Time Series Analysis and Forecasting

Problem set 3: Dynamic Models for Prediction

Emilie Krutnes Engen

Master in Big Data Analytics
Carlos III University of Madrid

Universidad
Carlos III de Madrid

Exercise 1

In this exercise we are investigating the appropriate transformation and order of differencing to obtain stationary data.

(a) U.S. Domestic Revenue Enplanements

We first regard the time series data *enplanements* containing data of monthly U.S. domestic revenue enplanements in million $ between 1996 and 2000. To determine whether a transformation is needed we first plot the time series by applying the following R command.

```
plot(enplanements,main='',ylab='Revenue ($ Million)',xlab='Year')
```

From the plot in Figure 1 we see that the data show a clear evidence of seasonality and trend. The plot also reveal an increasing variance. This suggest a transformation of the data is needed to help stabilize the variance of the time series.

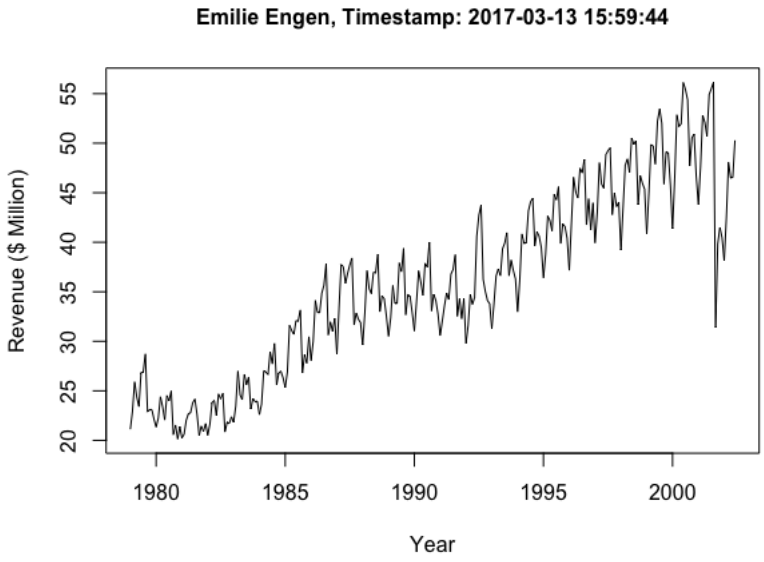

Figure 1: A time series plot of monthly U.S. domestic revenue enplanements in million $

We apply a Box-Cox transformation and plot the transformed time series with the following R command.

```
lambda <- BoxCox.lambda(enplanements)
bc_enplanements <- BoxCox(enplanements,lambda)
plot(bc_enplanements,ylab='Revenue ($ Million)')
```

From the plot in Figure 2 we see that variance is stabilized.

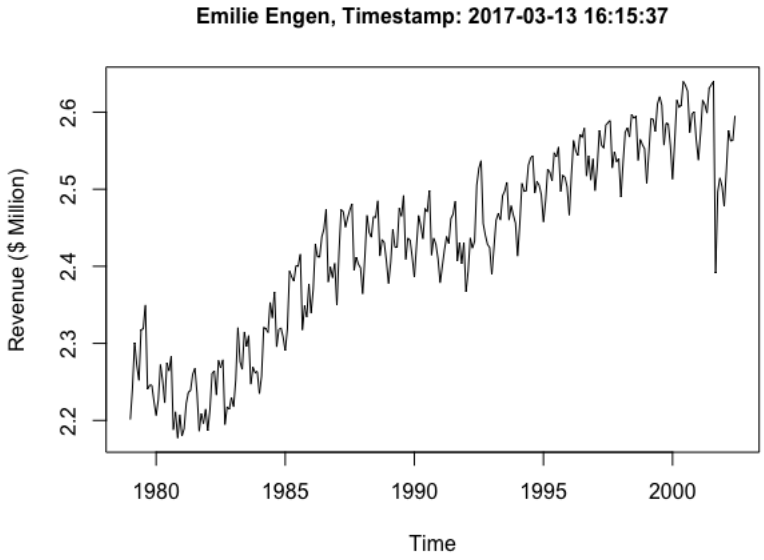

Figure 2: A time series plot of the transformed monthly U.S. domestic revenue enplanements in million $

