Time Series Analysis and Forecasting

Problem set 1: Dynamic Models for Prediction

Emilie Krutnes Engen

Master in Big Data Analytics
Carlos III University of Madrid

Universidad
Carlos III de Madrid

Exercise 1

(a) Forecasting the price of chicken

In this exercise we investigate the development in the annual average price of chicken in the U.S. between 1924 and 1993. The prices are given in dollars, after adjusting for inflation. The prices in the years after 1993 are not given, and we therefore attempt to forecast these prices by the means of some simple forecasting method. Before proceeding with the forecasting, we first study the time series data to decide on a appropriate method. The plot in Figure 1 is obtained by the following R code.

```
plot(chicken, xlab = 'Year', ylab = 'Price', main = '')
```

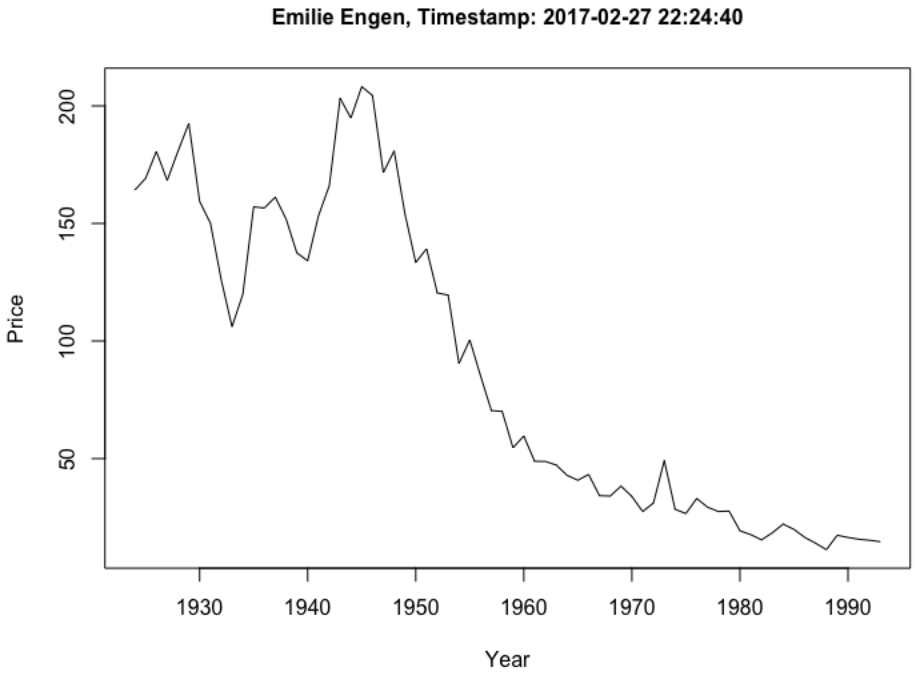

Figure 1: A time series plot of chicken price in the US from 1924 to 1993 in constant US dollars

Here there does not seem to be any evidence of a seasonal or cyclic pattern. However it appears to be a clear negative trend in the data after the price peaked in the mid 1940's. The increasing price around 1940 may be due to a lower supply of chicken during the war. As a result the drift method is considered the most appropriate out of the four benchmark methods. The drift method is an extension of the naive method that allows the forecasts to increase or decrease over time. The amount of change, referred to as the drift, is computed by calculating the average of the change in the historical data. Thus the forecast for $\hat{y}_{T+h}$ is

given by

$$\hat{y}_{T+h} = y_T + \sum_{t=2}^{T}(y_t - y_{t-1}) = y_t + h\frac{y_T - y_1}{T-1} \tag{1}$$

The prediction series, using the drift method is created, using the following R command.

```
chicken.fit <-rwf(chicken, h=5,drift=TRUE)
```

The historical data, followed by a five-year drift forecast is plotted by applying the following R command.

```
plot(chicken.fit, plot.conf=TRUE, xlab = 'Year', ylab = 'Price', main = '')
```

According to the plot in Figure 2 the forecast is given by the blue line, where the shaded area represent the confidence interval. The forecast seem appropriate, though as the price is already relatively low it is likely that the price fall will decrease or is close to reaching bottom. The confidence interval is ranging from -$50 to $50. It is unlikely that price will fall below zero and despite the recent trend we might experience an increasing price in the future, due to the market effect of prices on supply and demand. An alternative method in this case could be the simple naive method. The prices plotted in this exercise illustrates the possible disturbances in prices in the event of a crisis. In this case the war might have affected the supply of chicken and the economic situation in the US. Because the method used to adjust for inflation is unknown there is an uncertainty related to the accuracy of the price development. Another adjustment technique may provide a different plot. One might also consider applying some mathematical transformation before performing the forecasting that captures the slowing price decrease.

