

# Time Series Analysis - Lab 02 (Group 7)

Anubhav Dikshit (anudi287) and Maximilian Pfundstein (maxpf364)

2019-10-16

## Contents

<b>1</b>	<b>Assignment 1: Computations with simulated data</b>	<b>1</b>
1.1	Linear Regressions on Necessarily Lagged Variables and Appropriate Correlation . . . . .	1
1.2	Methods of Moments, Conditional Least Squares and Maximum Likelihood . . . . .	4
1.3	Sample and Theoretical ACF and PACF . . . . .	5
1.4	Forecast and Prediction . . . . .	7
1.5	Prediction Band . . . . .	8
<b>2</b>	<b>Assignment 2: ACF and PACF diagnostics</b>	<b>9</b>
2.1	ARIMA Model Suggestion . . . . .	9
2.2	More Datasets . . . . .	10
<b>3</b>	<b>Assignment 3: ARIMA modeling cycle</b>	<b>12</b>
3.1	Finding a Suitable ARIMA Model (oil) . . . . .	12
3.2	Finding a Suitable ARIMA Model (unemp) . . . . .	18
<b>4</b>	<b>Source Code</b>	<b>26</b>

## 1 Assignment 1: Computations with simulated data

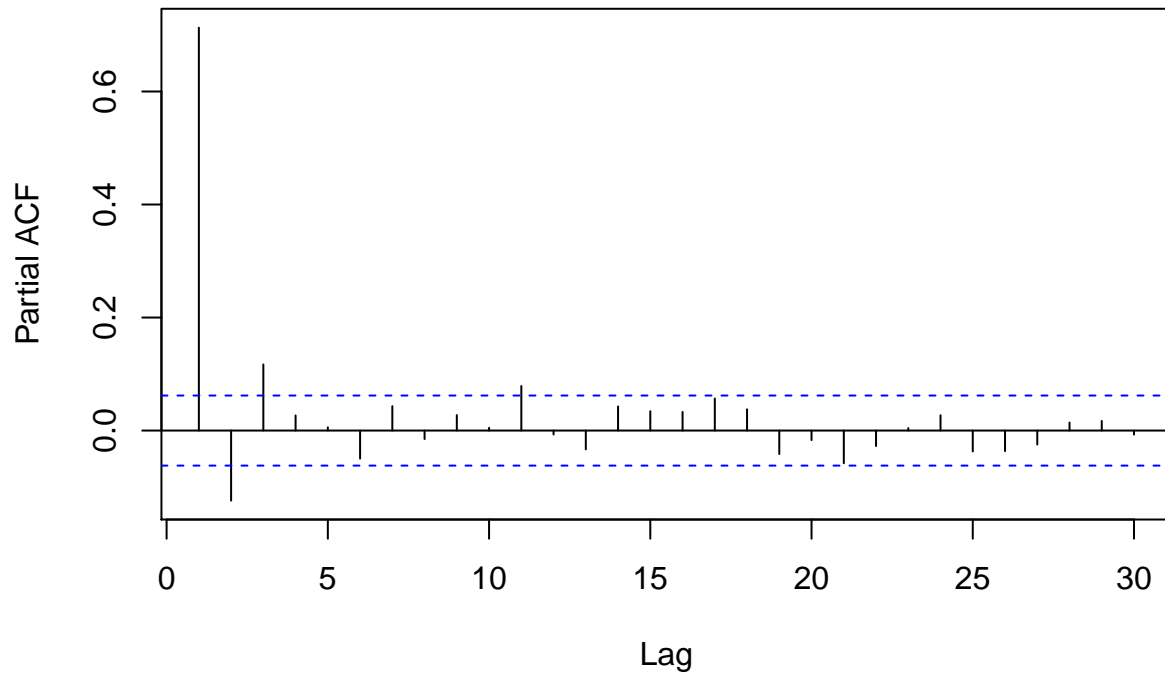
### 1.1 Linear Regressions on Necessarily Lagged Variables and Appropriate Correlation

**Task:** Generate 1000 observations from AR(3) process with  $\phi_1 = 0.8, \phi_2 = -0.2, \phi_3 = 0.1$ . Use these data and the definition of PACF to compute  $\phi_{33}$  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  $\phi_{33}$ .

**Answer:** First the sampling and looking at the built-in PACF.

```
model = list(ar = c(0.8, -0.2, 0.1), ma = c())
set.seed(12345)
series = arima.sim(model = model, n = 1000)
pacf(series)
print(pacf(series))
```

## Series series



```
##
## Partial autocorrelations of series 'series', by lag
##
##      1      2      3      4      5      6      7      8      9     10
## 0.713 -0.124 0.117 0.027 0.006 -0.050 0.043 -0.015 0.027 0.005
##     11     12     13     14     15     16     17     18     19     20
## 0.079 -0.007 -0.033 0.043 0.034 0.033 0.057 0.038 -0.042 -0.017
##     21     22     23     24     25     26     27     28     29     30
## -0.058 -0.027 0.004 0.027 -0.037 -0.036 -0.025 0.014 0.017 -0.007
```

Now we do it on our own. The following function does not only compute the value for a specific lag, but all lags up to given lag.max.

```
pacf_ar = function(series., lag.max = 30) {

  covariances = vector(length=lag.max)
  series = as.vector(series)

  for (lag in 1:lag.max) {

    # Create a dataframe with the lagged variables
    df = data.frame(y = series)
    df_colnames = c("y")

    if (lag == 1) {
      df = na.omit(cbind(df, lag(series, lag)))
      covariances[1] = cor(df[,1], df[,2])
      next
    }
  }
}
```

```

for (t in 1:(lag-1)) {
  df_colnames = c(df_colnames, paste("t_", t, sep=""))
  df = cbind(df, lag(series, t))
}

# Start at the right index (also omits NAs)
df = df[(1+lag):nrow(df),]
colnames(df) = df_colnames

# Second df
df2 = data.frame(y = series)
df2_colnames = c("y")

for (t in 1:(lag-1)) {
  df2_colnames = c(df2_colnames, paste("t+", t, sep=""))
  df2 = cbind(df2, lead(series, t))
}

# Start at the right index (also omits NAs)
df2 = df2[1:(nrow(df2)-lag),]
colnames(df2) = df2_colnames

# Performing LinReg
# We can take the residuals with intercept, as it does not affect correlation
x_t_dash = lm(y ~ ., df)$residual
x_t_dash_dash = lm(y ~ ., df2)$residual

covariances[lag] = cor(x_t_dash, x_t_dash_dash)
}
return(covariances)
}

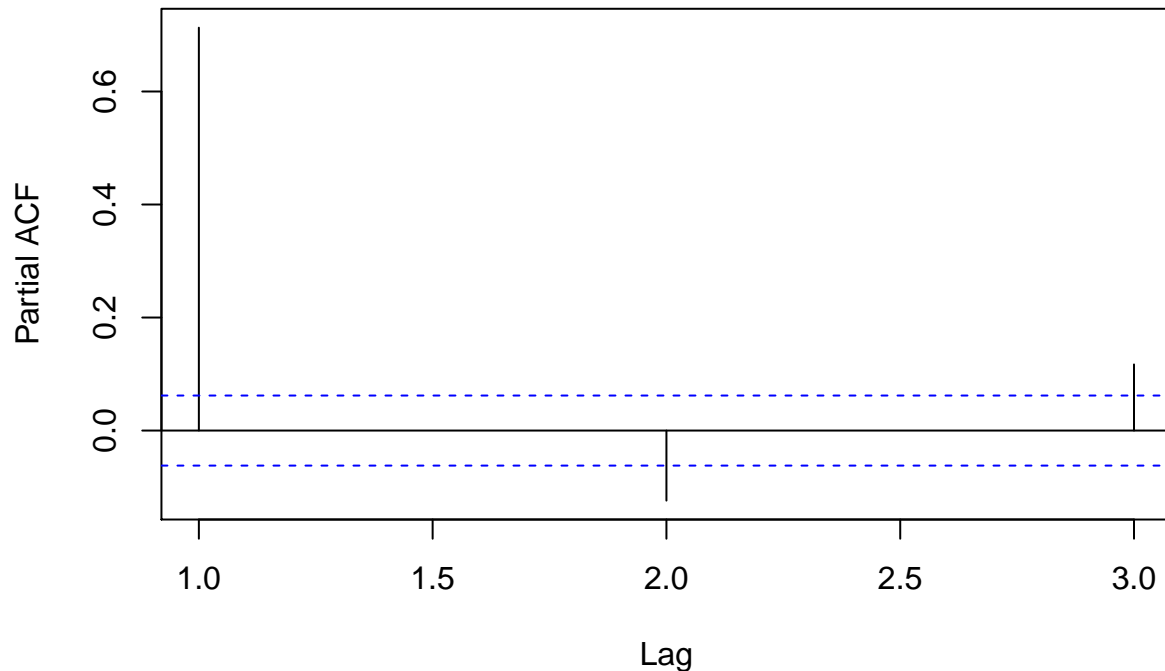
# Calculated
pacf_ar(series, 3)

## [1] 0.7139526 -0.1250914 0.1146076

# Function pacf
print(pacf(series, lag.max = 3))

```

## Series series



```
##
## Partial autocorrelations of series 'series', by lag
##
##      1      2      3
## 0.713 -0.124 0.117
```

```
# Theoretical
ARMAacf(model$ar, lag.max = 3, pacf = TRUE)
```

```
## [1] 0.7027027 -0.1212121 0.1000000
```

We see that all of the calculated values are slightly different, our own calculation differs a little bit more. We assume this is due to how exactly the metric is calculated. However, it's still close and shows almost the same values for higher lags, so we assume its correct.

## 1.2 Methods of Moments, Conditional Least Squares and Maximum Likelihood

**Task:** Simulate an AR(2) series with  $\phi_1 = 0.8, \phi_2 = 0.1$  and  $n = 100$ . Compute the estimated parameters and their standand errors by using three methods: method of moments (Yule-Walker equations), conditional least squares and maximum likelihood (ML) and compare their results to the true values. Which method does seem to give the best result? Does theoretical value for  $\phi_2$  fall within confidence interval for ML estimate?

**Answer:** Lets first simulate the time series and then fit the different models.

```
model = list(ar = c(0.8, 0.1), ma = c())
set.seed(12345)
series = arima.sim(model = model, n = 100)

MOM_Model = ar(series, order = 2, method = "yule-walker", aic = FALSE)
CLS_Model = ar(series, order = 2, method = "ols", aic = FALSE)
```

```
ML_Model = ar(series, order = 2, method = "mle", aic = FALSE)
```

```
df = data.frame(MOM_Model$ar, CLS_Model$ar, ML_Model$ar)
```

```
df
```

```
##      MOM_Model.ar CLS_Model.ar ML_Model.ar
## ar1      0.8029146    0.8066782    0.7968774
## ar2      0.1037053    0.1205352    0.1189369
```

It seems like the Methods of Moments works best in this case. Now lets look at the confidence interval.

As the function `ar()` does not seem to return the variance for the coefficients, we have to use the `arma()` function for that.

```
ML_Model_CI = arima(series, order = c(2,0,0), method = "ML")
```

```
sigma = ML_Model_CI$var.coef[2, 2]
phi_2 = ML_Model_CI$coef[2]
CI = c(phi_2 - 1.96 * sigma, phi_2 + 1.96 * sigma)
```

```
CI
```

```
##      ar2      ar2
## 0.09924714 0.13846032
```

The  $\phi_2$  estimate is 0.1189369, the CI is given by 0.0992471 for the lower boundary and 0.1384603 for the upper boundary. The `phi_2` estimate lies within the confidence interval.

### 1.3 Sample and Theoretical ACF and PACF

**Task:** Generate 200 observations of a seasonal ARIMA(0,0,1)  $\times$  (0,0,1)<sub>12</sub> model with coefficients  $\Theta = 0.6$  and  $\theta = 0.3$  by using `arma.sim()`. Plot sample ACF and PACF and also theoretical ACF and PACF. Which patterns can you see at the theoretical ACF and PACF? Are they repeated at the sample ACF and PACF?

