Econometrics II - Problem 3

William Radaic Peron

August 14, 2020

In this problem, we'll be tackling the issue of *identification* of an ARMA model. Namely, we will employ the *Box-Jenkins* model selection strategy, based upon the concept of *parsimony*.

The principle of *parsimony* is inspired on the trade-off between *fit*, i.e., R^2, and *degrees of freedom*. "Box and Jenkins argue that parsimonious models produce better forecasts than overparametrized models". (p. 76)

The Box-Jenkins strategy is divided in three main stages:

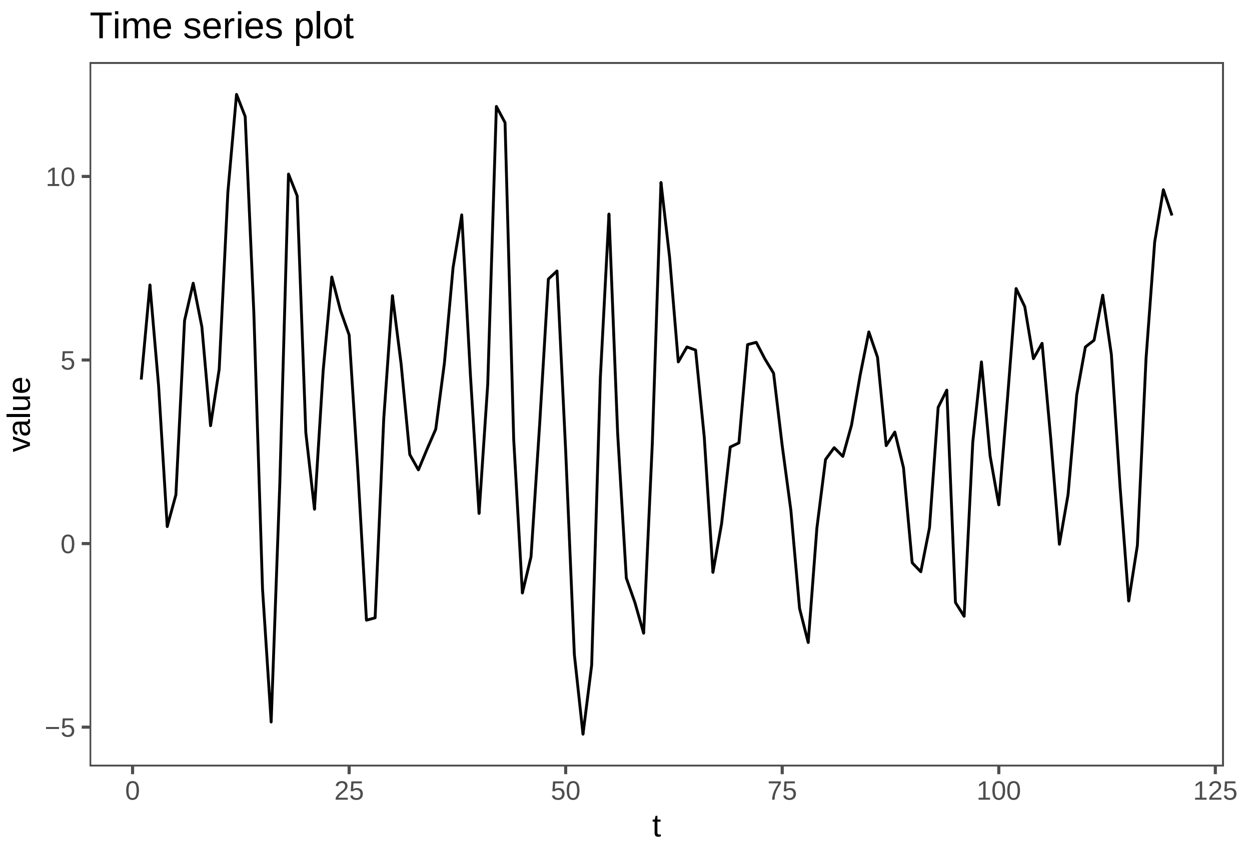
- Identification;
- Estimation;
- Diagnostic checking.

These estimations depend upon two essential conditions (discussed in earlier problems and lectures): *stationarity* and *invertibility*. Stationarity, as we have discussed earlier, is necessary to effectively *employ econometric methods* and to infer characteristics of a population through a given sample. Enders also points out that t-statistics and Q-statistics are based upon the assumption that the data are stationary (p. 77). This implies a condition on the *AR* process of an ARMA model (roots of characteristic polynomial outside of unity circle).

