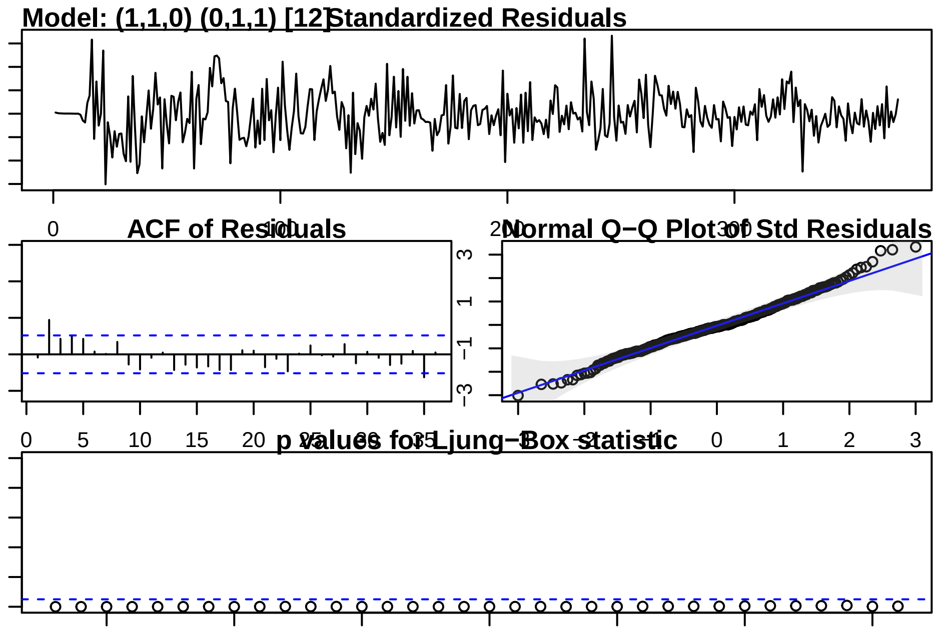
Looking at the ACF and PACF at lags proportional to the period 12, the ACF seems to cut off from 12 to 24, while the PACF between these 2 lags has decreased but less abruptly. This would suggest the seasonal component is SMA(1). Within the periods however, we observed the opposite: the ACF tails off while the PACF cuts off immediately after the 12th lag, suggesting that the non-seasonal component is AR(1). We put forward $ARIMA(1,1,0) \times (0,1,1)_{12}$ as a possible model for the log unemp data. To find a 2nd possible model, we tried a few different differencing and seasonal differencing orders until looking into $\nabla_{12}\nabla_2 log x_t$. The graphs below show the periodic ACF (just looking at lags 12, 24…) cuts off after the second period, while the PACF tails off. Within the periods however, both ACF and PACF seem to tail off. We tentatively suggest an $ARIMA(1,2,1) \times (0,1,2)_{12}$ model.

```r
d12lxt <- diff(log(xt), 12)
d2lxt <- diff(log(xt), 2)
d12d2lxt <- diff(d2lxt, 12)
par(mfrow=c(1,2))
acf(d12d2lxt)
pacf(d12d2lxt)
```