**Answer:** For creating the model we have to rewrite the given seasonal ARIMA model into a *normal* one. As we only have the MA part,  $x_t$  is given by:

$$x_t = \Theta_Q(B^S)\theta(B)w_t$$

We know that  $q = 1$ ,  $Q = 1$  and  $S = 12$  and we know, given from the slides, that:

$$\Theta_q(B^S) = q + \Theta_1(B^{1S}) + \dots + \Theta_Q(B^{QS})$$

Therefore:

$$x_t = (1 + \Theta_1 B^{12})(1 + \theta_1 B)w_t$$

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

$$x_t = w_t + \theta_1 w_{t-1} + \Theta_1 w_{t-12} + \theta_1 \Theta_1 w_{t-13}$$

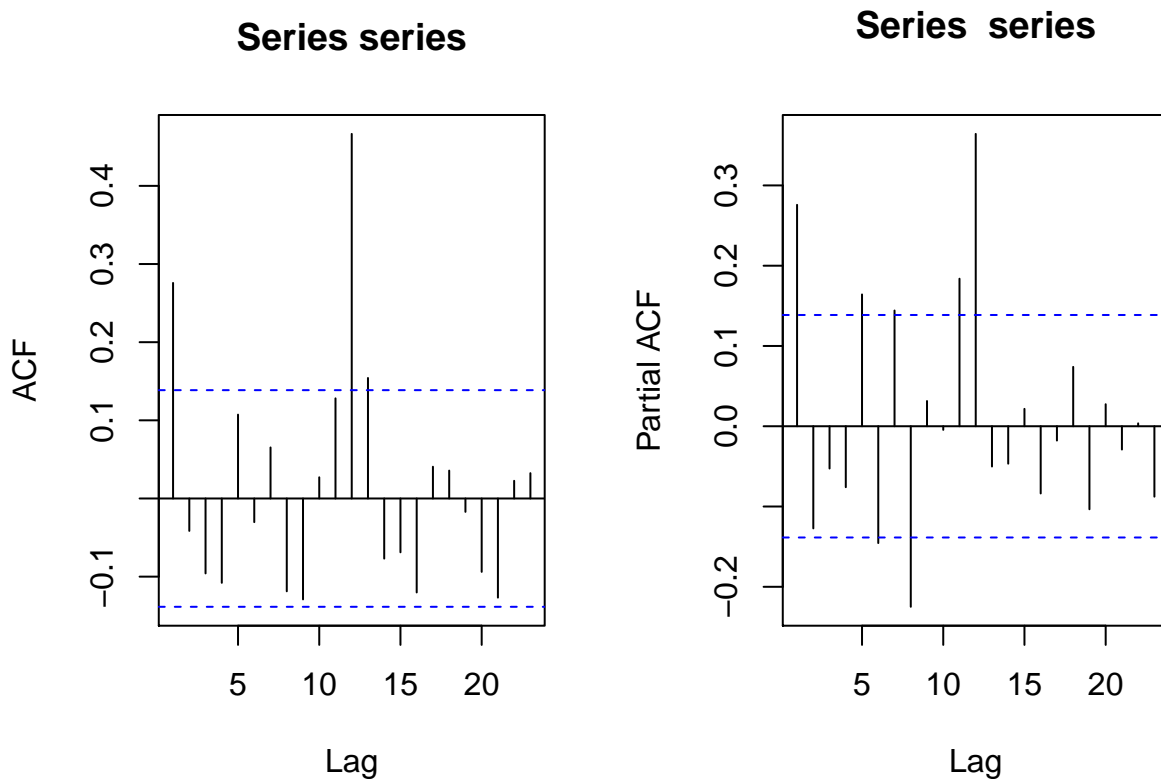
According to this the model is  $MA(\theta_1, \text{rep}(0, 0), \Theta_1, \theta_1 \Theta_1)$ .

Looking at the plots we see that the theoretical plot clearly shows the seasonality while the plot generated from the sample just indicates it. This is expected to some extent as we will have some white noise added to our samples. So we say that yes, they are repeated, but not as clearly visible as the theoretical one.

```
theta = 0.3
Theta = 0.6

model = list(ma = c(theta, rep(0, 10)), Theta, theta*Theta)
set.seed(12345)
series = arima.sim(model, n=200)

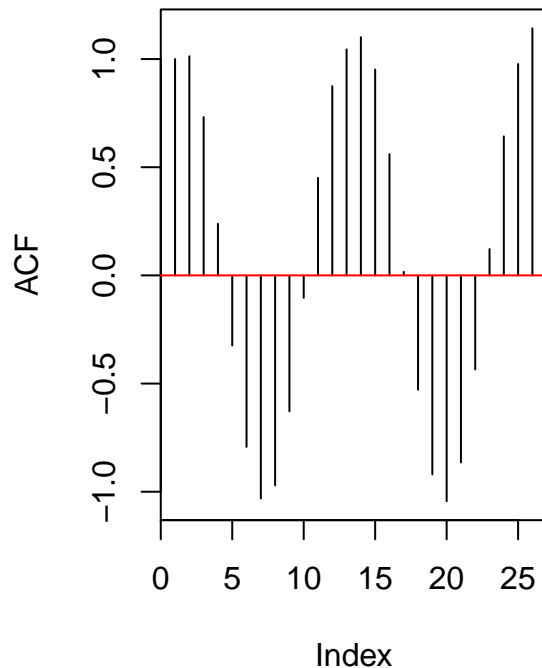
par(mfrow = c(1,2))
acf(series)
pacf(series)
```



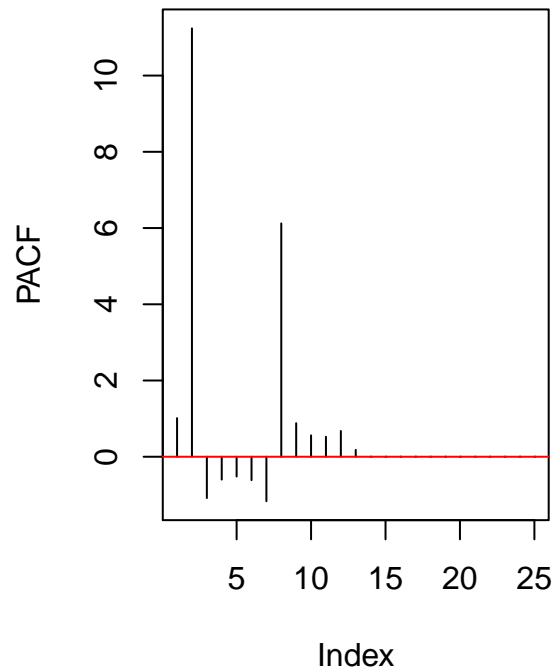
```
theoretical_acf = ARMAacf(model$ma, lag.max = 25, pacf = FALSE)
theoretical_pacf = ARMAacf(model$ma, lag.max = 25, pacf = TRUE)

par(mfrow = c(1,2))
plot(theoretical_acf, type = "h", main = "Theoretical ACF", ylab = "ACF")
abline(h = 0, col = "red")
plot(theoretical_pacf, type = "h", main = "Theoretical PACF", ylab = "PACF")
abline(h = 0, col = "red")
```

## Theoretical ACF



## Theoretical PACF



### 1.4 Forecast and Prediction

**Task:** Generate 200 observations of a seasonal  $\text{ARIMA}(0, 0, 1) \times (0, 0, 1)_{12}$  model with coefficients  $\Theta = 0.6$  and  $\theta = 0.3$  by using `arima.sim()`. Fit  $\text{ARIMA}(0, 0, 1) \times (0, 0, 1)_{12}$  model to the data, compute forecasts and a prediction band 30 points ahead and plot the original data and the forecast with the prediction band. Fit the same data with function `gausspr()` from package `kernlab` (use default settings). Plot the original data and predicted data from  $t = 1$  to  $t = 230$ . Compare the two plots and make conclusions.

**Answer:** First we fit the model to the series (it is the same as before) and create the predictions.

```
fitted_model = arima(series,
                      order = c(0, 0, 1),
                      seasonal = list(order= c(0, 0, 1), period = 12))

prediction = predict(fitted_model, n.ahead = 30)
```

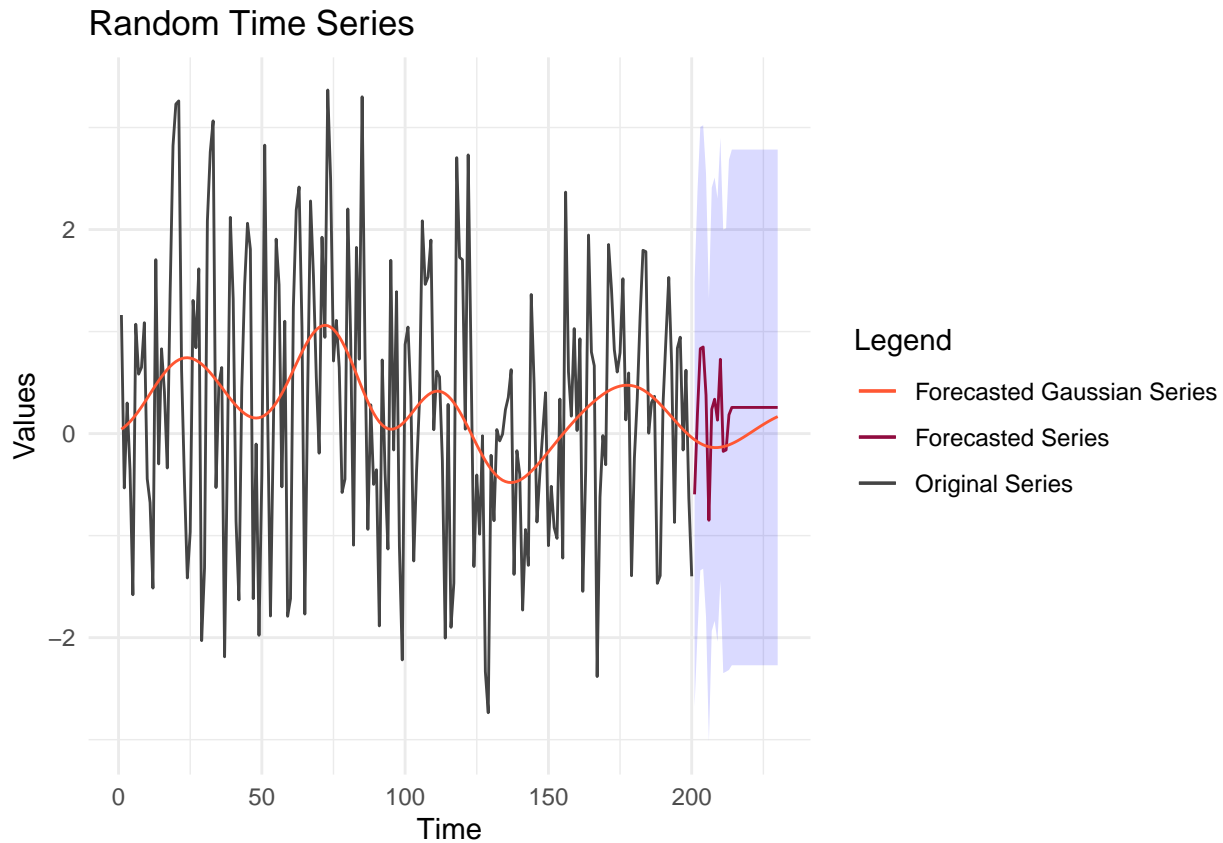
Now we fit using the `gausspr()` function and predict again.

```
fitted_model_gausspr = gausspr(c(1:200), series)

## Using automatic sigma estimation (sigest) for RBF or laplace kernel
prediction_gausspr = predict(fitted_model_gausspr, c(1:230))
```

As expected the Gaussian yields in a smoother graph, but this completely misses out all the important spikes. Our forecasted series seems to do a way better job in prediction the future data. We observe that the forecasted line becomes flat during the end.

## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.



## 1.5 Prediction Band

**Task:** Generate 50 observations from ARMA(1, 1) process with  $\phi = 0.7$ ,  $\theta = 0.50$ . Use first 40 values to fit an ARMA(1,1) model with  $\mu = 0$ . Plot the data, the 95% prediction band and plot also the true 10 values that you initially dropped. How many of them are outside the prediction band? How can this be interpreted?

**Answer:** All of the forecasted data lies within the prediction band. This is (mostly) expected, as we assume that just 5 percent of the data does not lie within the bands. For a prediction of 10 future values, the probability for having all values within the 95 percent prediction band is around  $0.95^{10} = 0.5987369$ . So with a probability around 40 percent we expect at least one data point outside the prediction band.

```
model = list(ma = c(0.7), ar = c(0.5))
set.seed(12345)
series = arima.sim(model, n=50)
fitted_model = arima(series[1:40], order = c(1, 0, 1), include.mean = FALSE)
prediction = predict(fitted_model, n.ahead = 10)
```

## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.





## 2 Assignment 2: ACF and PACF diagnostics

### 2.1 ARIMA Model Suggestion

**Task:** For data series `chicken` in package `astsa` (denote it by `x_t`) plot 4 following graphs up to 40 lags:  $ACF(x_t)$ ,  $PACF(x_t)$ ,  $ACF(\nabla x_t)$ ,  $PACF(\nabla x_t)$  (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.