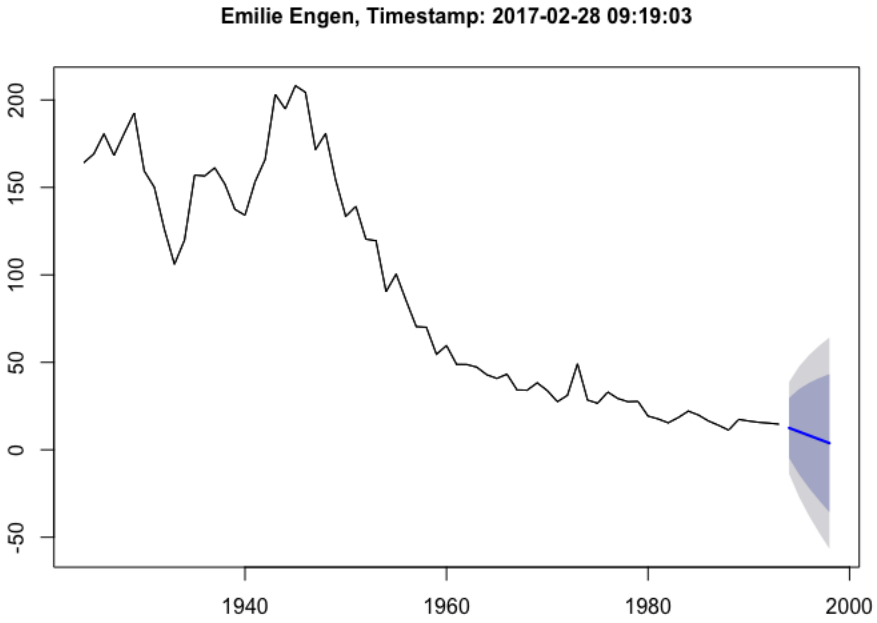

Figure 2: A time series plot of chicken price in the US from 1924 to 1993 followed by a five-year forecast

(b) Forecasting monthly accidental deaths in the United States

In this exercise we investigate the time series data of monthly accidental deaths in the United States from 1973 to 1979. Before deciding on forecasting method, we first study the characteristics of the time series data. The plot in Figure 3 is obtained by the following R code.

```
plot(usdeaths, xlab = 'Year', ylab = 'Number of deaths', main = '')
```

From the plot we see a clear evidence of seasonality, but no clear trend. This underlying seasonal pattern can be investigated further by a seasonal plot. This is done by applying the following R code.

```
seasonplot(usdeaths, xlab = 'Month', ylab = 'Number of deaths', main = '',
    year.labels=TRUE, col=1:7, pch=19)
```

The seasonal plot in Figure 4 supports the assumption of seasonality in the data.

We therefore consider the seasonal naïve method as our benchmark method for forecasting future accidental deaths. In this case the forecast is equal to the last observed value from the same season. Thus the forecast for $\hat{y}_{T+h}$ is given by