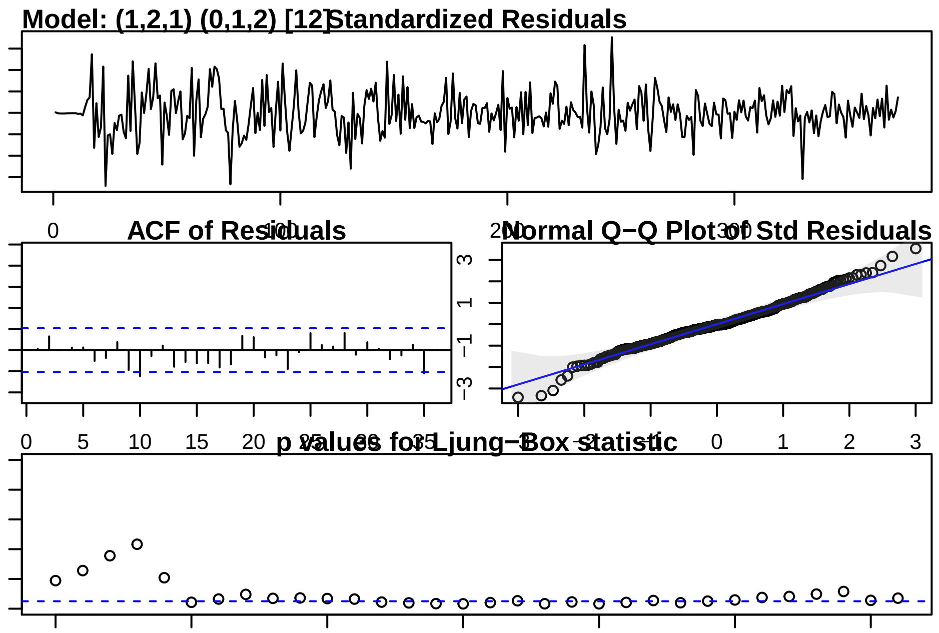


We next fitted both models and examined the residuals using `sarima()`

```r
par(mar=c(1,1,1,1))
fit1 <- sarima(logxt, 1, 1, 0, 0, 1, 1, S = 12, details=FALSE)
```



```r
par(mar=c(1,1,1,1))
fit2 <- sarima(logxt, 1, 2, 1, 0, 1, 2, S = 12, details=FALSE)
```



From these plots we can see that both model residuals show low ACF values and quasi-normal quantiles. The first model's Ljung-Box Test however, shows that the residuals are not independent, while the same test for the 2nd model shows that the residuals are independant up to lag 5. Therefore the 2nd model seems better according to these tests. Finally we compared the AIC and BIC scores of both models:

```r
fit1$AIC
```

```
## [1] -4.555215
```

```r
fit1$BIC
```

```
## [1] -5.534146
```

```r
fit2$AIC
```

```
## [1] -4.515402
```

```r
fit2$BIC
```

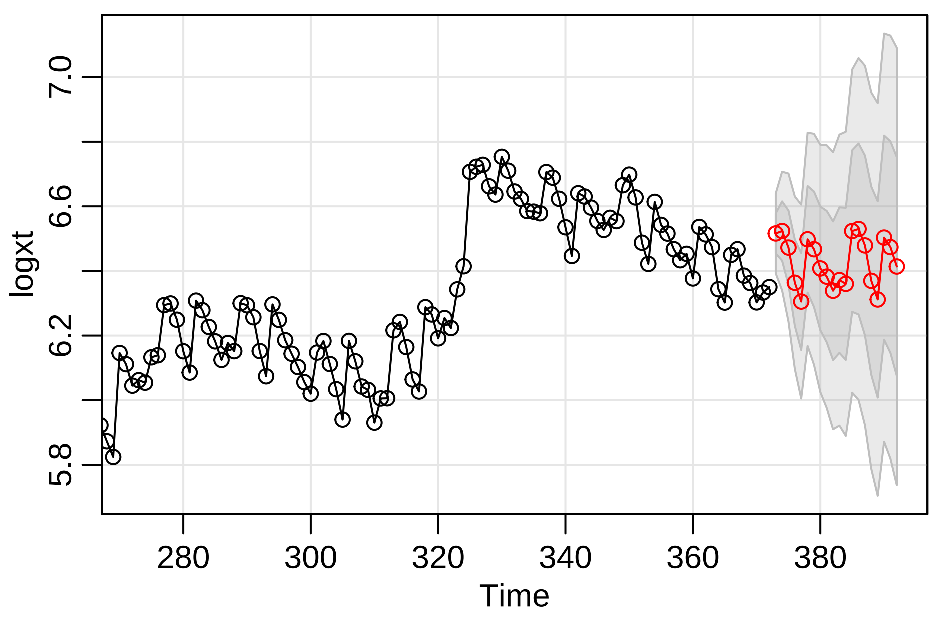
```
## [1] -5.462729
```

The scoring shows that the 2nd model not only is more appropriate given the tests, but also shows higher performance scores, despite higher complexity. The final model is therefore:

$$(1 - \phi B)\nabla_{12}\nabla x_t = (1 + \theta B)(1 + \Theta_1 B^{12} + \Theta_2 B^{24})w_t$$

where: $\theta = -0.8019$ $\Theta_1 = -0.7411$ $\Theta_2 = 0.0405$ $\phi = -0.0776$ Finally we used this model to forecast 20 steps ahead, as shown in the plot below:

```r
sarima_pred <- sarima.for(logxt, n.ahead = 20, 1, 1, 0, 0, 1, 2, S=12)
```



The prediction looks quite good, keeping the shape of the data and the 95% prediction band not too large at first.