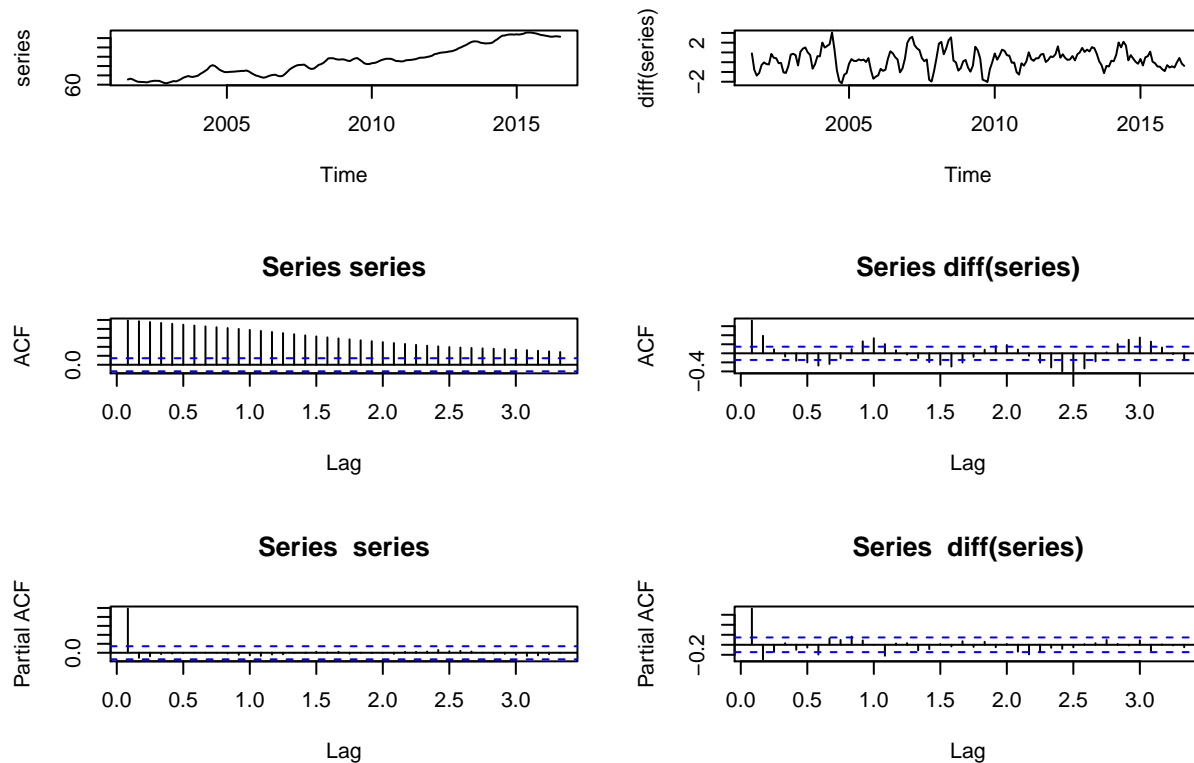
**Answer:** We will use this small helper function to create plots of the time series, including ACF and PACF, also for the first difference of the data.

```
plot_diagnostics = function(series, max.lag = 40) {
  par(mfrow = c(3, 2))
  plot(series)
  plot(diff(series))
  acf(series, lag.max = max.lag)
  acf(diff(series), lag.max = max.lag)
  pacf(series, lag.max = max.lag)
  pacf(diff(series), lag.max = max.lag)
}
```

**Answer:** Looking at the ACF plot we see a decreasing correlation over time. As the correlation is continually following a downwards trend, while all of the lags keep being statistically significant, we should have a closer look at the first difference (this is because the original series is not stationary). Here we see a spike at lag 1 and 2, as well as an indicator for the seasonality with a lag of 12. Looking at the PACF plots we can observe the seasonality again, also we still see a significant spike at lag 1 and 2. Therefore we choose the following

model:  $\text{ARIMA}(2, 1, 0) \times (1, 0, 0)_{12}$ .

```
plot_diagnostics(astsa::chicken)
```

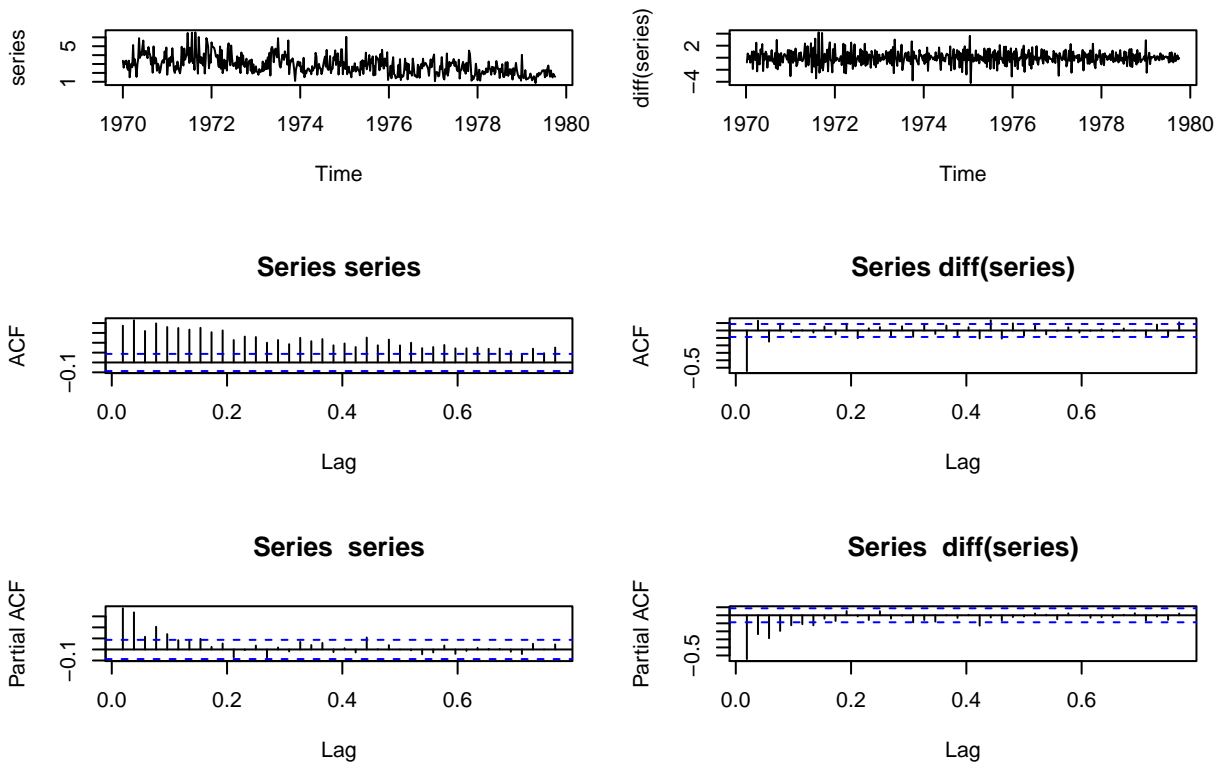


## 2.2 More Datasets

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

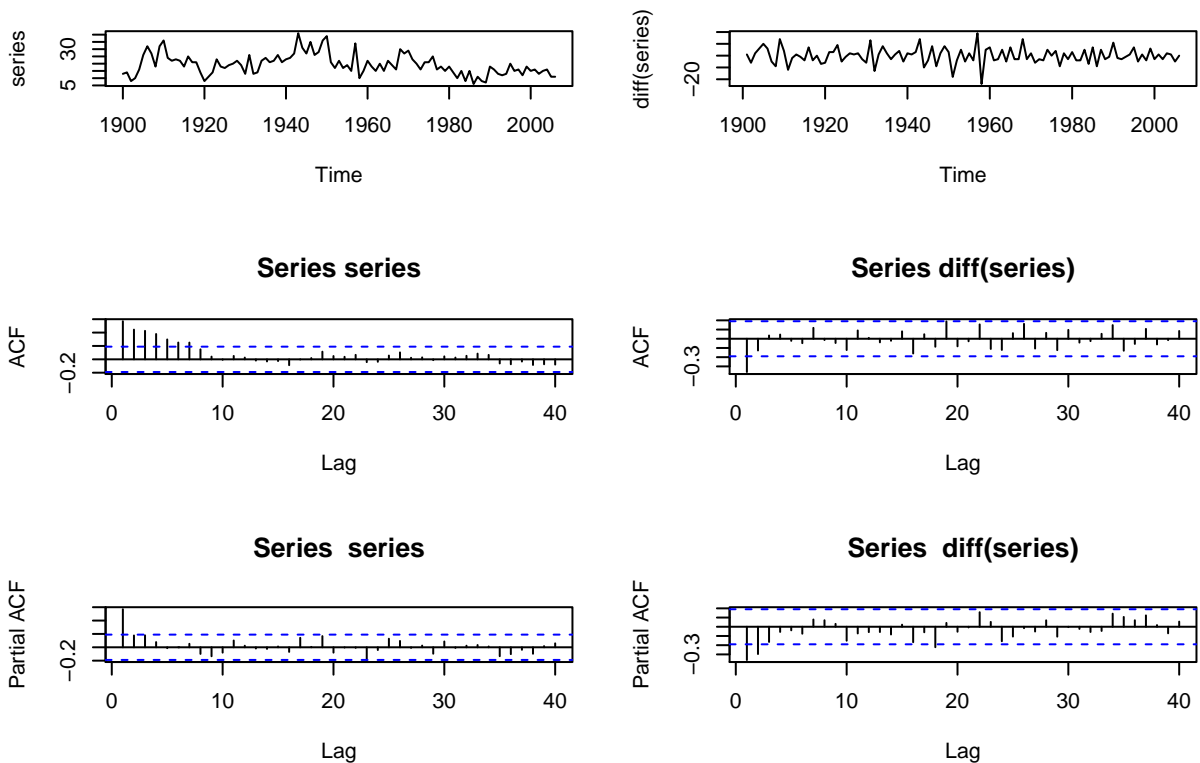
**so2:** We see again that it makes sense to take the first difference to make the process stationary, as the ACF plot only shows a really slow decay. For the differenced ACF plot, we see three or two four significant spikes, but only the first one seems to be strongly significant. The differentiated PACF plot also shows several spikes in the beginning. It does not seem that there is a seasonality, so the chosen model is  $\text{ARIMA}(0, 1, 1)$ .

```
plot_diagnostics(astsa::so2)
```



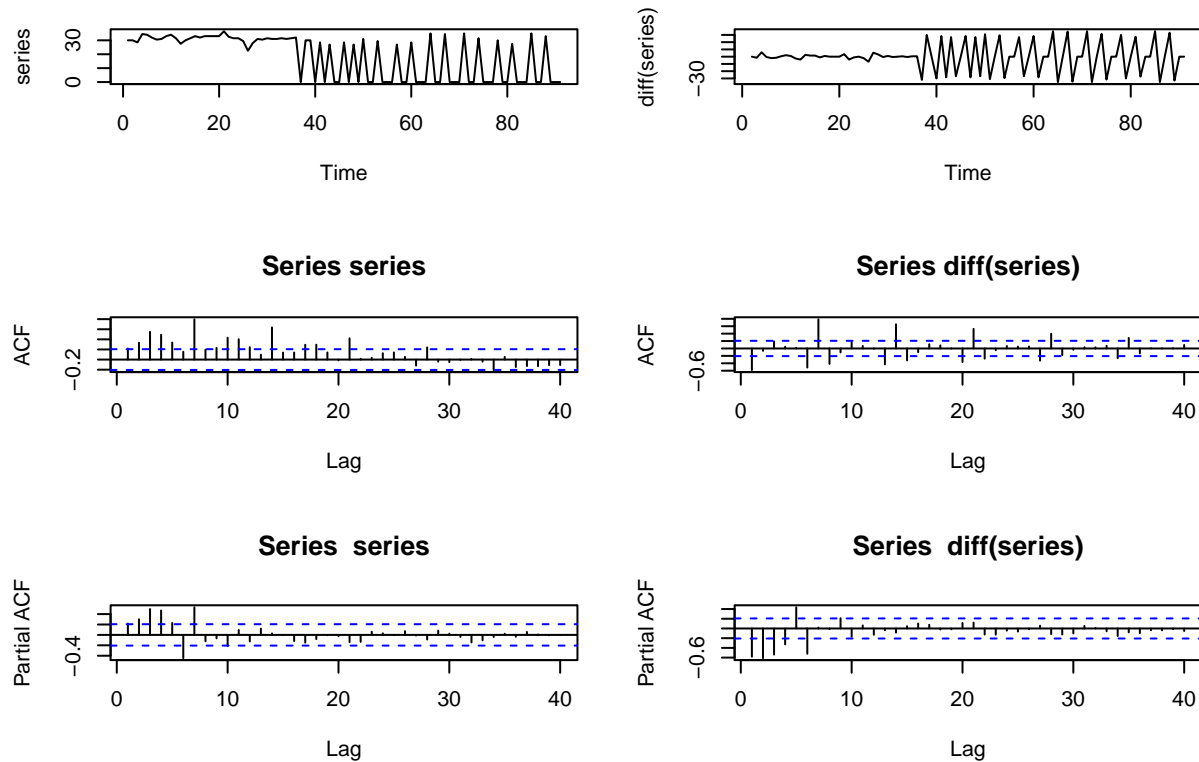
**EQcount:** We can see the typical behaviour of an AR process in the ACF plot, which is quickly decaying and we have no further increase in spikes. So we assume an underlying AR process. As we observe no seasonality, nor a major change in the differentiated plots, we chose the model  $AR(1)$ .

```
plot_diagnostics(astsa::EQcount)
```