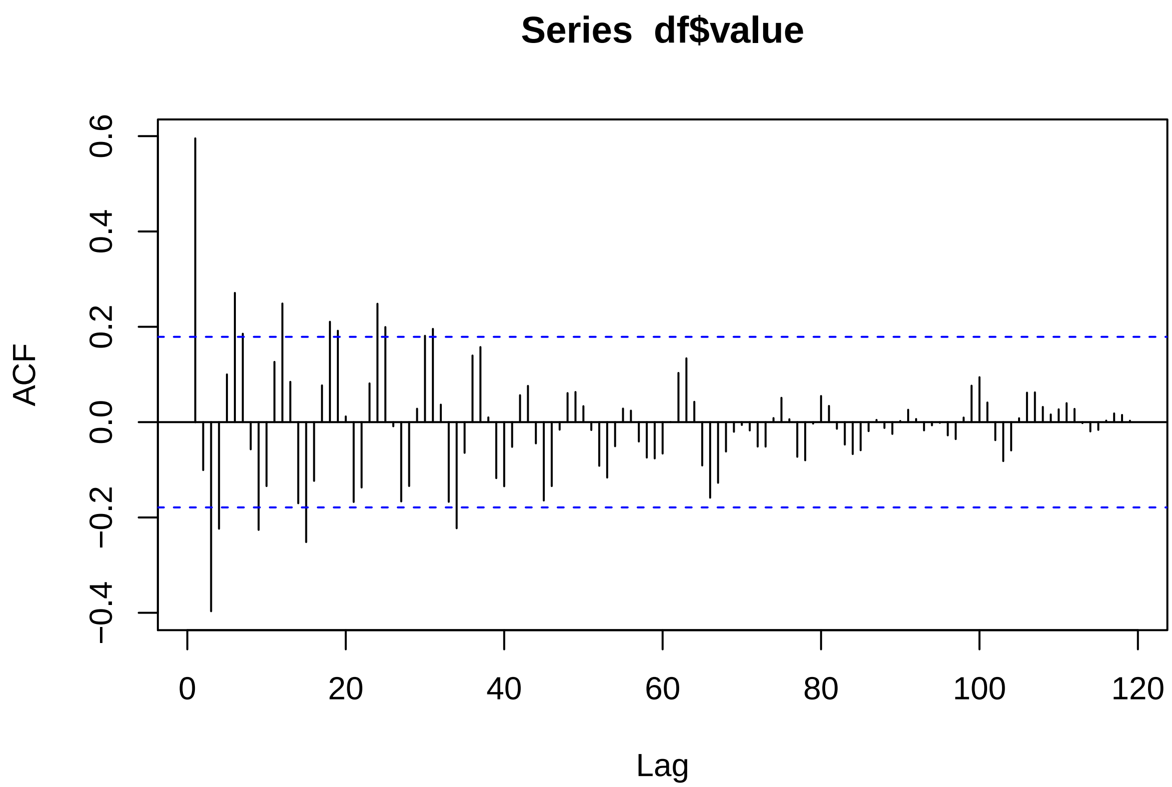
Furthermore, the model shall be *invertible* – i.e., if it can be represented by a finite or convergent AR model. This implies a condition on the *MA* process – i.e., if it can be written as an AR(∞).

We're going to check these conditions intuitively by plotting the ACFs and PACFs of the time series:

```
df <- data.frame(df)

pplot <- ggplot(data = df, aes(x = t, y = value)) + geom_line() + ggtitle("Time series plot") + theme_f
pplot
```




```
acf_ts <- Acf(df$value, lag.max = 5000)
```




```
acf_test_values <- acf_ts$acf/sd(acf_ts$acf)

head(data.frame(acf_test_values))
```

```
##   acf_test_values
## 1       6.5814152
## 2       3.9180772
## 3      -0.6619326
## 4      -2.6109255
## 5      -1.4713722
## 6       0.6589976
```