$$\hat{y}_{T+h} = y_{T+h-km} \tag{2}$$

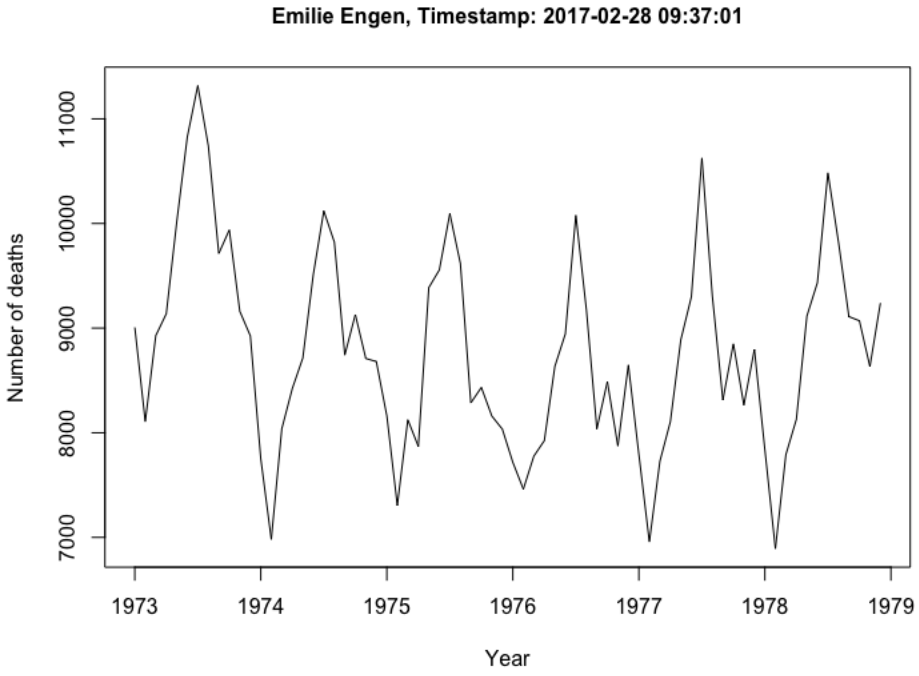

Figure 3: A time series plot of monthly accidental deaths in the US from 1973 to 1979

where m is the seasonal period, in this case the month, and $k = \lfloor (h-1)/m \rfloor + 1$. The prediction series, using the drift method is created, using the following R command.

```
usdeaths.fit <- snaive(usdeaths, h=12)
```

The historical data, followed by a one-year seasonal naïve forecast is plotted by applying the following R command.

```
plot(usdeaths.fit, plot.conf=TRUE, xlab = 'Month', ylab = 'Number of deaths',
    main = '')
```

According to the plot in Figure 5 the forecast is given by the blue line, where the shaded area represent the confidence interval. The forecast seem appropriate, although the next years monthly deaths are not likely to be precisely the same as the previous year. One might consider improving the forecast by performing calender and population adjustments.

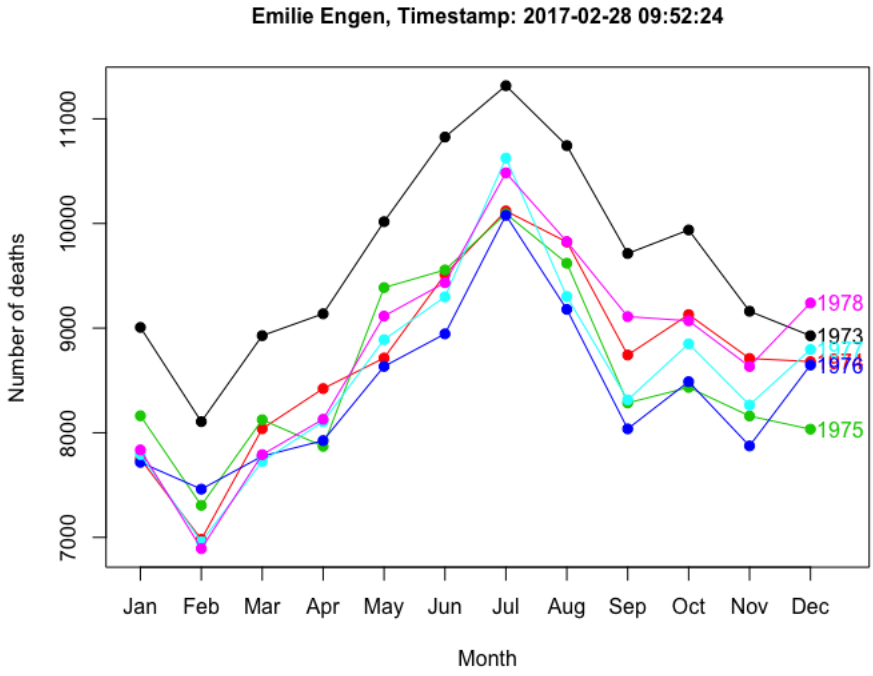

Figure 4: A seasonal plot of monthly accidental deaths in the US from 1973 to 1979

Exercise 2

In this exercise we consider the daily closing IBM stock price for a given period of 369 days.