**HCT:** The ACF plot of the first difference shows a slowly decaying difference, while having recurring spikes. As the spikes are separated by 7 lags, we assume a weekly seasonality. The PACF shows significance at lag 3, as well as a significance for lag 7. Therefore we suggest the model  $\text{ARIMA}(7, 1, 1) \times (0, 0, 1)_7$ .

```
plot_diagnostics(astsa::HCT)
```



### 3 Assignment 3: ARIMA modeling cycle

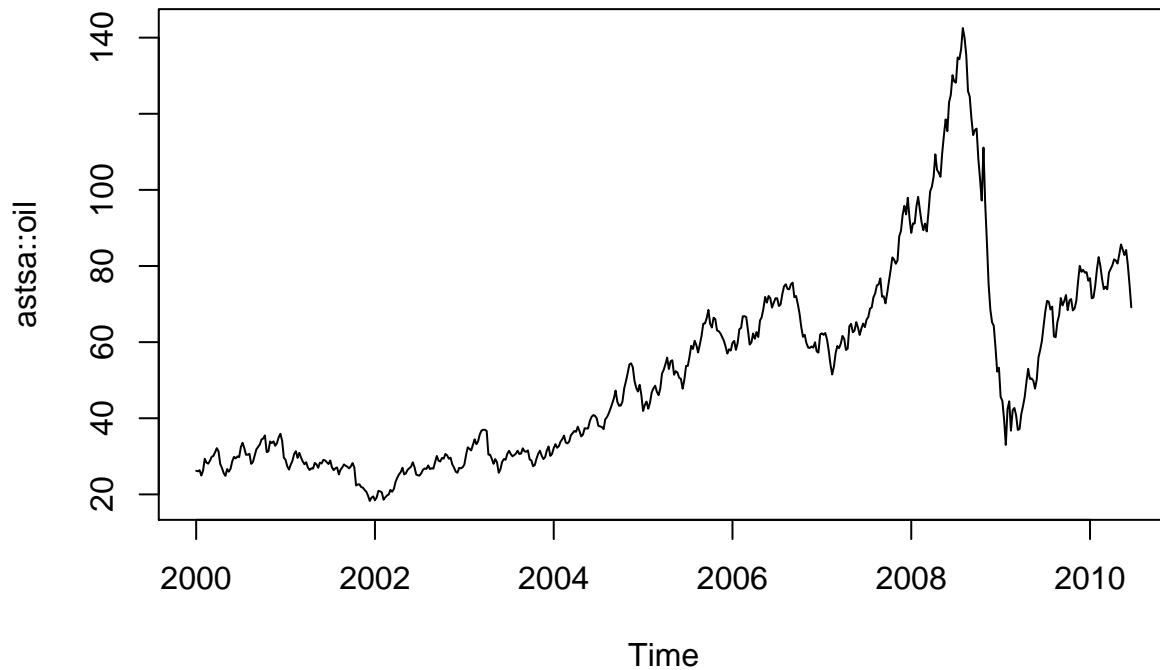
In this assignment, you are assumed to apply a complete ARIMA modeling cycle starting from visualization and detrending and ending up with a forecasting.

#### 3.1 Finding a Suitable ARIMA Model (oil)

**Task:** Find a suitable  $\text{ARIMA}(p, d, q)$  model for the data set `oil` present in the library `astsa`. Your modeling should include the following steps in an appropriate order: visualization, unit root test, detrending by differencing (if necessary), transformations (if necessary), ACF and PACF plots when needed, EACF analysis, Q-Q plots, Box-Ljung test, ARIMA fit analysis, control of the parameter redundancy in the fitted model. When performing these steps, always have 2 tentative models at hand and select one of them in the end. Validate your choice by AIC and BIC and write down the equation of the selected model. Finally, perform forecasting of the model 20 observations ahead and provide a suitable plot showing the forecast and its uncertainty.

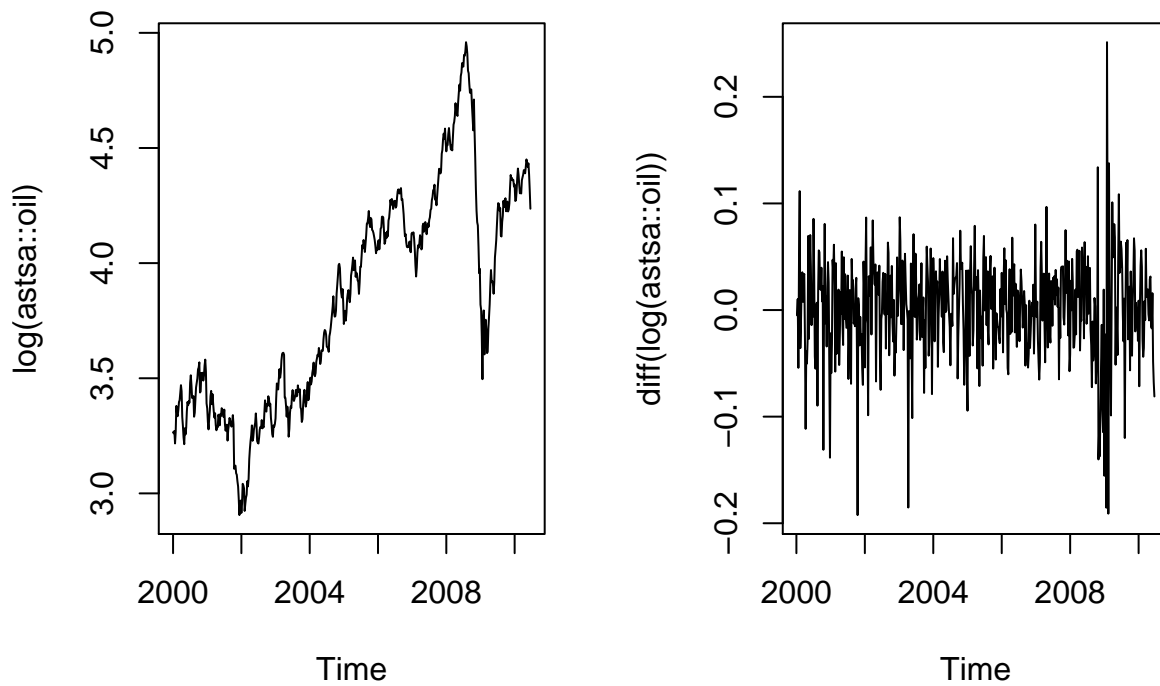
**Answer:** As a first step we plot the data and take a first look at it. We can see that the time series is not stationary, so we will have to take a look at the first difference to gain more insights. Also it looks like that the data and variance is growing exponentially, so we will log the data first (transformation), then taking the difference.

```
plot(astsa::oil)
```



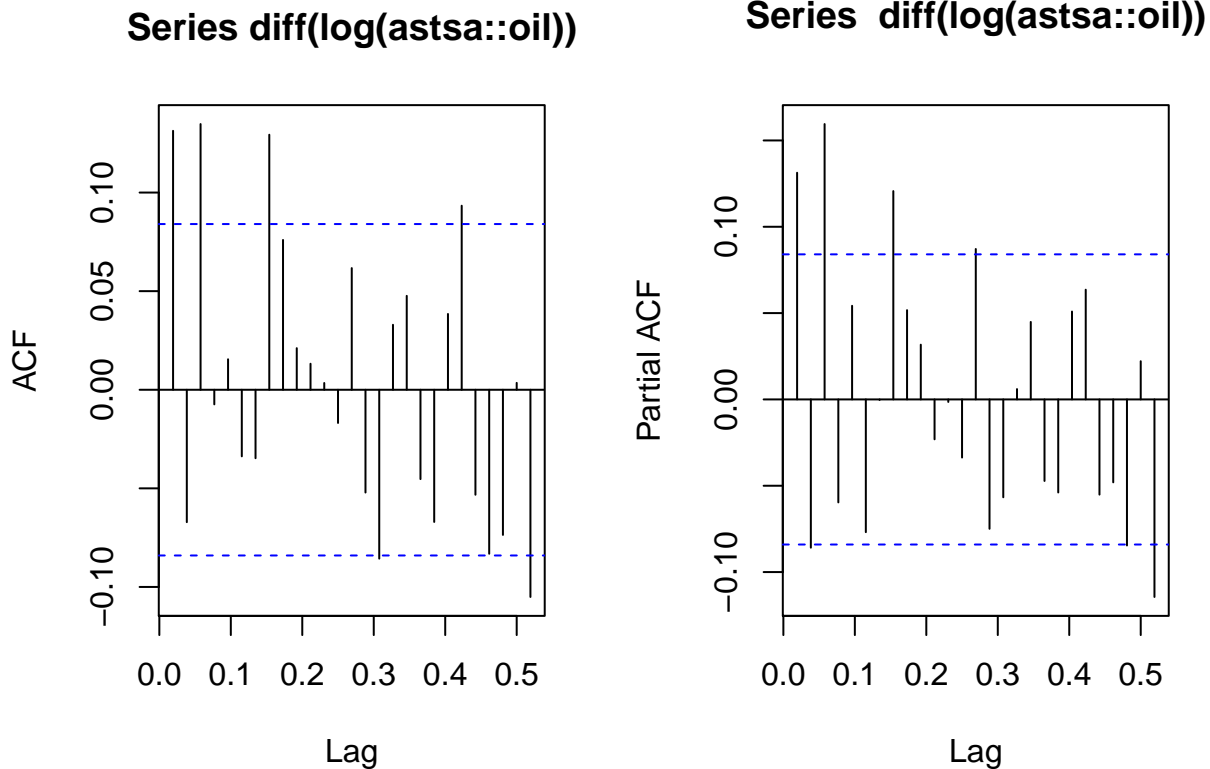
Looking at the first difference, we can now see that the time series seems to be stationary. We will take a look at the ACF and PACF plots to obtain more information about the correlation.

```
par(mfrow = c(1, 2))  
plot(log(astsa::oil))  
plot(diff(log(astsa::oil)))
```



```
par(mfrow = c(1, 2))  
acf(diff(log(astsa::oil)))
```

```
pacf(diff(log(astsa::oil)))
```



Now it is time to decide for two models which we will use. Therefore we will also consider information gained from the `eacf()` function.

```
eacf(diff(log(astsa::oil)))
```

```
## 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
```

We see a formed triangle at ARIMA(0,1,3). As we have two equivalent models with a higher order, namely ARIMA(0,1,4) and AIRMA(1,1,3), we will consider the following two models for the following analysis.