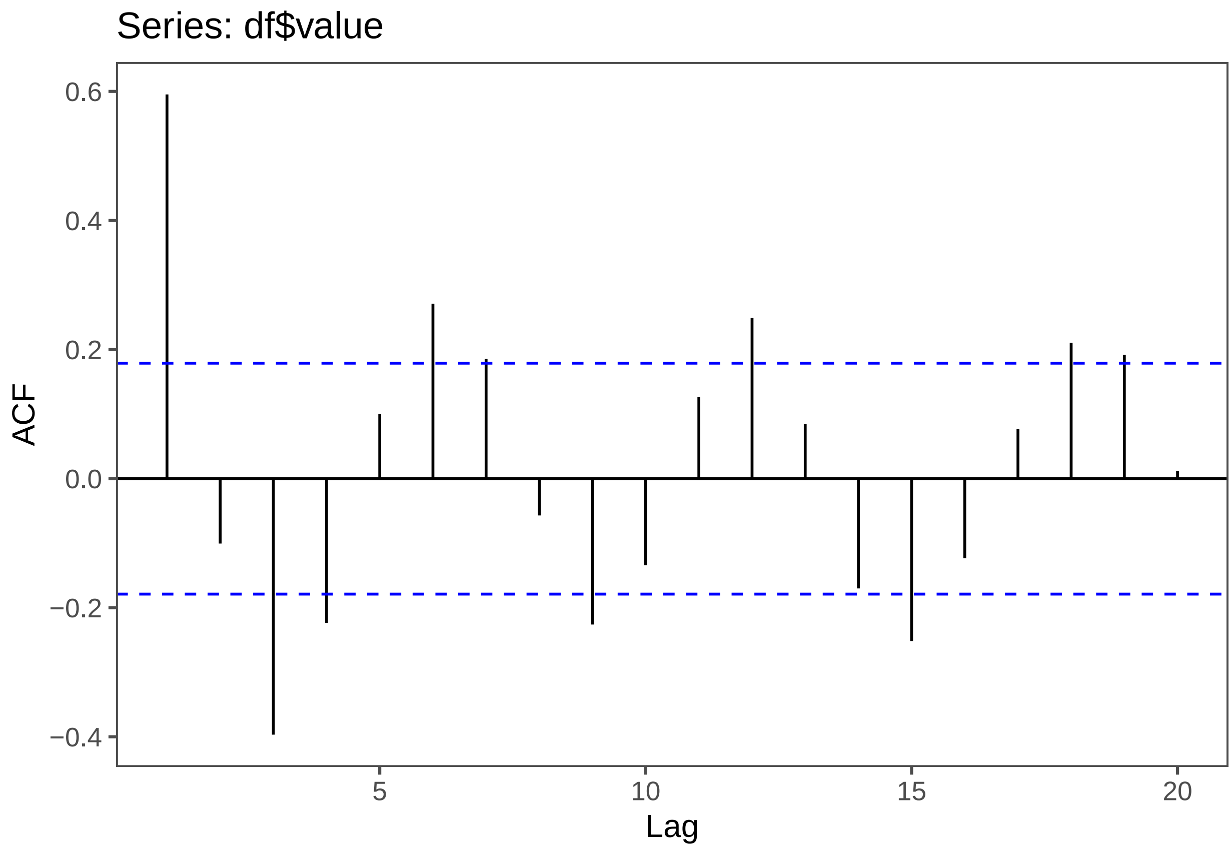
```
facst <- ggAcf(df$value, type = "correlation", lag.max = 20, plot = T) + theme_few()
faclt <- ggAcf(df$value, type = "correlation", lag.max = 5000, plot = T) + theme_few()

facpst <- ggPacf(df$value, type = "correlation", lag.max = 100, plot = T) + theme_few()
```

```
## Warning: Ignoring unknown parameters: type
```