However, due to the seasonality and trend, the time series is not stationary. Therefore we try differencing the data. The number of lags are set to 12, because of the monthly recorded data. From the following R code we obtain the plot of the differentiated time series and the associated ACF and PACF plots, presented in Figure 3.

```
tsdisplay(diff(bc_enplanements,12),main=paste('Emilie Engen, Timestamp:',Sys.
    time()),cex.main=1)
```

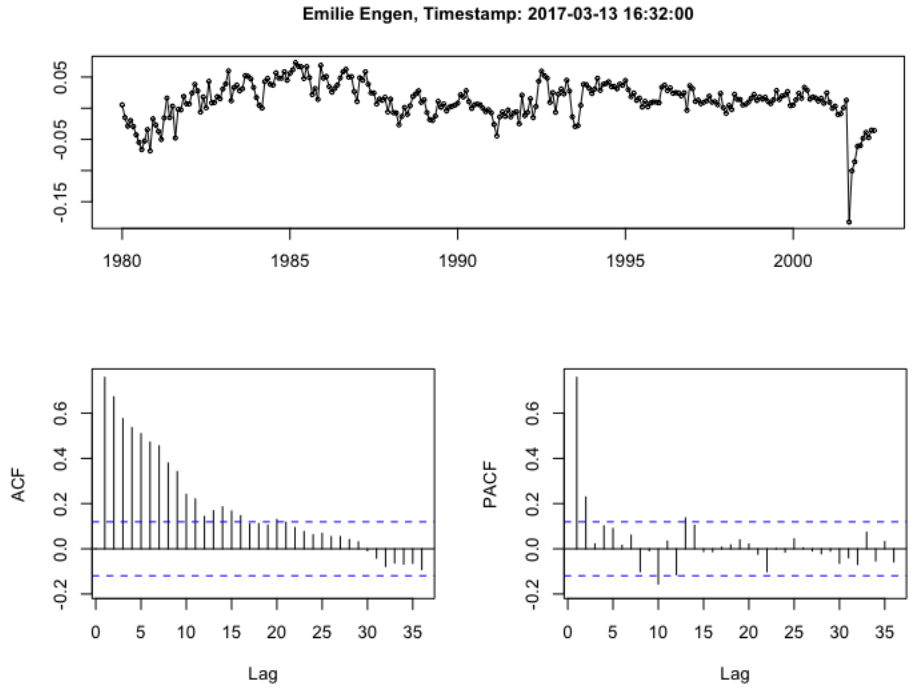

Figure 3: A time series, ACF and PACF plots the differentiated data

There are still a lot of autocorrelation in the data and the ACF and PACF plots suggest a second order difference. We obtain the plots from the second order difference by applying the following R code.

```
tsdisplay(diff(diff(bc_enplanements,12)),main=paste('Emilie Engen, Timestamp:'
    ,Sys.time()),cex.main=1)
```

The plots in Figure 4 suggest that a Box-Cox transformation and second order differencing have made the series look relatively stationary. However, the ACF and PACF plots suggest that there still is some hidden information in the data. We carry out a unit root test to determine more objectively if the data is stationary. This is done by applying Augmented Dickey-Fuller (ADF) test with the following R command.

```
adf.test(diff(diff(bc_enplanements,12)), alternative = "stationary")
```