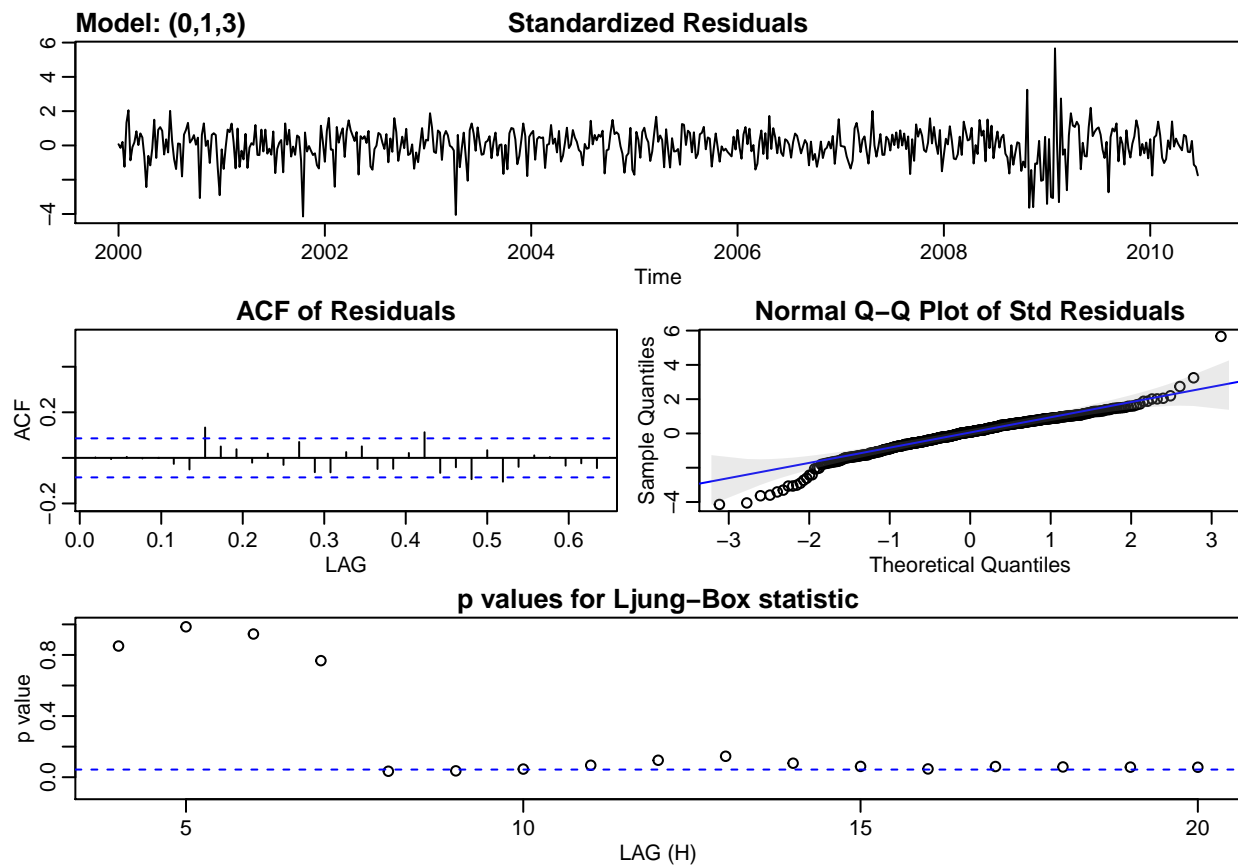
```
# We will use sarima as it will directly create the necessary plots
modelA = sarima(log(astsa::oil), p=0, d=1, q=3)
```

```
## initial value -3.058495
## iter 2 value -3.086110
## iter 3 value -3.086980
## iter 4 value -3.087501
## iter 5 value -3.087521
## iter 6 value -3.087521
## iter 7 value -3.087522
```

```

## iter    8 value -3.087522
## iter    9 value -3.087522
## iter    9 value -3.087522
## iter    9 value -3.087522
## final   value -3.087522
## converged
## initial value -3.087448
## iter    2 value -3.087448
## iter    3 value -3.087449
## iter    3 value -3.087449
## iter    3 value -3.087449
## final   value -3.087449
## converged

```



```

modelB = sarima(log(astsa::oil), p=1, d=1, q=3)

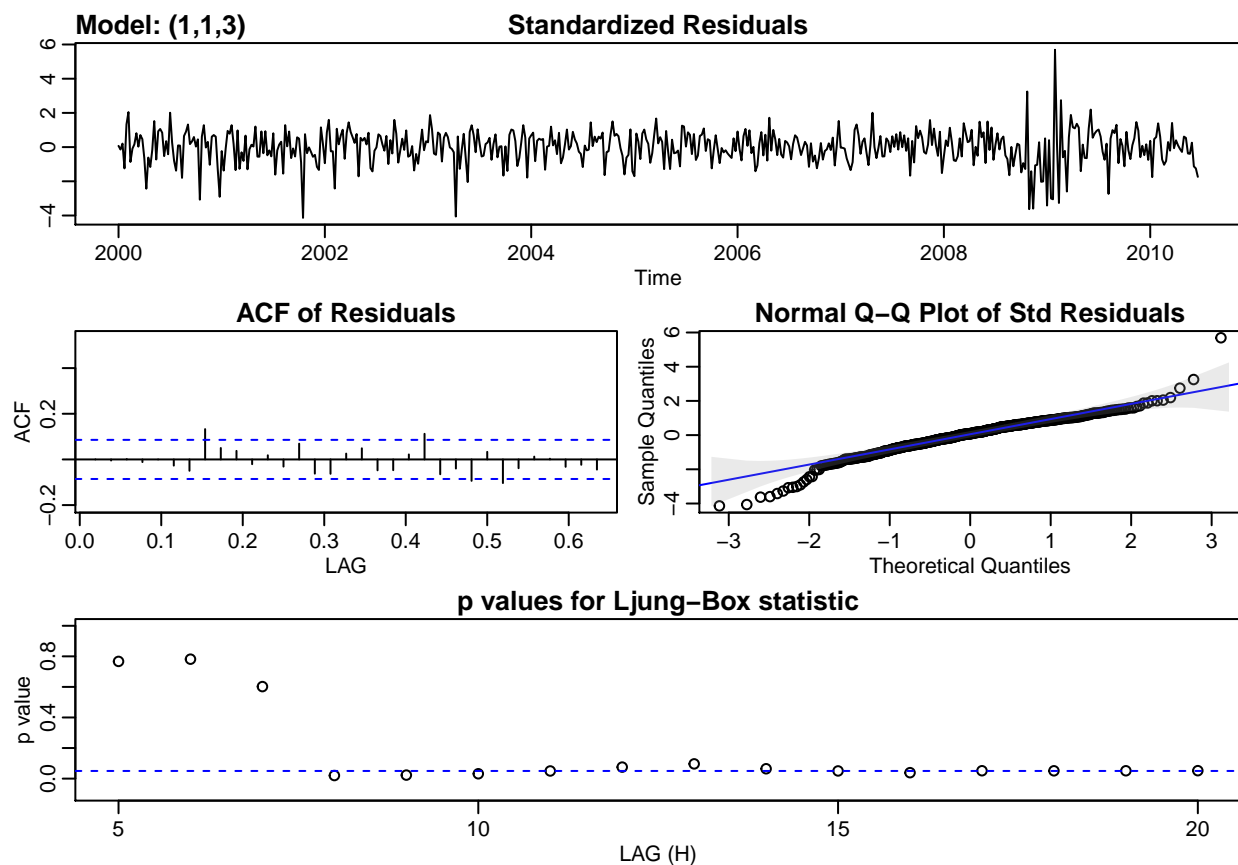
```

```

## initial value -3.057594
## iter    2 value -3.081639
## iter    3 value -3.086469
## iter    4 value -3.086671
## iter    5 value -3.086741
## iter    6 value -3.086743
## iter    7 value -3.086743
## iter    8 value -3.086746
## iter    9 value -3.086748
## iter   10 value -3.086749
## iter   11 value -3.086749

```

```
## iter 12 value -3.086750
## iter 13 value -3.086750
## iter 14 value -3.086750
## iter 15 value -3.086750
## iter 15 value -3.086750
## iter 15 value -3.086750
## final value -3.086750
## converged
## initial value -3.087502
## iter 2 value -3.087503
## iter 3 value -3.087503
## iter 4 value -3.087503
## iter 5 value -3.087503
## iter 6 value -3.087503
## iter 6 value -3.087503
## iter 6 value -3.087503
## final value -3.087503
## converged
```



The AIC and BIC for the models are the following:

```
# Lower AIC/BIC is better
AIC(modelA$fit)
```

```
## [1] -1805.339
```

```
BIC(modelA$fit)
```



```
## [1] -1783.844
```

```
AIC(modelB$fit)
```

```
## [1] -1803.398
```

```
BIC(modelB$fit)
```

```
## [1] -1777.605
```

According to the AIC and BIC score, both models seem to be nearly equally good. Also looking at the Q-Q plots we see that they almost have the exact same behaviour, having a mostly straight line for the quantiles while falling off towards the tails. `modelA` seems slightly better, so we choose this one.

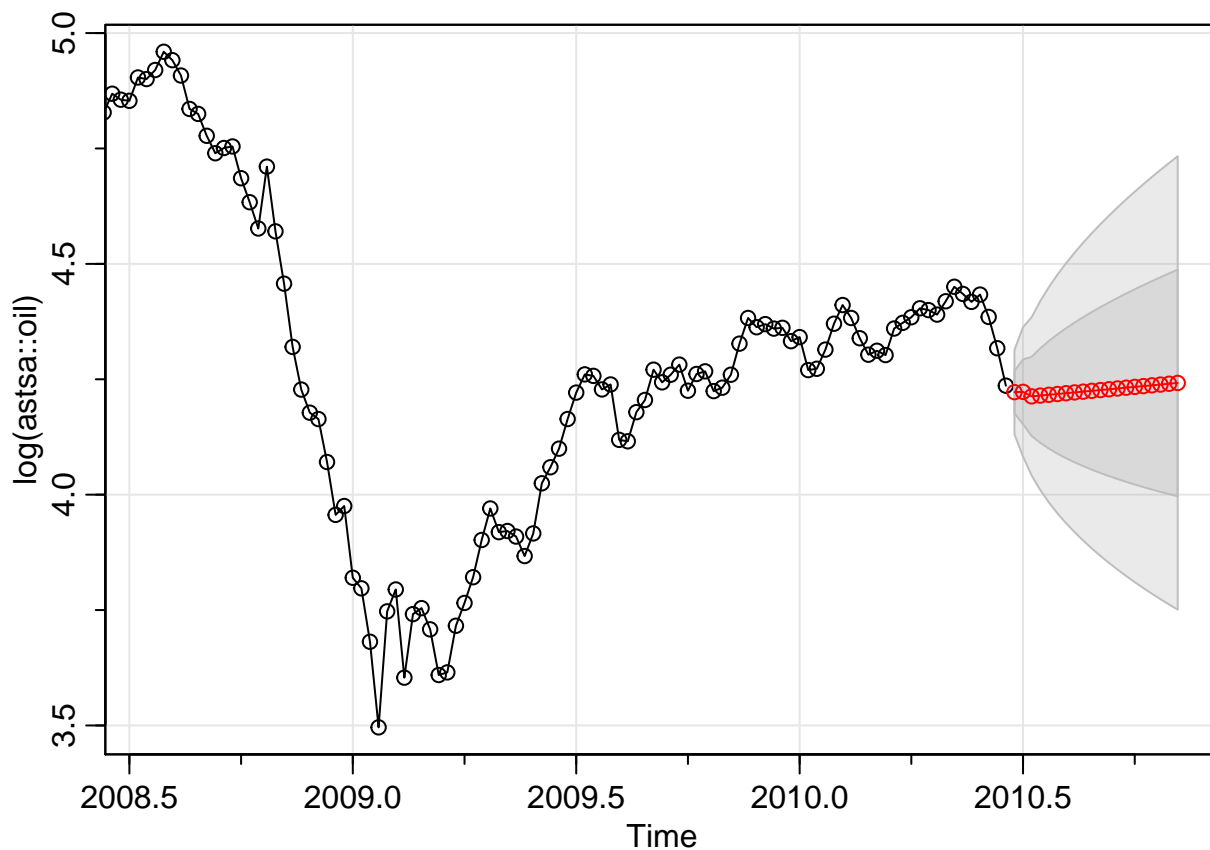
Therefore the model is given by:  $\Delta x_t = w_t + 0.1688w_{t-1} - 0.0900w_{t-2} + 0.1447w_{t-3}$

Finally, let's check for redundancy: As we only have a AR terms, there can be no parameter redundancy.