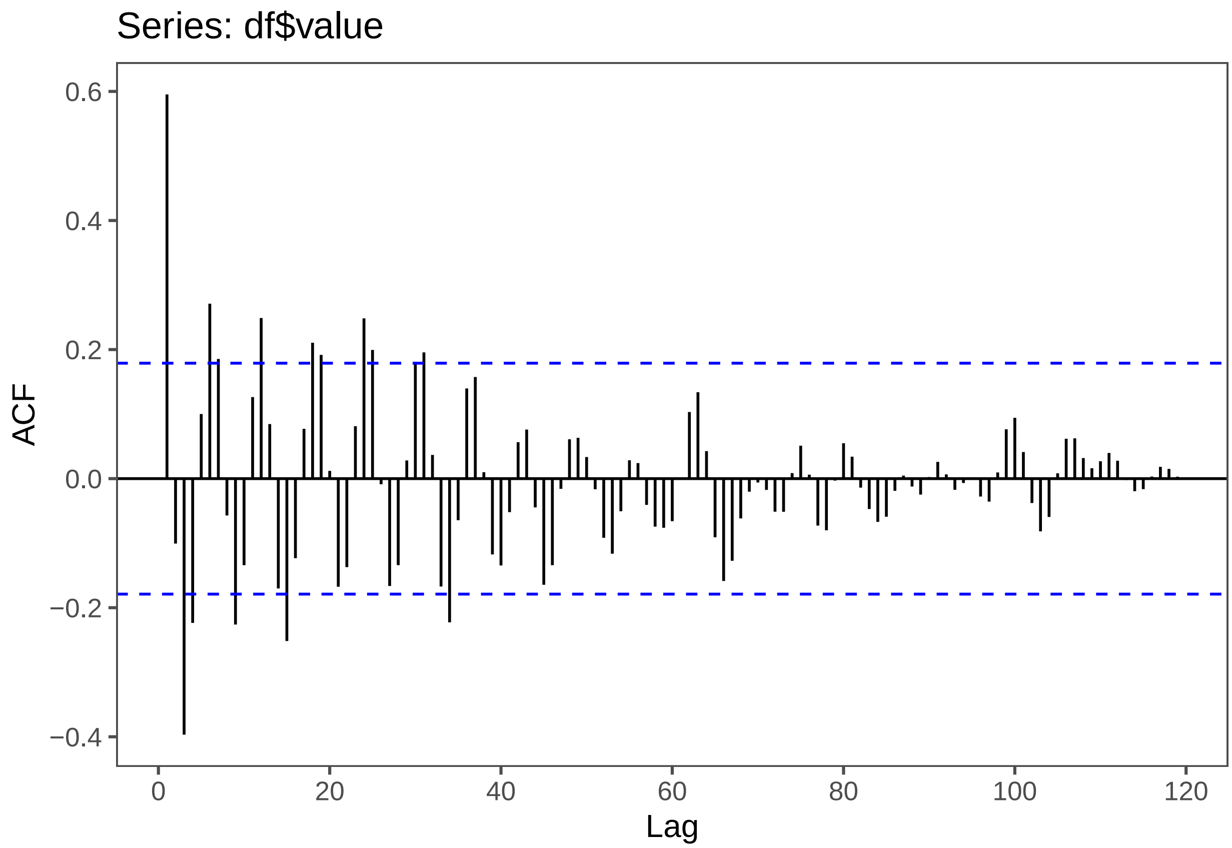
```
facplt <- ggPacf(df$value, type = "correlation", lag.max = 5000, plot = T) + theme_few()
```

```
## Warning: Ignoring unknown parameters: type
```