(a) Plot of the IBM stock prices

Before investigating possible forecasting methods, we first provide a time series plot of the data. This is done by applying the following R code.

```
plot(ibmclose, xlab = 'Day', ylab = 'Closing price $', main = '')
```

From this we obtain the plot presented in Figure 6.

The time series neither show a clear trend nor any seasonal or cyclic pattern.

(b) Partitioning the data into a train and test set

We proceed by partitioning the data into a train and test set. The train set includes the first 300 observations while the test set includes the remaining 69 observations. The train partition is declared by applying the following R code.

```
ibmclose_train <- window(ibmclose,start=1,end=300)
```

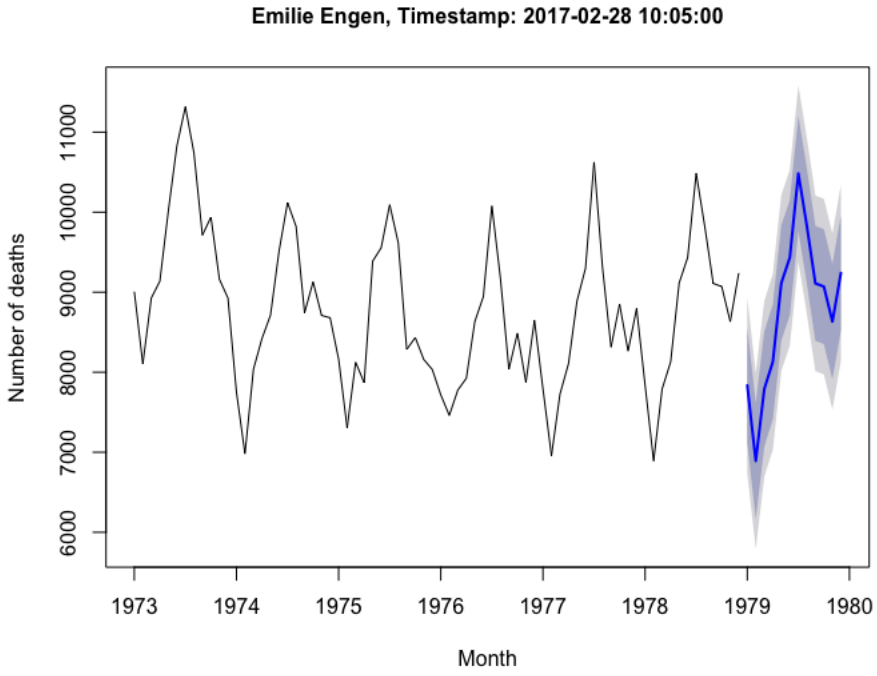

Figure 5: A time series plot of monthly accidental deaths in the US from 1973 to 1979 followed by a one-year forecast

(c) Applying simple forecasting methods

After partitioning the data set we test the different benchmark methods on the test partition: the mean, the naïve, the seasonal naïve and the drift method. The forecasts are obtained by running the following R code.

```
ibmclose_train.fit1 <- meanf(ibmclose_train, h=69)
ibmclose_train.fit2 <- naive(ibmclose_train, h=69)
ibmclose_train.fit3 <- snaive(ibmclose_train, h=69)
ibmclose_train.fit4 <-rwf(ibmclose_train, h=69,drift=TRUE)$
```

The results from the forecasts are presented is Figure 7.