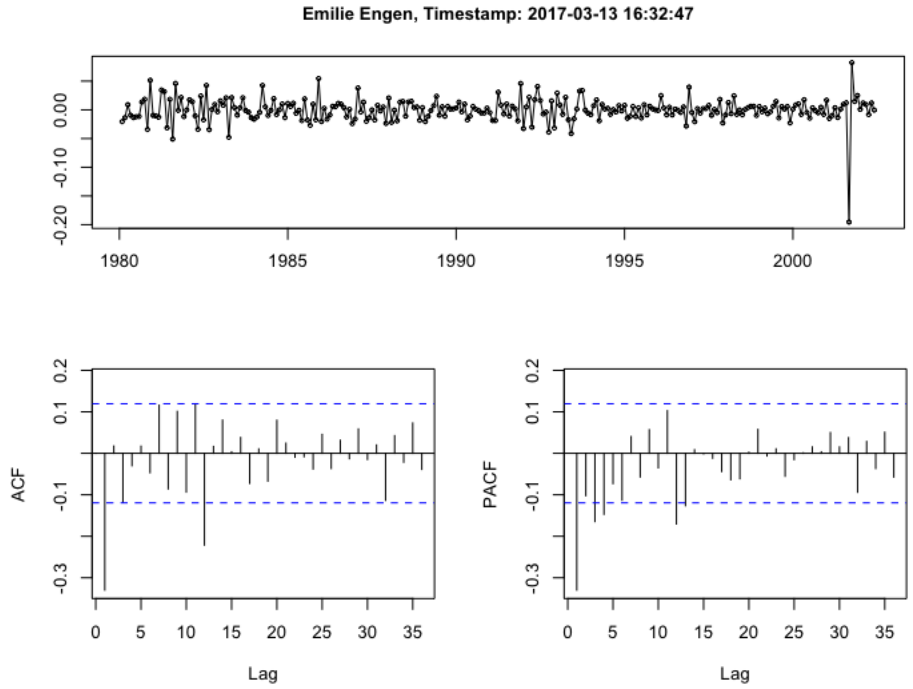

Figure 4: A time series, ACF and PACF plots the second order differentiated data

The null-hypothesis for the ADF test is that the data is non-stationary. From the test we obtain a p-value less than 0.01, which suggest that the data is stationary. With a confidence level of 5 % we reject the null-hypothesis.

(b) Australian Short-Term Overseas Visitors

We now repeat the process for a different time series, *visitors*, containing data of monthly Australian short-term overseas visitors from May 1985 to April 2005. To determine whether a transformation is needed we first plot the time series by applying the following R command.

```
plot(visitors,main='',ylab='Number of visitors (Thousands)',xlab='Year')
```

From the plot in Figure 5 we see that this time series also show a clear evidence of seasonality and trend. The plot also indicate an increasing variance. This suggest a transformation of the data is needed to help stabilize the variance of the time series.