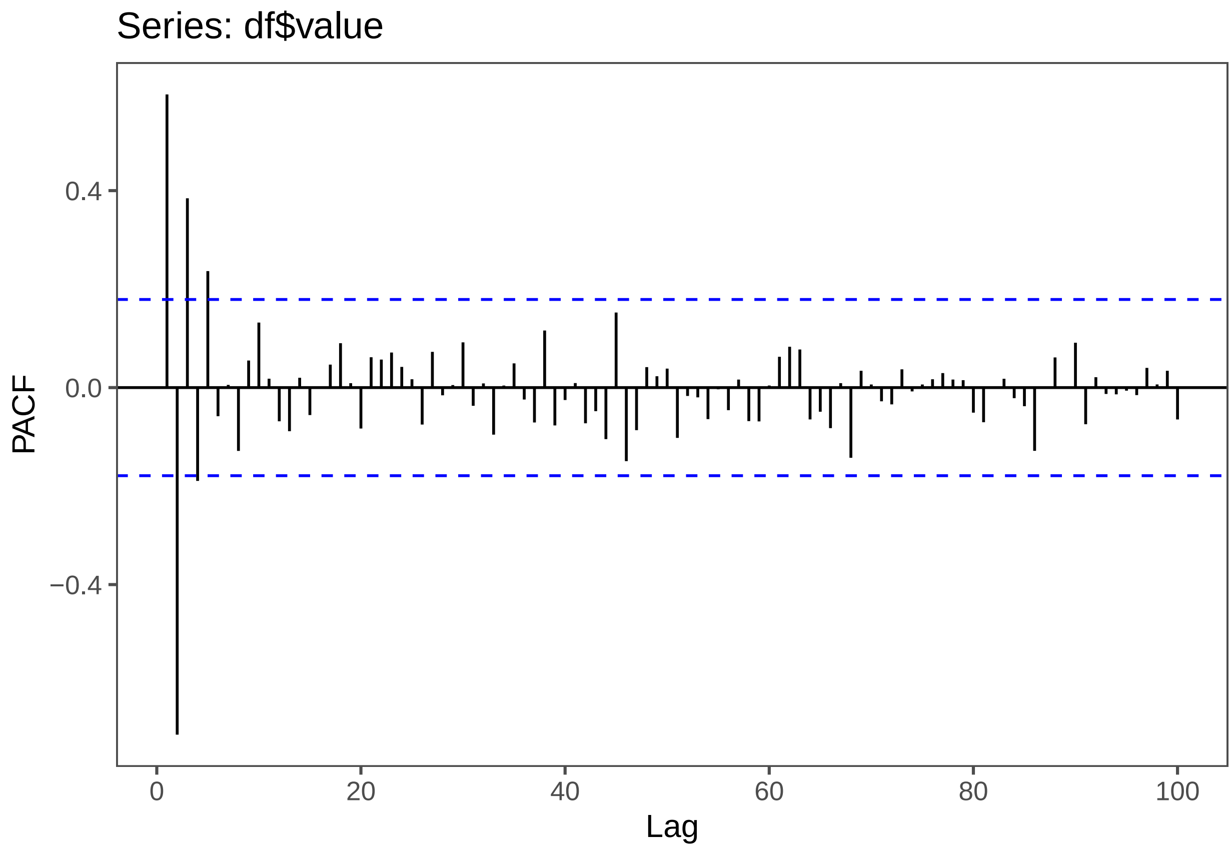
```
facst
```



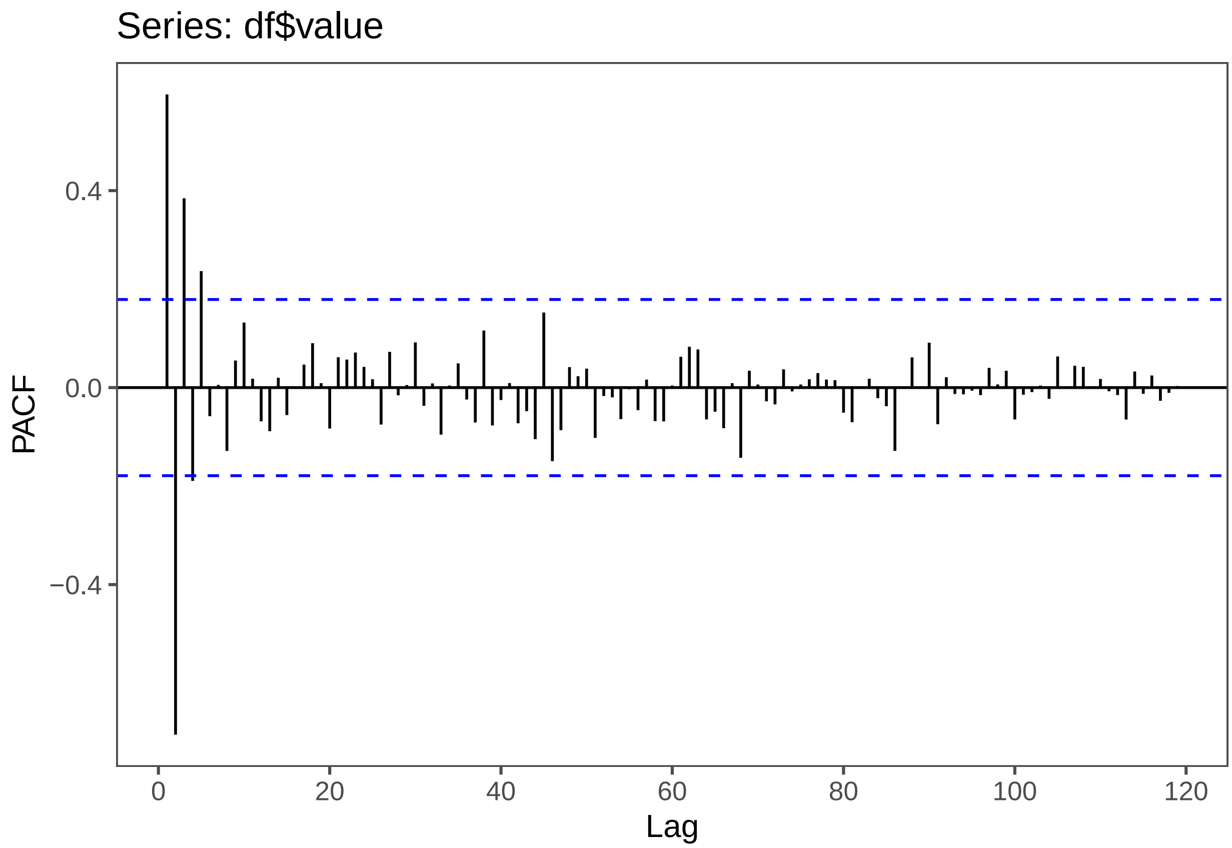

```
faclt
```




```
facpst
```




```
facplt
```