From the plot it is clear that none of the methods succeed in providing an accurate forecast of the test data. However the naïve and drift methods certainly outperform the mean method. Notice that the naïve and the seasonal naïve method give the same forecast, as the data do not include any seasonality. The accuracy of the different benchmark methods can be further analyzed by computing different error measures. The errors are obtained from the following R code.

```
ibmclose_test <- window(ibmclose, start=301)
a1<-accuracy(ibmclose_train.fit1, ibmclose_test);a1
a2<-accuracy(ibmclose_train.fit2, ibmclose_test);a2
a3<-accuracy(ibmclose_train.fit3, ibmclose_test);a3
a4<-accuracy(ibmclose_train.fit4, ibmclose_test);a4
```

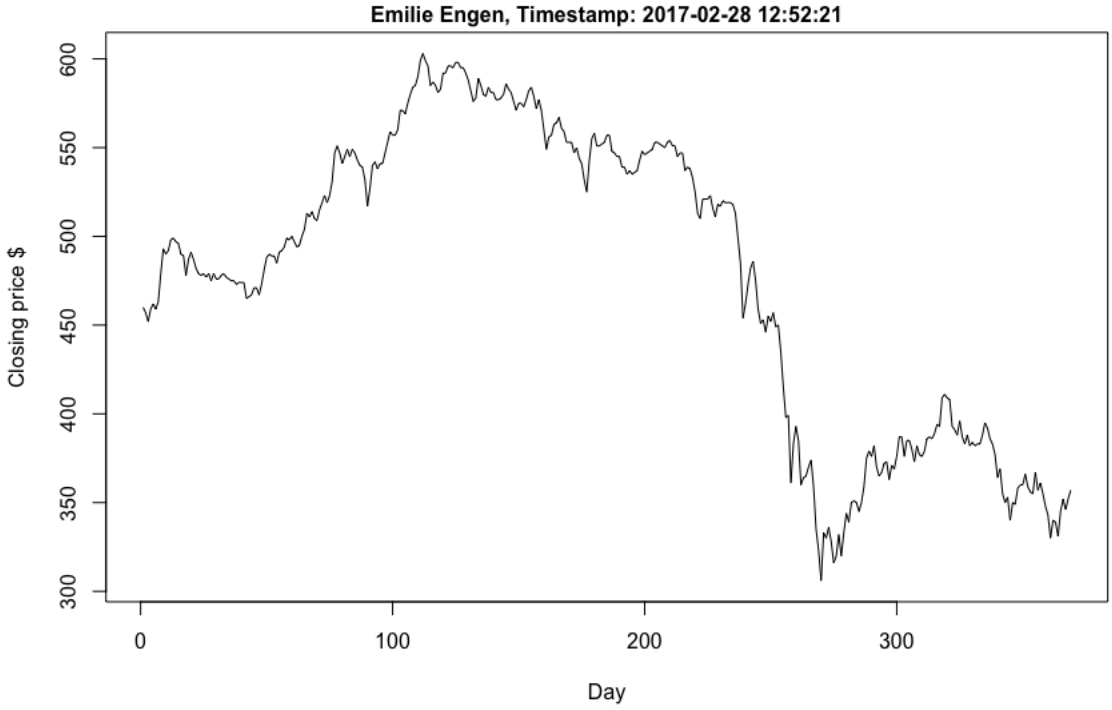

Figure 6: A time series plot of the closing IBM stock price for a period of 369 days

According to the error measures in Table 1 the naïve and drift methods yield very similar results. In general the drift method provide a slightly lower error than the naïve method, with the exception of the mean error (ME) and the mean percentage error (MPE). Notice that for stock market indexes, the best forecasting method is often the naïve method. However, since the drift method yield slightly better results, we will proceed the analysis, using the drift method.

Error Measures								
Method	Mean		Naïve		Seasonal Naïve		Drift	
Error	Train	Test	Train	Test	Train	Test	Train	Test
ME	0.00	-130.62	-0.28	-3.72	-0.28	-3.72	0.00	6.11
RMSE	73.62	132.13	7.30	20.25	7.30	20.25	7.30	17.07
MAE	58.72	130.62	5.10	17.03	5.10	17.03	5.13	13.97
MPE	-2.64	-35.48	-0.08	-1.29	-0.08	-1.29	-0.03	1.42
MAPE	13.03	35.48	1.12	4.67	1.12	4.67	1.12	3.71
MASE	11.52	25.63	1.00	3.34	1.00	3.34	1.01	2.74
ACF1	0.99	0.93	0.14	0.93	0.14	0.93	0.14	0.90
Theil's U	NA	19.06	NA	2.97	NA	2.97	NA	2.36