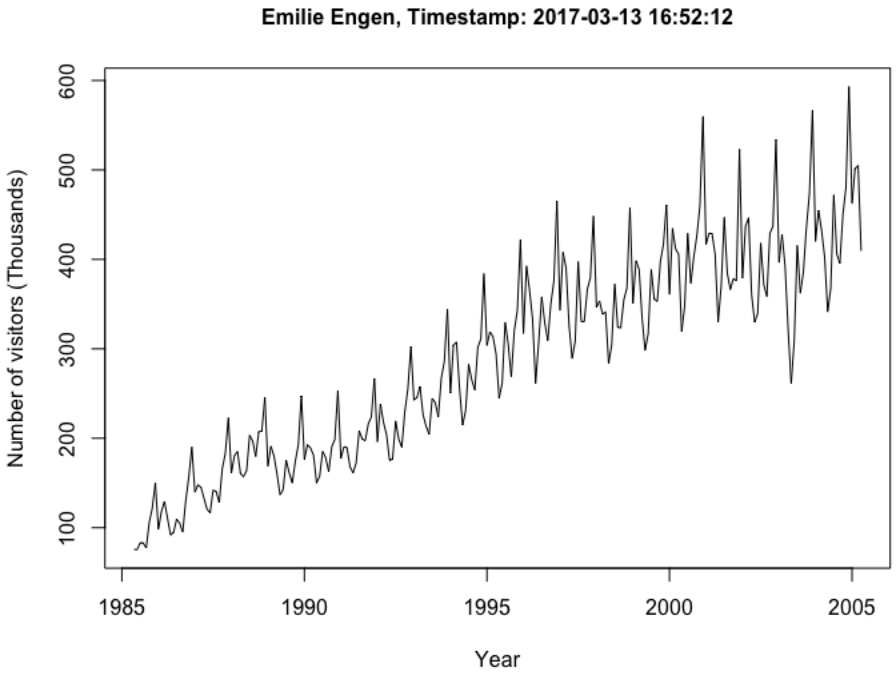

Figure 5: A time series plot of monthly Australian short-term overseas visitors

We apply a Box-Cox transformation and plot the transformed time series with the following R command.

```
lambda <- BoxCox.lambda(visitors)
bc_visitors <- BoxCox(visitors,lambda)
plot(bc_visitors,ylab='Number of visitors (Thousands)')
```

From the plot in Figure 6 we see that variance is stabilized.

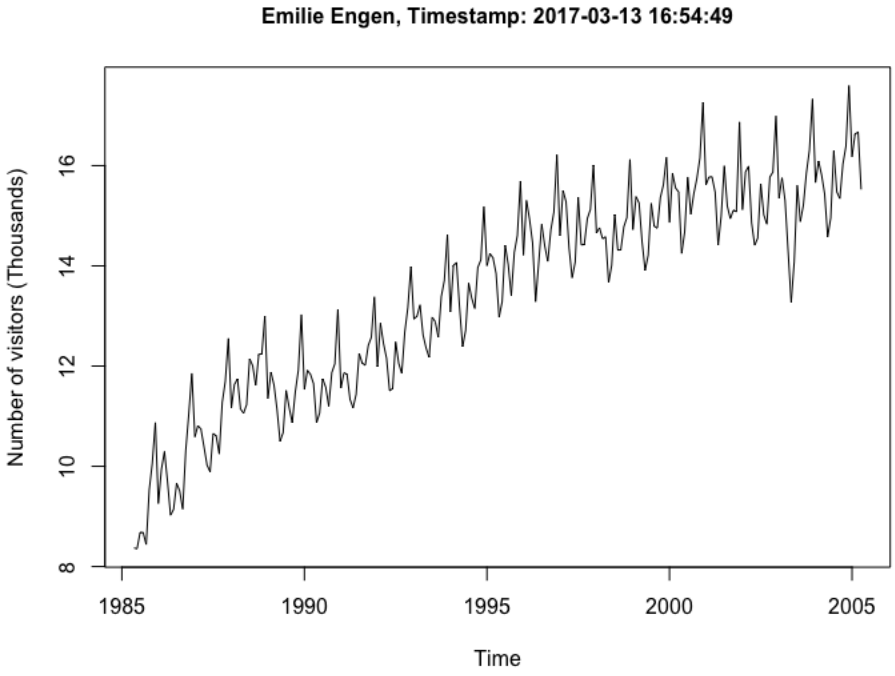

Figure 6: A time series plot of the transformed monthly Australian short-term overseas visitors

However, due to the seasonality and trend, the time series is not stationary. Therefore we try differencing the data. From the following R code we obtain the plot of the differentiated time series and the associated ACF and PACF plots, presented in Figure 7.

```
tsdisplay(diff(bc_visitors,12),main=paste('Emilie Engen, Timestamp:',Sys.time
    ()),cex.main=1)
```