Aside from usual methods, we'll employ the following criteria:

- Akaike Information Criterion (AIC).

$$AIC = T * ln(SSR) + 2n$$

- Schwartz Bayesian Criterion (SBC).

$$SBC = T * ln(SSR) + n * ln(T)$$

n denotes the number of parameters estimated (an useful metric given the importance of the degrees of freedom). T denotes the number of *usable* observations. Note that, when comparing different models, it is important to *fix* T to ensure that the AIC and SBC values are comparable and are capturing only variations in the actual model and not the effect of changing T.

The objective with these criteria is to *minimize* their values. "As the fit of the model improves, the AIC and SBC will approach -∞." (p. 70) AIC and SBC have different advantages and drawbacks: while the former is biased toward overparametrization and more powerful in small samples, SBC is consistent and has superior large sample properties. If both metrics point to the same model, we should be fairly confident that it is, indeed, the correct specification.

It is also important to apply hypothesis tests to the estimates of the population parameters μ, σ^2 and ρ_s – $\bar{y}, \hat{sigma}^2, r_s$, respectively. Worthy of note here is r_s, which presents the following distributions under the null that y_t is stationary with $\varepsilon_t \sim \mathcal{N}$:

$$Var(r_s) = T^{-1} \qquad for\, s = 1$$

$$Var(r_s) = T^{-1}(1 + 2\sum_{j=1}^{s-1} r_j^2) \qquad for\, s > 1$$

The Q-statistic is also introduced by Enders in this chapter. It is used to test whether a group of autocorrelations is significantly different from zero.

$$Q = T\sum_{k=1}^{s} r_k^2$$

Under the null of $r_k = 0 \forall k$, Q is asymptotically χ^2 with s degrees of freedom. "Certainly, a white-noise process (in which all autocorrelations should be zero) would have a Q value of zero". (p. 68)

An alternative form for Q is presented by Ljung and Box (1978):

$$Q = T(T+2)\sum_{k=1}^{s} \frac{r_k^2}{(T-k)}$$

Furthermore, it is also important to check whether the residuals of the model are actually *white noise*. This can be done via the Q-statistic, which *should not result in the rejection of the null*. If that is not the case, the model specified is not the best one available, as there's still a relevant underlying variable (y or ε).

Let's now perform the *estimation stage*. This shall be done via the function *auto.arima* from the package *forecast*.

```
aa_model <- auto.arima(df$value, num.cores = 24, max.d = 0, max.D = 0, stepwise = F)

summary(aa_model)
```

```
## Series: df$value
## ARIMA(2,0,1) with non-zero mean
##
## Coefficients:
##          ar1      ar2    ma1    mean
##       0.7524  -0.5545  0.797  3.5305
## s.e.  0.0818   0.0813  0.064  0.3485
##
## sigma^2 estimated as 3.002:  log likelihood=-235.87
## AIC=481.75   AICc=482.27   BIC=495.68
##
## Training set error measures:
##                      ME     RMSE      MAE      MPE    MAPE      MASE
## Training set 0.01363546 1.703559 1.426984 5.237664 82.3373 0.5612557
##                     ACF1
## Training set 0.004670695
```

```
print("t-values: ")
```

```
## [1] "t-values: "
```

```
aa_t <- matrix(NA, nrow = 4)

for (i in c(1:4)) {

aa_t[i] <- aa_model$coef[i]/sqrt(aa_model$var.coef[i,i])

}

aa_t <- data.frame(aa_t)

aa_t
```

```
##        aa_t
## 1  9.203615
## 2 -6.822352
## 3 12.444488
## 4 10.129782
```

```r
aa_q <- Box.test(aa_model$residuals, lag = aa_model$arma[1] + aa_model$arma[2])
aa_q
```

```
##
##  Box-Pierce test
##
## data:  aa_model$residuals
## X-squared = 0.078351, df = 3, p-value = 0.9943
```

The results of *auto.arima* imply that the best model is an ARMA(2,1):

$$y_t = c + \Phi_1 y_{t-1} + \Phi_2 y_{t-2} + \theta_1 \varepsilon_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim wn(0, \sigma^2)$$

Furthermore, the Q-statistic *(Box.test)* seems to indicate that ε_t is truly white noise.