Table 1

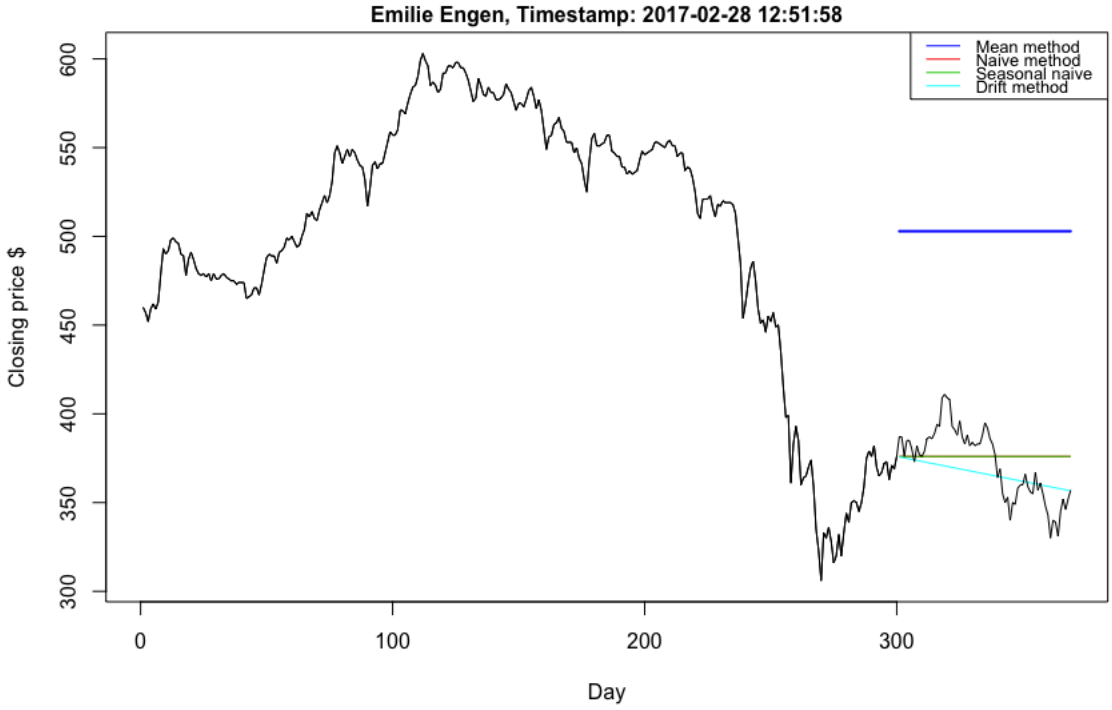

Figure 7: Forecast of the closing IBM stock price on the last 69 days

(d) Computing and plotting the residuals

In order to further evaluate the drift method, we want to take a closer look at the residuals. A residual e_t is defined as the difference between the observed and the predicted value from a given forecast method. In time series forecasting, a residual is based on one-step forecasts, meaning that the forecast $\hat{y}_t$ of y_t is based on the observations $y_1, \ldots, y_{t1}$. For the drift method the residuals are given by the following equation.

$$e_t = y_t - \hat{y}_t = y_{T+h} - \hat{y}_{T+h} \tag{3}$$

The residuals are computed and plotted by running the following R code.

```
res_drift <- residuals(rwf(ibmclose));res_drift
plot(res_drift, main='',ylab='Residuals', xlab='Day')
```

From the residuals plot in Figure 8 we have that the mean of the residuals are close to zero. However the variance in the residuals is not constant with time, which suggest that there are some information hidden in the data. This may lead us to assume that the data is non-stationary.