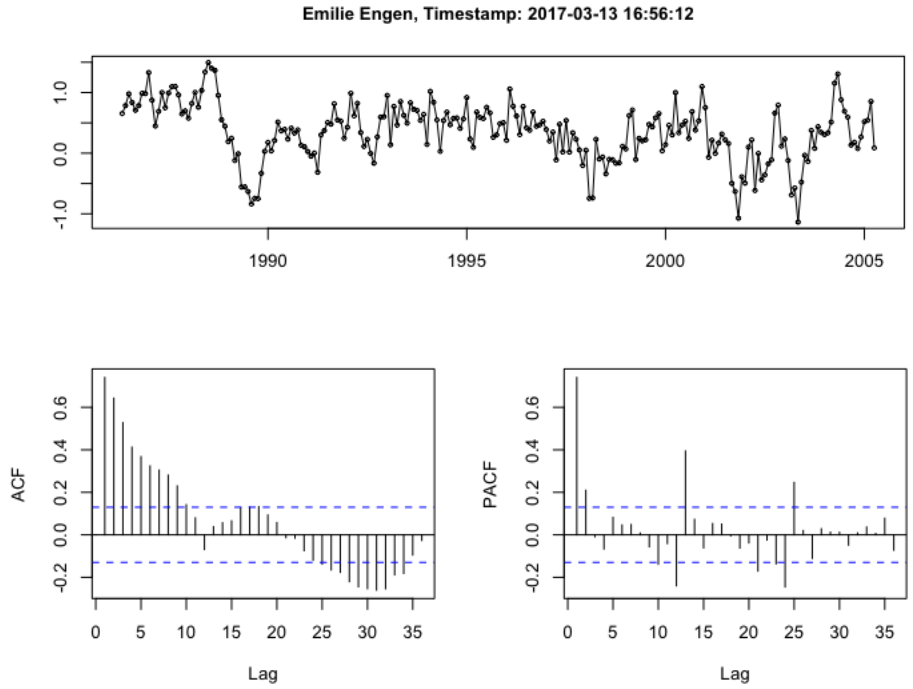

Figure 7: A time series, ACF and PACF plots the differentiated data

There are still a lot of autocorrelation in the data and the ACF and PACF plots suggest a second order difference. We obtain the plots from the second order difference by applying the following R code.

```
tsdisplay(diff(diff(bc_visitors,12)),main=paste('Emilie Engen, Timestamp:',Sys
    .time()),cex.main=1)
```

The plots in Figure 8 suggest that a Box-Cox transformation and second order differencing have made the series look relatively stationary. However, the ACF and PACF plots suggest that there still is some hidden information in the data. We carry out a unit root test to determine more objectively if the data is stationary, using the following R command.

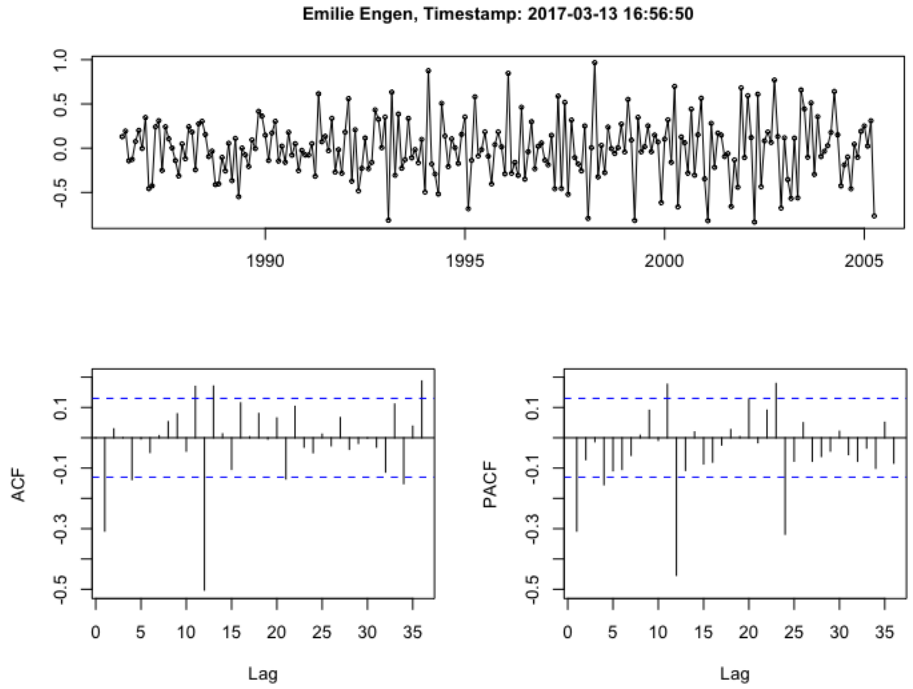

Figure 8: A time series, ACF and PACF plots the second order differentiated data

```
adf.test(diff(diff(bc_visitors,12)), alternative = "stationary")
```

From the test we obtain a p-value less than 0.01. Using a 5 % confidence level, this test suggests that the data is stationary.