Let's now run some different models and compare them against the results of *auto.arima*. We'll begin with some overspecified model. First, an ARMA(2,2):

$$y_t = c + \Phi_1 y_{t-1} + \Phi_2 y_{t-2} + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \varepsilon_t, \quad \varepsilon_t \sim wn(0, \sigma^2)$$

```r
arma22 <- Arima(df$value, order = c(2, 0, 2))

summary(arma22)
```

```
## Series: df$value
## ARIMA(2,0,2) with non-zero mean
##
## Coefficients:
##          ar1      ar2     ma1     ma2    mean
##       0.7417  -0.5502  0.8113  0.0149  3.5311
## s.e.  0.1442   0.0950  0.1692  0.1631  0.3514
##
## sigma^2 estimated as 3.028:  log likelihood=-235.87
## AIC=483.74   AICc=484.48   BIC=500.46
##
## Training set error measures:
##                      ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 0.01360342 1.703508 1.426237 4.791703 81.74486 0.5609619
##                     ACF1
## Training set 0.001161589
```

```r
arma22_t <- matrix(NA, nrow = 5)

for (i in c(1:5)) {

arma22_t[i] <- arma22$coef[i]/sqrt(arma22$var.coef[i,i])

}

arma22_t <- data.frame(arma22_t)
```

```
arma22_t
```

```
##      arma22_t
## 1  5.14206433
## 2 -5.78853038
## 3  4.79577309
## 4  0.09134106
## 5 10.04968297
```

```
arma22_q <- Box.test(arma22$residuals, lag = arma22$arma[1] + arma22$arma[2])
arma22_q
```

```
##
##  Box-Pierce test
##
## data:  arma22$residuals
## X-squared = 0.26458, df = 4, p-value = 0.992
```

The t-value of $ma2$ is not able to reject the null hypothesis. Furthermore, the Q-statistic *(Box.test)* seems to indicate that ε_t is truly white noise.

Now, an ARMA(3,1):

$$y_t = c + \Phi_1 y_{t-1} + \Phi_2 y_{t-2} + \Phi_3 y_{t-3} + \theta_1 \varepsilon_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim wn(0, \sigma^2)$$

```
arma31 <- Arima(df$value, order = c(3, 0, 1))

summary(arma31)
```

```
## Series: df$value
## ARIMA(3,0,1) with non-zero mean
##
## Coefficients:
##          ar1     ar2     ar3     ma1    mean
##       0.7610  -0.565  0.0112  0.7923  3.5312
## s.e.  0.1215   0.137  0.1174  0.0825  0.3516
##
## sigma^2 estimated as 3.028:  log likelihood=-235.87
## AIC=483.74   AICc=484.48   BIC=500.46
##
## Training set error measures:
##                      ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 0.01359734 1.703504 1.426202 4.779284 81.71699 0.5609482
##                      ACF1
## Training set 0.0008885573
```

```
arma31_t <- matrix(NA, nrow = 5)

for (i in c(1:5)) {

arma31_t[i] <- arma31$coef[i]/sqrt(arma31$var.coef[i,i])

}

arma31_t <- data.frame(arma31_t)
```

```r
arma31_t
```

```
##      arma31_t
## 1  6.26539086
## 2 -4.12346904
## 3  0.09501299
## 4  9.59788435
## 5 10.04172094
```

```r
arma31_q <- Box.test(arma31$residuals, lag = arma31$arma[1] + arma31$arma[2])
arma31_q
```

```
##
##  Box-Pierce test
##
## data:  arma31$residuals
## X-squared = 0.25911, df = 4, p-value = 0.9923
```

The t-value of $ar3$ is not able to reject the null hypothesis. Furthermore, the Q-statistic *(Box.test)* seems to indicate that ε_t is truly white noise.