We proceed by plotting the correlation between the residuals, given by the following R code.

```
Acf(res_drift, main='')
```

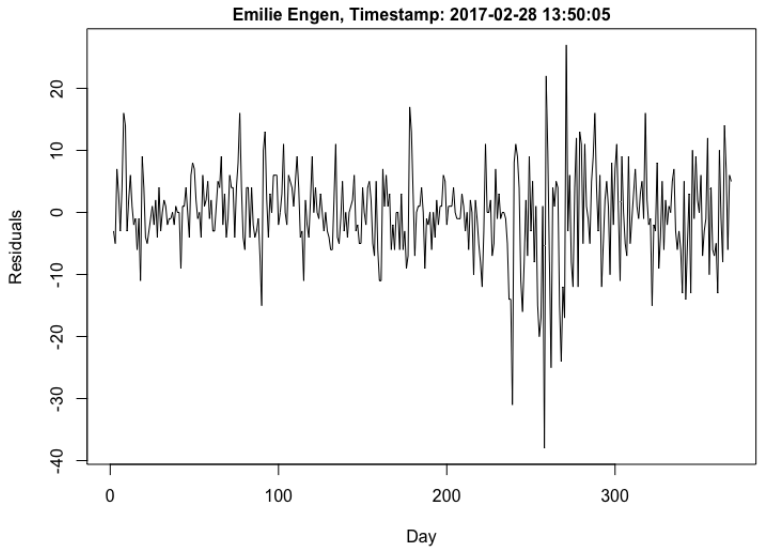

Figure 8: Residuals plot from forecasting the closing IBM stock price with the drift method

From the auto-correlation plot in Figure 9 it appears that there is some correlation in the residuals, indicated by lag 6,16 and 17. This suggest that there are some information in the data that we are not able to capture using the simple drift method.

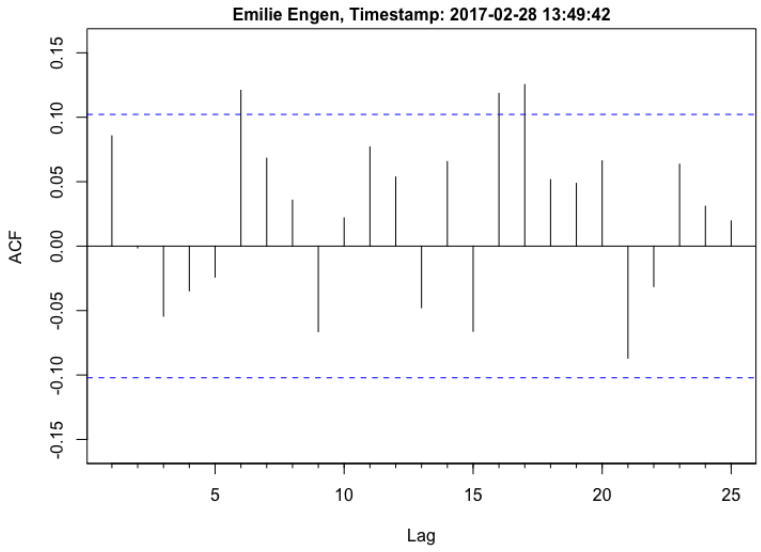

Figure 9: ACF of the residuals from the drift method applied to the closing IBM stock price with the drift method

We plot the distribution of the residuals in a histogram by applying the following R code.

```
hist(res_drift, nclass='FD', main='',xlab='Residuals',col='light blue')
```

The histogram in Figure 10 suggest that the residuals are close to following a normal distribution. Consequently, assuming a normal distribution in the prediction intervals seems like a proper assumption.

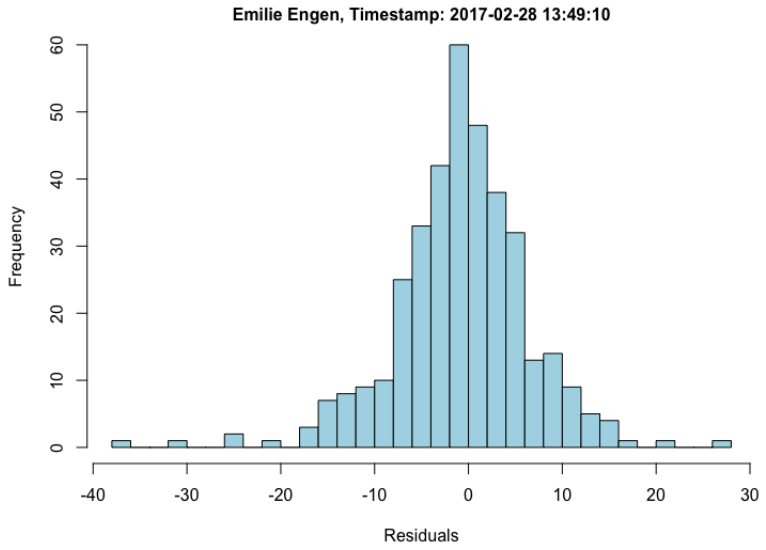

Figure 10: Histogram of the residuals from the drift method applied to the closing IBM stock price with the drift method

Exercise 3

In this exercise we investigate a data set that represent the monthly sales (in thousands) of a product A for a plastics manufacturer for a period of five years.