The forecast looks like this.

```
# Do we have to take the log here?!
```

```
sarima.for(log(astsa::oil), 0, 1, 3, n.ahead = 20)
```



```
## $pred
```

```
## Time Series:
```

```
## Start = c(2010, 26)
```

```
## End = c(2010, 45)
```

```
## Frequency = 52
```

```
## [1] 4.222141 4.222731 4.212938 4.214647 4.216356 4.218066 4.219775
```

```
## [8] 4.221485 4.223194 4.224904 4.226613 4.228323 4.230032 4.231741
```

```
## [15] 4.233451 4.235160 4.236870 4.238579 4.240289 4.241998
```

```
##
```

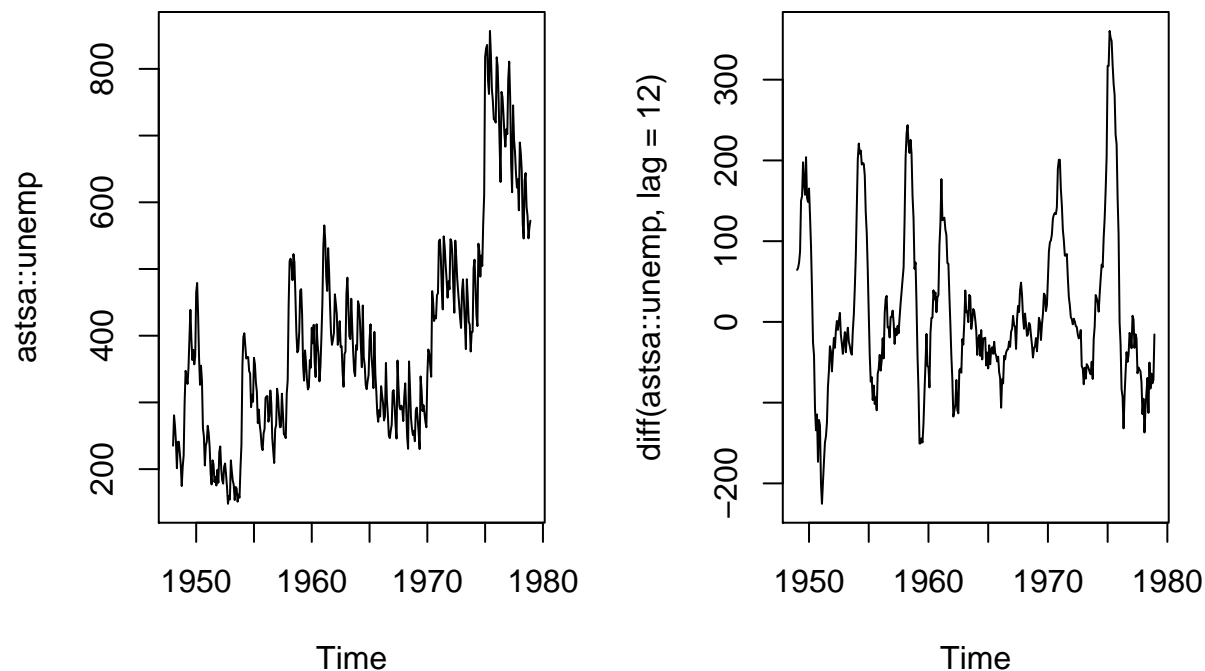
```
## $se
## Time Series:
## Start = c(2010, 26)
## End = c(2010, 45)
## Frequency = 52
## [1] 0.04561249 0.07016150 0.08569792 0.10226755 0.11650396 0.12918085
## [7] 0.14072033 0.15138273 0.16134203 0.17072132 0.17961149 0.18808192
## [13] 0.19618697 0.20397021 0.21146718 0.21870731 0.22571532 0.23251220
## [19] 0.23911597 0.24554218
```

### 3.2 Finding a Suitable ARIMA Model (unemp)

**Task:** Find a suitable  $ARIMA(p, d, q) \times (P, D, Q)_s$  model for the data set `unemp` present in the library `astsa`. Your modeling should include the following steps in an appropriate order: visualization, detrending by differencing (if necessary), transformations (if necessary), ACF and PACF plots when needed, EACF analysis, Q-Q plots, Box-Ljung test, ARIMA fit analysis, control of the parameter redundancy in the fitted model. When performing these steps, always have 2 tentative models at hand and select one of them in the end. Validate your choice by AIC and BIC and write down the equation of the selected model (write in the backshift operator notation without expanding the brackets). Finally, perform forecasting of the model 20 observations ahead and provide a suitable plot showing the forecast and its uncertainty.

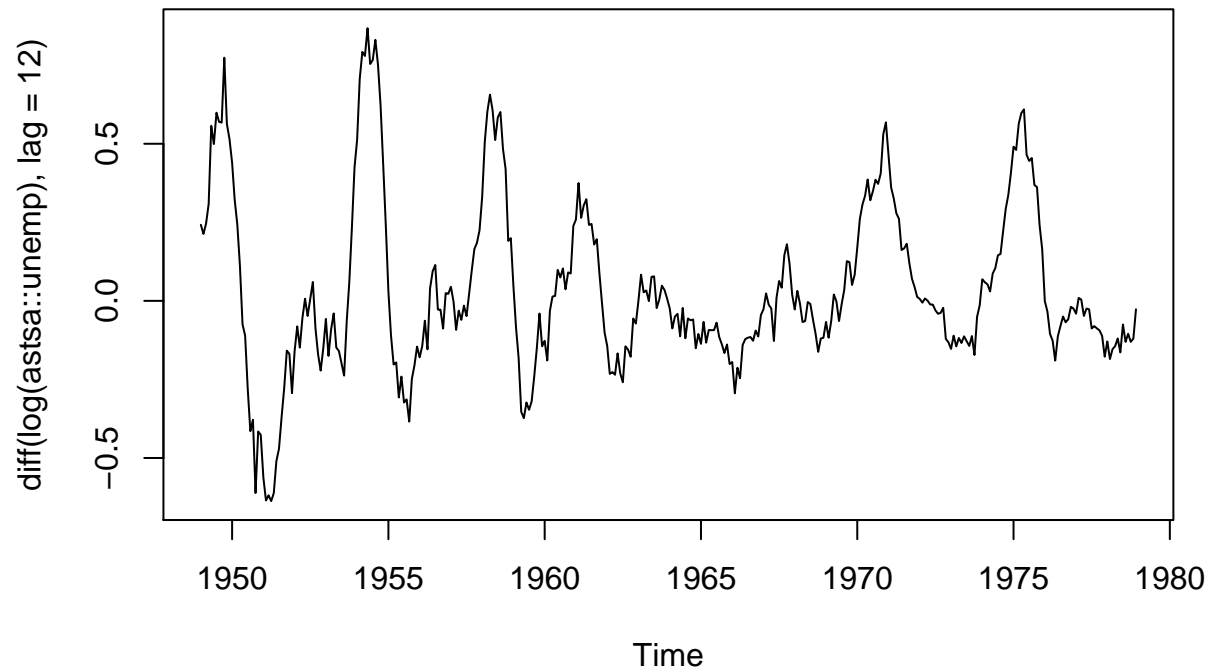
**Answer:** Again we take the data and take a first look at it. We see that it has some interesting climbs and that the series is not stationary. Therefore the next logical step is to take the difference. As we know from the description that we are dealing with monthly data, we will take the first difference with a lag of 12.

```
par(mfrow = c(1, 2))
plot(astsa::unemp)
plot(diff(astsa::unemp, lag=12))
```



That looks better now, but still the variance seems to be a problem, therefore we will also log the data. We see that the result is way better, seems stationary and the variance does not escalate.

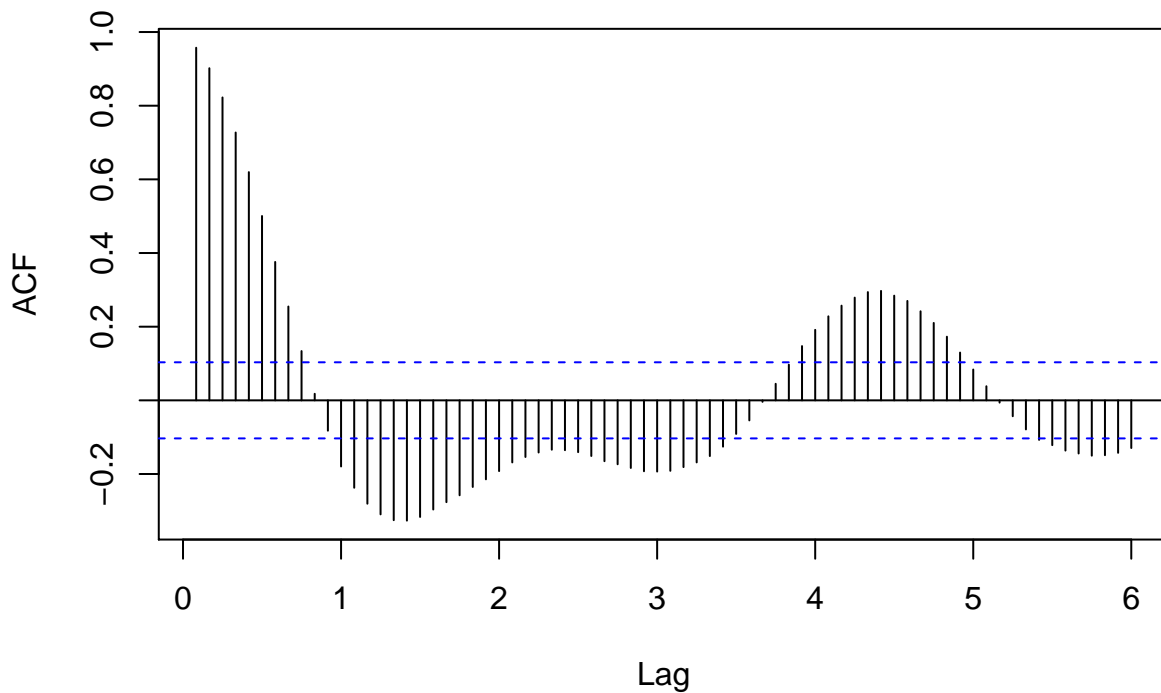
```
plot(diff(log(astsa::unemp), lag=12))
```



As a next step, after applying the differences and transformations, we will look at the ACF and PACF plot. We see, that even after taking the difference with lag 12, we still have seasonality.

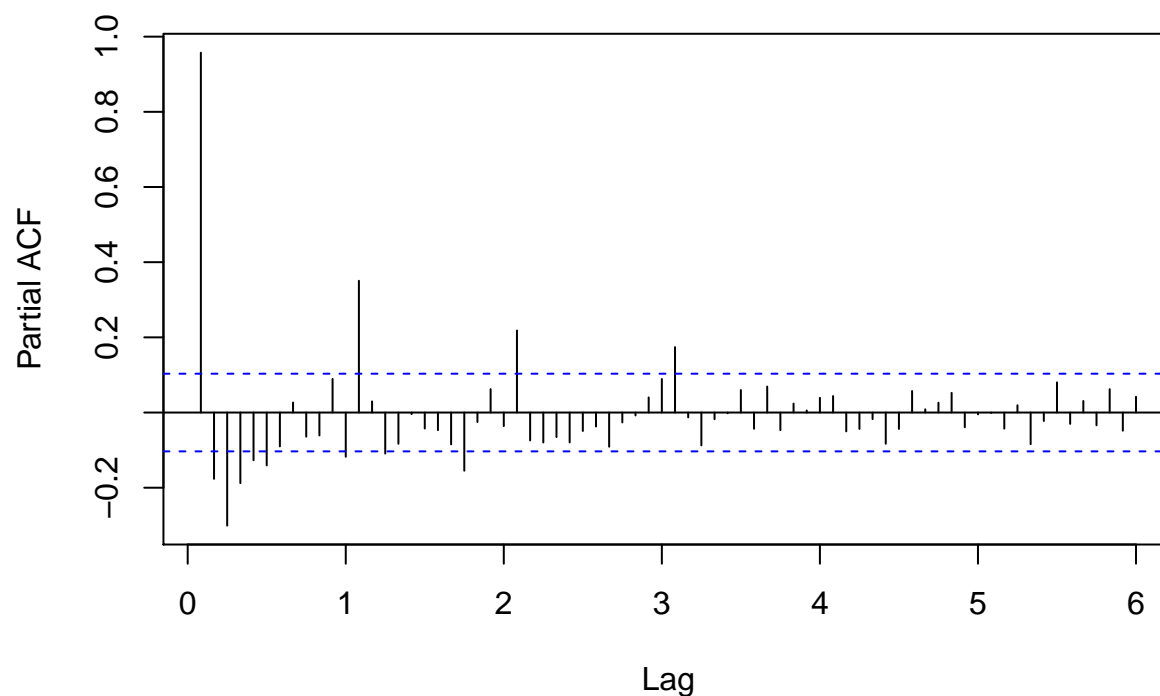
```
acf(diff(log(astsa::unemp), lag=12), lag.max = 12 * 6)
```

### Series $\text{diff}(\log(\text{astsa::unemp}), \text{lag} = 12)$



```
pacf(diff(log(astsa::unemp), lag=12), lag.max = 12 * 6)
```

## Series `diff(log(astsa::unemp), lag = 12)`



It's time to look at the `eacf()` output. We consider the following two models:  $\text{ARIMA}(2, 1, 2) \times (0, 0, 1)_{12}$  and  $\text{ARIMA}(2, 1, 3) \times (0, 0, 1)_{12}$

```
eacf(diff(log(astsa::unemp), lag=12))
```