Now, let's try some *underspecified models*. Beginning with an ARMA(2,0):

$$y_t = c + \Phi_1 y_{t-1} + \Phi_2 y_{t-2} + \varepsilon_t, \quad \varepsilon_t \sim wn(0, \sigma^2)$$

```r
arma20 <- Arima(df$value, order = c(2, 0, 0))

summary(arma20)
```

```
## Series: df$value
## ARIMA(2,0,0) with non-zero mean
##
## Coefficients:
##          ar1      ar2    mean
##       1.0226  -0.7153  3.5194
## s.e.  0.0634   0.0635  0.2694
##
## sigma^2 estimated as 4.24:  log likelihood=-256.37
## AIC=520.74   AICc=521.09   BIC=531.89
##
## Training set error measures:
##                      ME     RMSE      MAE      MPE     MAPE      MASE      ACF1
## Training set 0.01106086 2.033314 1.630805 73.16585 128.4039 0.6414218 0.2915253
```

```r
arma20_t <- matrix(NA, nrow = 3)

for (i in c(1:3)) {

arma20_t[i] <- arma20$coef[i]/sqrt(arma20$var.coef[i,i])

}

arma20_t <- data.frame(arma20_t)

arma20_t
```

```
##    arma20_t
## 1  16.12226
## 2 -11.26848
## 3  13.06561
```

```r
arma20_q <- Box.test(arma20$residuals, lag = arma20$arma[1] + arma20$arma[2])
arma20_q
```

```
##
##  Box-Pierce test
##
## data:  arma20$residuals
## X-squared = 13.728, df = 2, p-value = 0.001045
```

The Q-statistic indicates that there is an ommited variable – namely, ε_{t-1} that we have just excluded from the model.

Now, an ARMA(1,1):

$$y_t = c + \Phi_1 y_{t-1} + +\theta_1 \varepsilon_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim wn(0, \sigma^2)$$

```r
arma11 <- Arima(df$value, order = c(1, 0, 1))

summary(arma11)
```

```
## Series: df$value
## ARIMA(1,0,1) with non-zero mean
##
## Coefficients:
##          ar1     ma1    mean
##       0.4476  0.9244  3.6027
## s.e.  0.0831  0.0323  0.6234
##
## sigma^2 estimated as 4.026:  log likelihood=-253.74
## AIC=515.48   AICc=515.82   BIC=526.63
##
## Training set error measures:
##                       ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 0.006283661 1.981184 1.621537 51.07775 142.3988 0.6377767
##                   ACF1
## Training set 0.2028683
```

```r
arma11_t <- matrix(NA, nrow = 3)

for (i in c(1:3)) {

arma11_t[i] <- arma11$coef[i]/sqrt(arma11$var.coef[i,i])

}

arma11_t <- data.frame(arma11_t)

arma11_t
```

```
##    arma11_t
## 1  5.387948
## 2 28.624391
## 3  5.779126
```

```r
arma11_q <- Box.test(arma11$residuals, lag = arma11$arma[1] + arma11$arma[2])
arma11_q
```

```
##
##  Box-Pierce test
##
## data:  arma11$residuals
## X-squared = 12.668, df = 2, p-value = 0.001775
```

Again, the Q-statistic indicates that there is an ommited variable – namely, y_{t-1} that we have just excluded from the model.

Finally, let's compare the AIC and BIC values for all these models.

```r
criteria <- matrix(NA, nrow = 5, ncol = 3)

aa_criteria <- data.frame("ARMA(2,1)*", aa_model$aic, aa_model$bic)

names(aa_criteria) <- c("Model", "AIC", "BIC")

arma22_criteria <- data.frame("ARMA(2,2)", arma22$aic, arma22$bic)

names(arma22_criteria) <- c("Model", "AIC", "BIC")

arma31_criteria <- data.frame("ARMA(3,1)", arma31$aic, arma31$bic)

names(arma31_criteria) <- c("Model", "AIC", "BIC")

arma20_criteria <- data.frame("ARMA(2,0)", arma20$aic, arma20$bic)

names(arma20_criteria) <- c("Model", "AIC", "BIC")

arma11_criteria <- data.frame("ARMA(1,1)", arma11$aic, arma11$bic)

names(arma11_criteria) <- c("Model", "AIC", "BIC")


criteria <- rbind.data.frame(aa_criteria, arma22_criteria, arma31_criteria, arma20_criteria, arma11_cri

criteria
```

```
##        Model      AIC      BIC
## 1 ARMA(2,1)* 481.7460 495.6834
## 2  ARMA(2,2) 483.7376 500.4625
## 3  ARMA(3,1) 483.7369 500.4619
## 4  ARMA(2,0) 520.7381 531.8880
## 5  ARMA(1,1) 515.4753 526.6253
```

As we can clearly see, the model chosen by *auto.arima* is the optimal choice according both to AIC and BIC.