Exercise 2

In this exercise we investigate the time series *hsales*, which is monthly sales of new one-family houses sold in the USA since 1973.

(a) Investigating the need for a transformation

We first investigate if any transforming is needed by plotting the time series, by applying the following R command.

```
plot(hsales,main='',ylab='House Sales',xlab='Year')
```

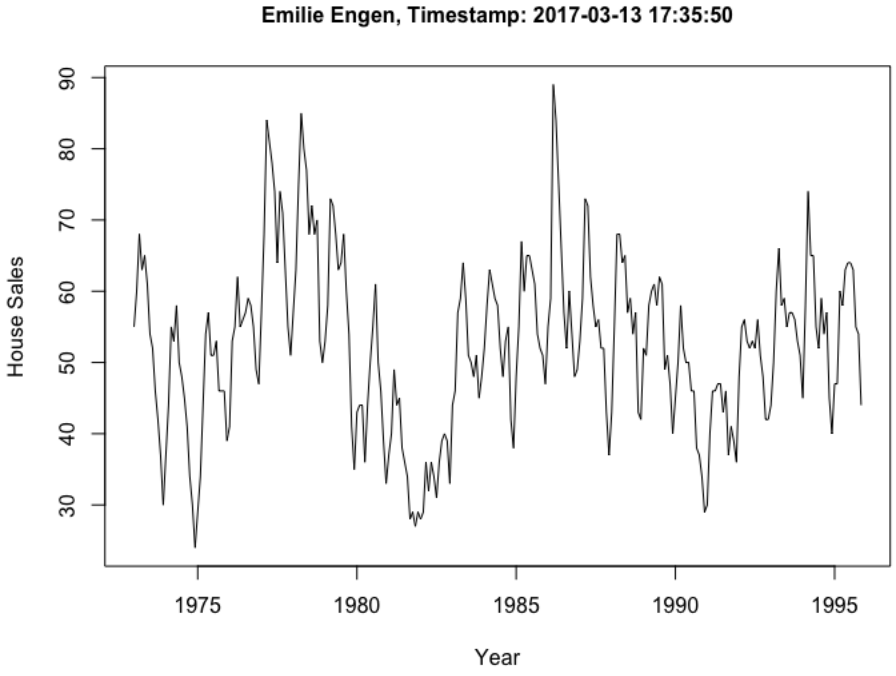

Figure 9: A time series plot of monthly sales of new one-family houses sold in the USA

The plot presented in Figure 9 suggest the occurrence of seasonality and a trend cycle. This is better illustrated by plotting the STL decomposed data, as presented in Figure 10.

```
plot(stl(hsales,'periodic'),main='')
```