(a) Plot of the monthly sales

Initially we plot the monthly sales by using the following R code.

```
plot(plastics, xlab = 'Year', ylab = 'Monthly sales', main = '')
```

Figure 11: A time series plot of monthly sales of plastics

The plot in Figure 11 show clear evidence of both seasonality and a positive trend. We investigate this further plotting a seasonal plot. The plot in Figure 12 is obtained by the following R code.

```
seasonplot(plastics, xlab = 'Month', ylab = 'Monthly sales', main = '', year.
    labels=TRUE, year.labels.left=TRUE, col=2:6, pch=19)
```

Figure 12: A seasonal plot of monthly sales of plastics

This plot supports the assumption of a seasonal pattern in the data. An alternative to the seasonal plot is the month plot, given by the following R code.

```
monthplot(plastics, xlab = 'Month', ylab = 'Monthly sales', main = '')
```

The month plot is presented in Figure 13 and also emphasises the seasonal patterns. Here the data for each season are collected together in separate mini time plots. The mean for each month is illustrated by the horizontal line. The seasonal pattern is clearly illustrated in this plot. We further see that the trend slows towards the end of the period and that in the last year the sales decrease to a lower value than the previous year, indicating a trend cycle.

Figure 13: A month plot showing the monthly sales of plastics

(b) Investigate for seasonality and trend

We continue our analysis of the data set, by performing a STL decomposition in order to calculate the trend-cycle and the seasonal indices. By changing the parameter $s.window$ we can experiment with the seasonality component. By setting $s.window =' periodic'$ we say that the seasonal component can not change over time. By changing this parameter to $s.window = 12$, $s.window = 24$ and $s.window = 36$ we see that the differences in the plots are quit small. By setting the seasonal component to periodic we plot the STL decomposition, using the following R code.

```
fit <- stl(plastics, t.window=12, s.window='periodic', robust=TRUE)
plot(fit)
```

The plot of the STL decomposition is presented in Figure 14

Figure 14: The three components obtained from a robust STL decomposition with fixed seasonality

(c) Comparing computational result and graphical interpretation

From the STL decomposition we see a clear seasonality and a trend slowing towards the end of the period. Thus it is clear that the decomposition of the data supports the assumptions made from the previous graphical interpretations.

(d) Adjusting for seasonality

We proceed by computing and plotting the seasonal adjusted data, by applying the following R code.

```
plastics_seasadj <- seasadj(fit)
plot(plastics, xlab = 'Year', ylab = 'Monthly sales', main = '')
lines(plastics_seasadj,col='blue')
```

The plot is presented in Figure 15. Here the you can see that the seasonal component is removed from the original data, leaving only the seasonal adjusted data.

Figure 15: A plot of the seasonal adjusted data

(e) Forecasting with seasonally adjusted data

We then forecast the seasonally adjusted data with the naïve method by using the following R code.

```
adj_fit <- naive(plastics_seasadj,h=12)
plot(adj_fit, plot.conf=TRUE, xlab = 'Year', ylab = 'Monthly sales', main='')
```

The plot obtained from this forecast is presented in Figure 16

Figure 16: A plot of forecasting the seasonal adjusted data with the naïve method

(f) Re-seasonalize to give forecast in original scale

Then we reseasonalized the results by adding in the seasonal naïve forecasts of the seasonal component. This is obtained by the following R code.

```
fcast <- forecast(fit, method="naive")
plot(fcast, xlab = 'Year', ylab = 'Monthly sales', main='')
```

The resulting forecasts of the original data are shown in Figure 17.

Figure 17: A plot of forecasting the reseasonalized data with the naïve method