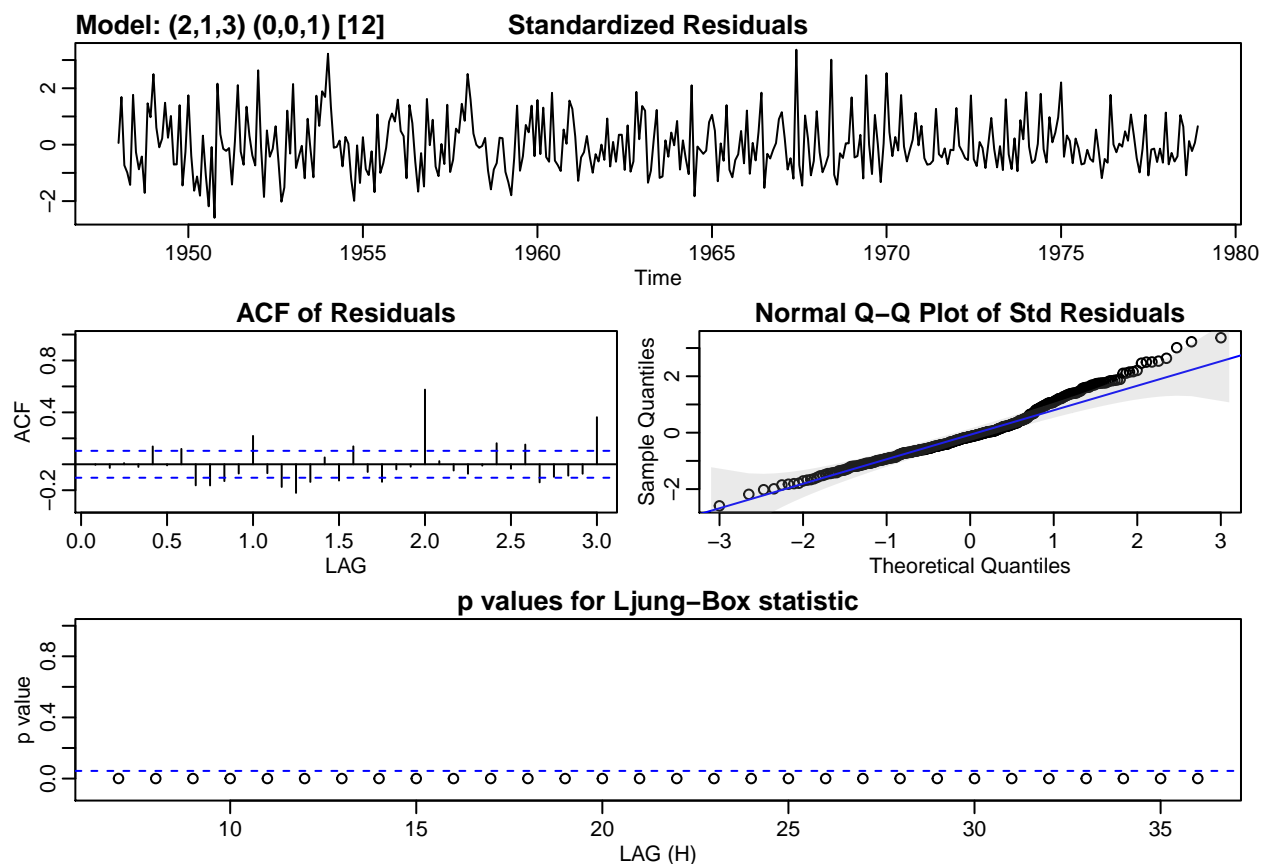
```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x x x x x x x o o x x x
## 1 x x x x x x x x x o o x x x
## 2 x x o o o o o o o x x x x o
## 3 x o o o o o x o o o x x x x
## 4 x o x o o o o o o o o x x x
## 5 x x x o o o o o o o o x x o
## 6 x x x x x o o o o o o x x o
## 7 x x x x x o o o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

Let's fit the models.

```
modelA = sarima(log(astsa::unemp), p=2, d=1, q=3, P=0, D=0, Q=1, S=12)
```

```
## initial value -2.175651
## iter 2 value -2.202968
## iter 3 value -2.357857
## iter 4 value -2.377921
## iter 5 value -2.385991
## iter 6 value -2.388091
## iter 7 value -2.389265
## iter 8 value -2.394275
## iter 9 value -2.396407
## iter 10 value -2.398407
## iter 11 value -2.399769
```

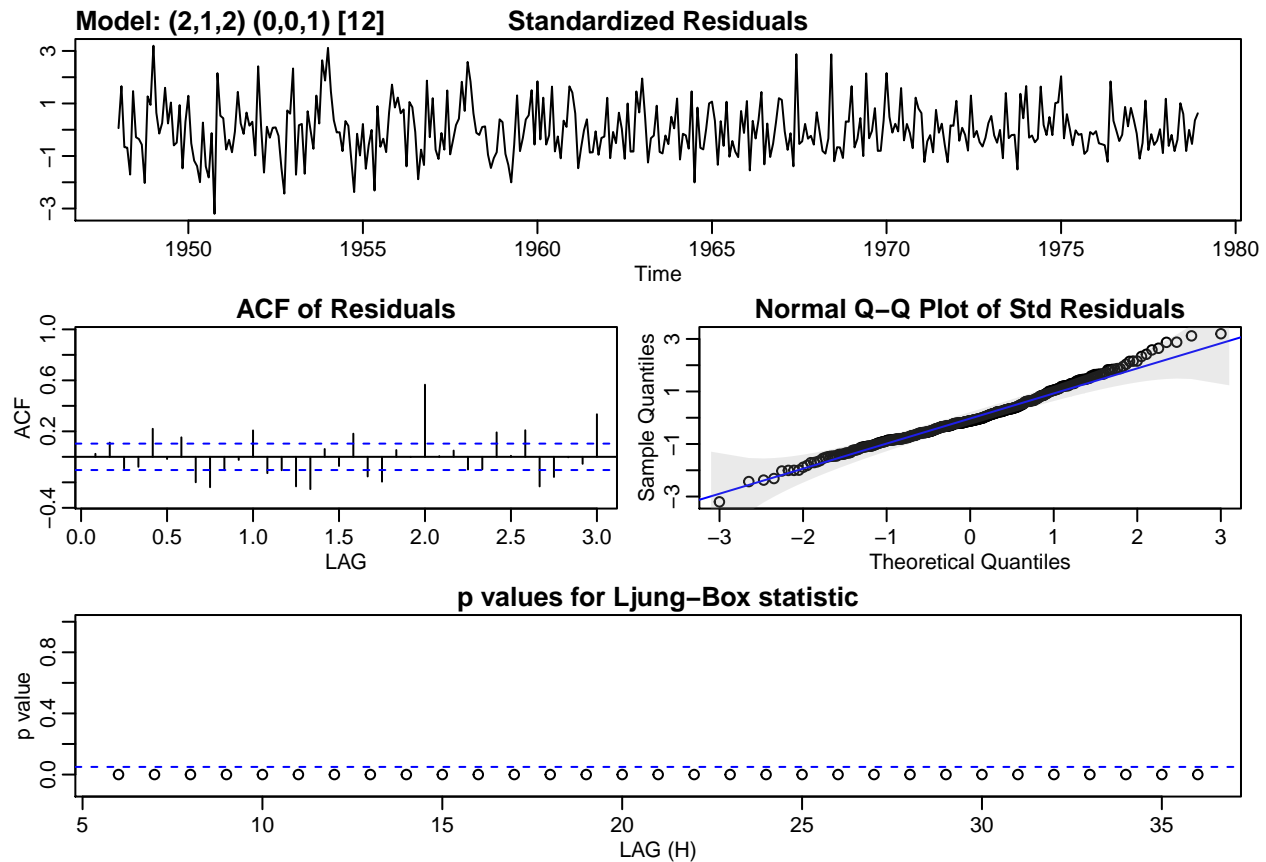
```
## iter 12 value -2.400568
## iter 13 value -2.401910
## iter 14 value -2.403252
## iter 15 value -2.404537
## iter 16 value -2.405381
## iter 17 value -2.405530
## iter 18 value -2.405661
## iter 19 value -2.405688
## iter 20 value -2.405728
## iter 21 value -2.405738
## iter 22 value -2.405749
## iter 23 value -2.405755
## iter 24 value -2.405764
## iter 25 value -2.405772
## iter 26 value -2.405774
## iter 27 value -2.405775
## iter 27 value -2.405775
## iter 27 value -2.405775
## final value -2.405775
## converged
## initial value -2.414812
## iter 2 value -2.414824
## iter 3 value -2.415320
## iter 4 value -2.415394
## iter 5 value -2.415422
## iter 6 value -2.415612
## iter 7 value -2.415890
## iter 8 value -2.416015
## iter 9 value -2.416301
## iter 10 value -2.416550
## iter 11 value -2.417057
## iter 12 value -2.417346
## iter 13 value -2.417634
## iter 14 value -2.417674
## iter 15 value -2.417701
## iter 16 value -2.417725
## iter 17 value -2.417726
## iter 18 value -2.417726
## iter 19 value -2.417726
## iter 20 value -2.417727
## iter 21 value -2.417727
## iter 21 value -2.417727
## iter 21 value -2.417727
## final value -2.417727
## converged
```



```
modelB = sarima(log(astsa::unemp), p=2, d=1, q=2, P=0, D=0, Q=1, S=12)
```

```
## initial value -2.175651
## iter 2 value -2.242799
## iter 3 value -2.350306
## iter 4 value -2.371716
## iter 5 value -2.373275
## iter 6 value -2.375909
## iter 7 value -2.376006
## iter 8 value -2.376226
## iter 9 value -2.378360
## iter 10 value -2.379991
## iter 11 value -2.381860
## iter 12 value -2.387323
## iter 13 value -2.388559
## iter 14 value -2.389176
## iter 15 value -2.389426
## iter 16 value -2.389540
## iter 17 value -2.389899
## iter 18 value -2.390429
## iter 19 value -2.390795
## iter 20 value -2.391079
## iter 21 value -2.391879
## iter 22 value -2.392017
## iter 23 value -2.393341
## iter 24 value -2.394998
## iter 25 value -2.395739
```

```
## iter 26 value -2.397159
## iter 27 value -2.397953
## iter 28 value -2.398395
## iter 29 value -2.399287
## iter 30 value -2.400234
## iter 31 value -2.400344
## iter 32 value -2.400446
## iter 33 value -2.400453
## iter 34 value -2.400459
## iter 35 value -2.400463
## iter 36 value -2.400463
## iter 37 value -2.400463
## iter 37 value -2.400463
## final value -2.400463
## converged
## initial value -2.415916
## iter 2 value -2.417095
## iter 3 value -2.417709
## iter 4 value -2.418360
## iter 5 value -2.418564
## iter 6 value -2.418676
## iter 7 value -2.418717
## iter 8 value -2.419138
## iter 9 value -2.419406
## iter 10 value -2.419602
## iter 11 value -2.419959
## iter 12 value -2.419989
## iter 13 value -2.420203
## iter 14 value -2.420403
## iter 15 value -2.420455
## iter 16 value -2.420481
## iter 17 value -2.420572
## iter 18 value -2.420667
## iter 19 value -2.420672
## iter 20 value -2.420674
## iter 21 value -2.420676
## iter 22 value -2.420677
## iter 23 value -2.420678
## iter 24 value -2.420678
## iter 25 value -2.420679
## iter 26 value -2.420680
## iter 27 value -2.420680
## iter 27 value -2.420680
## final value -2.420680
## converged
```



The AIC and BIC for the models are the following:

```
# Lower AIC/BIC is better
AIC(modelA$fit)
```

```
## [1] -725.101
```

```
BIC(modelA$fit)
```

```
## [1] -693.7714
```

```
AIC(modelB$fit)
```

```
## [1] -729.2925
```

```
BIC(modelB$fit)
```

```
## [1] -701.8791
```

According to the AIC and BIC score, both models seem to be nearly equally good. Also looking at the Q-Q plots we see that `modelB` has a mostly straight line for the quantiles while falling of towards the tails. It looks better compared to `modelA`. It seems that `modelB` is slightly better, so we choose this one.

Therefore the model is given by:  $\Delta x_t - 0.0198x_{t-1} - 0.9829x_{t-2} = w_t + 0.0757w_{t-1} + 1w_{t-2} + 0.4898w_{t-12}$

Finally, lets check for redundancy:

```
ar_terms = c(-0.0198, -0.9829)
ma_terms = c(0.0757, 1, rep(0, 9), 0.4898)

polyroot(ar_terms)
```



```
## [1] -0.02014447+0i
```

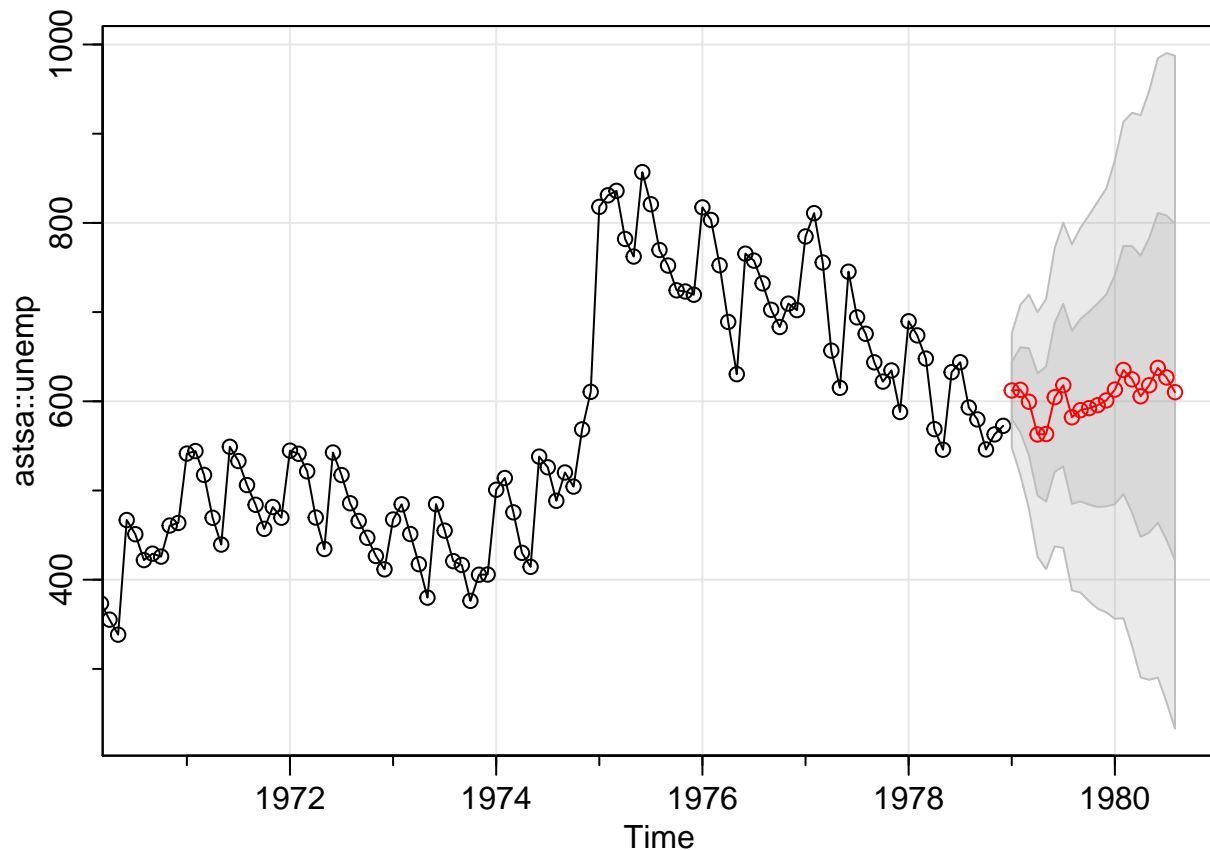
```
polyroot(ma_terms)
```

```
## [1] -0.0757000+0.0000000i -0.6235361+0.8691254i -0.6235361-0.8691254i
## [4] 1.0287239-0.3319613i 0.6386658+0.8690940i -1.0135571+0.3319809i
## [7] 0.0075535-1.0742771i 1.0287239+0.3319613i 0.0075535+1.0742771i
## [10] -1.0135571-0.3319809i 0.6386658-0.8690940i
```

So it looks like we don't have any redundancy, as we only have one term for the AR part which seems far away enough to not influence the other terms.

```
# Do we have to take the log here?!
```

```
sarima.for(astsa::unemp, p=2, d=1, q=2, P=0, D=0, Q=1, S=12, n.ahead = 20)
```



```
## $pred
##      Jan      Feb      Mar      Apr      May      Jun      Jul
## 1979 612.1120 612.8730 599.5147 562.8109 563.2820 604.5993 618.0675
## 1980 613.1313 635.1422 624.5804 605.6000 617.9956 637.4790 626.7701
##      Aug      Sep      Oct      Nov      Dec
## 1979 582.0328 590.0581 592.3169 595.6517 600.9701
## 1980 610.2090
##
## $se
##      Jan      Feb      Mar      Apr      May      Jun      Jul
## 1979 31.90156 47.63057 60.02243 68.58094 75.75319 83.61794 91.14232
## 1980 128.46495 139.15141 149.42503 157.59579 165.12427 173.56991 181.81041
##      Aug      Sep      Oct      Nov      Dec
## 1979 96.98407 102.26105 108.22151 114.05916 118.77682
```

```
## 1980 188.58443
```

## 4 Source Code

```
library(dplyr)
library(ggplot2)
library(kernlab)
library(astsa)
library(TSA)
knitr::opts_chunk$set(echo = TRUE)
set.seed(12345)

model = list(ar = c(0.8, -0.2, 0.1), ma = c())
set.seed(12345)
series = arima.sim(model = model, n = 1000)
pacf(series)
print(pacf(series))

pacf_ar = function(series., lag.max = 30) {

  covariances = vector(length=lag.max)
  series = as.vector(series)

  for (lag in 1:lag.max) {

    # Create a dataframe with the lagged variables
    df = data.frame(y = series)
    df_colnames = c("y")

    if (lag == 1) {
      df = na.omit(cbind(df, lag(series, lag)))
      covariances[1] = cor(df[,1], df[,2])
      next
    }

    for (t in 1:(lag-1)) {
      df_colnames = c(df_colnames, paste("t_", t, sep=""))
      df = cbind(df, lag(series, t))
    }

    # Start at the right index (also omits NAs)
    df = df[(1+lag):nrow(df),]
    colnames(df) = df_colnames

    # Second df
    df2 = data.frame(y = series)
    df2_colnames = c("y")

    for (t in 1:(lag-1)) {
      df2_colnames = c(df2_colnames, paste("t+", t, sep=""))
      df2 = cbind(df2, lead(series, t))
    }
  }
}
```

```

}

# Start at the right index (also omits NAs)
df2 = df2[1:(nrow(df2)-lag),]
colnames(df2) = df2_colnames

# Performing LinReg
# We can take the residuals with intercept, as it does not affect correlation
x_t_dash = lm(y ~ ., df)$residual
x_t_dash_dash = lm(y ~ ., df2)$residual

covariances[lag] = cor(x_t_dash, x_t_dash_dash)
}
return(covariances)
}

# Calculated
pacf_ar(series, 3)

# Function pacf
print(pacf(series, lag.max = 3))

# Theoretical
ARMAacf(model$ar, lag.max = 3, pacf = TRUE)

model = list(ar = c(0.8, 0.1), ma = c())
set.seed(12345)
series = arima.sim(model = model, n = 100)

MOM_Model = ar(series, order = 2, method = "yule-walker", aic = FALSE)
CLS_Model = ar(series, order = 2, method = "ols", aic = FALSE)
ML_Model = ar(series, order = 2, method = "mle", aic = FALSE)

df = data.frame(MOM_Model$ar, CLS_Model$ar, ML_Model$ar)

df

ML_Model_CI = arima(series, order = c(2,0,0), method = "ML")

sigma = ML_Model_CI$var.coef[2, 2]
phi_2 = ML_Model_CI$coef[2]
CI = c(phi_2 - 1.96 * sigma, phi_2 + 1.96 * sigma)

CI

is_within_ci = function() {
  if (df$ML_Model.ar[2] > CI[1] && df$ML_Model.ar[2] < CI[2]) {
    return("The phi_2 estimate lies within the confidence interval.")
  }
}

```

```

    return("The phi_2 estimate does not lie within the confidence interval.")
}

theta = 0.3
Theta = 0.6

model = list(ma = c(theta, rep(0, 10), Theta, theta*Theta))
set.seed(12345)
series = arima.sim(model, n=200)

par(mfrow = c(1,2))
acf(series)
pacf(series)
theoretical_acf = ARMAacf(model$ma, lag.max = 25, pacf = FALSE)
theoretical_pacf = ARMAacf(model$ma, lag.max = 25, pacf = TRUE)

par(mfrow = c(1,2))
plot(theoretical_acf, type = "h", main = "Theoretical ACF", ylab = "ACF")
abline(h = 0, col = "red")
plot(theoretical_pacf, type = "h", main = "Theoretical PACF", ylab = "PACF")
abline(h = 0, col = "red")

fitted_model = arima(series,
                      order = c(0, 0, 1),
                      seasonal = list(order= c(0, 0, 1), period = 12))

prediction = predict(fitted_model, n.ahead = 30)

fitted_model_gausspr = gausspr(c(1:200), series)
prediction_gausspr = predict(fitted_model_gausspr, c(1:230))

df = data.frame(time = c(1:length(series)),
                 data = series)

df2 = data.frame(time = c((length(series)+1):
                           (length(prediction$pred)+length(series))),
                 forecast = prediction$pred,
                 upper_boundary = prediction$pred + 1.96*prediction$se,
                 lower_boundary = prediction$pred - 1.96*prediction$se)

df3 = data.frame(time = c(1:230),
                 gaussian = prediction_gausspr)

ggplot() +
  geom_ribbon(aes(x = df2$time,
                 ymin=df2$lower_boundary,
                 ymax=df2$upper_boundary,
                 fill = "#0000ff", alpha = 0.15)) +
  geom_line(aes(x = df$time, y = df$data, colour = "Original Series")) +

```

```

geom_line(aes(x = df2$time, y = df2$forecast, colour = "Forecasted Series")) +
geom_line(aes(x = df3$time, y = df3$gaussian, colour = "Forecasted Gaussian Series")) +
labs(title = "Random Time Series", y = "Values", x = "Time", color = "Legend") +
scale_color_manual(values = c("#FF5733", "#900C3F", "#444444")) +
theme_minimal()

model = list(ma = c(0.7), ar = c(0.5))
set.seed(12345)
series = arima.sim(model, n=50)
fitted_model = arima(series[1:40], order = c(1, 0, 1), include.mean = FALSE)
prediction = predict(fitted_model, n.ahead = 10)

df = data.frame(time = c(1:length(series)),
                data = series)

df2 = data.frame(time = c(41:50),
                forecast = prediction$pred,
                upper_boundary = prediction$pred + 1.96*prediction$se,
                lower_boundary = prediction$pred - 1.96*prediction$se)

ggplot() +
  geom_ribbon(aes(x = df2$time,
                ymin=df2$lower_boundary,
                ymax=df2$upper_boundary),
            fill = "#0000ff", alpha = 0.15) +
  geom_line(aes(x = df$time, y = df$data, colour = "Original Series")) +
  geom_line(aes(x = df2$time, y = df2$forecast, colour = "Forecasted Series")) +
  labs(title = "LRandom Time Series", y = "Values", x = "Time", color = "Legend") +
  scale_color_manual(values = c("#FF5733", "#900C3F")) +
  theme_minimal()

plot_diagnostics = function(series, max.lag = 40) {
  par(mfrow = c(3, 2))
  plot(series)
  plot(diff(series))
  acf(series, lag.max = max.lag)
  acf(diff(series), lag.max = max.lag)
  pacf(series, lag.max = max.lag)
  pacf(diff(series), lag.max = max.lag)
}

plot_diagnostics(astsa::chicken)

plot_diagnostics(astsa::so2)

plot_diagnostics(astsa::EQcount)

```

```

plot_diagnostics(astsa::HCT)

plot(astsa::oil)

par(mfrow = c(1, 2))
plot(log(astsa::oil))
plot(diff(log(astsa::oil)))

par(mfrow = c(1, 2))
acf(diff(log(astsa::oil)))
pacf(diff(log(astsa::oil)))

eacf(diff(log(astsa::oil)))

# We will use sarima as it will directly create the necessary plots
modelA = sarima(log(astsa::oil), p=0, d=1, q=3)
modelB = sarima(log(astsa::oil), p=1, d=1, q=3)

# Lower AIC/BIC is better
AIC(modelA$fit)
BIC(modelA$fit)

AIC(modelB$fit)
BIC(modelB$fit)

# Do we have to take the log here?!
sarima.for(log(astsa::oil), 0, 1, 3, n.ahead = 20)

par(mfrow = c(1, 2))
plot(astsa::unemp)
plot(diff(astsa::unemp, lag=12))

plot(diff(log(astsa::unemp), lag=12))

acf(diff(log(astsa::unemp), lag=12), lag.max = 12 * 6)
pacf(diff(log(astsa::unemp), lag=12), lag.max = 12 * 6)

eacf(diff(log(astsa::unemp), lag=12))

modelA = sarima(log(astsa::unemp), p=2, d=1, q=3, P=0, D=0, Q=1, S=12)

```

```

modelB = sarima(log(astsa::unemp), p=2, d=1, q=2, P=0, D=0, Q=1, S=12)

# Lower AIC/BIC is better
AIC(modelA$fit)
BIC(modelA$fit)

AIC(modelB$fit)
BIC(modelB$fit)

ar_terms = c(-0.0198, -0.9829)
ma_terms = c(0.0757, 1, rep(0, 9), 0.4898)

polyroot(ar_terms)
polyroot(ma_terms)

# Do we have to take the log here?!
sarima.for(astsa::unemp, p=2, d=1, q=2, P=0, D=0, Q=1, S=12, n.ahead = 20)

```