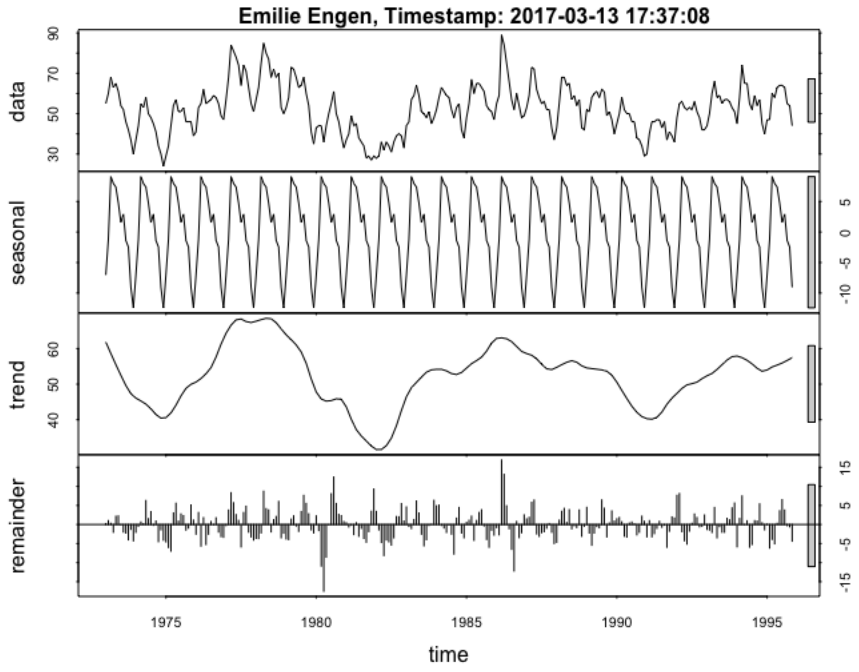

Figure 10: A STL decomposed plot of monthly sales of new one-family houses sold in the USA

From this plot, there is no evidence of variance in the seasonal pattern. Therefore we conclude that no transformation is needed.

(b) Stationary data

Due to the seasonal pattern the time series is non-stationary. We therefore try to difference the data by applying the following R command.

```
tsdisplay(diff(hsales,12),main=paste('Emilie Engen, Timestamp:',Sys.time()),
    cex.main=1)
```

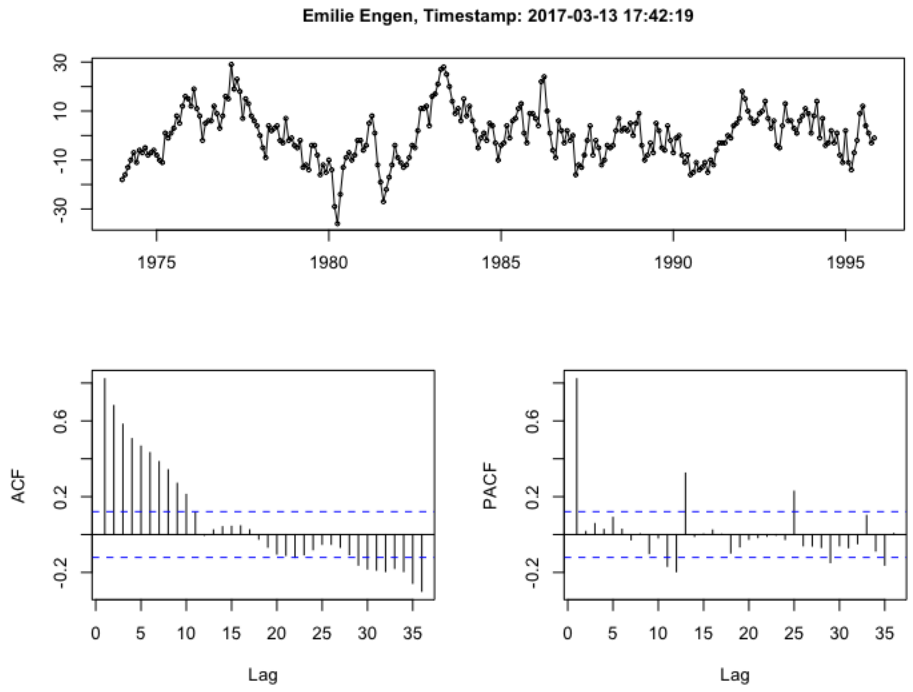

Figure 11: A time series, ACF and PACF plots the differentiated data

The plots obtained in Figure 11 suggest that there is still autocorrelation present in the data. The ACF and PACF plots suggest a second order differencing.

```
tsdisplay(diff(diff(hsales,12)),main=paste('Emilie Engen, Timestamp:',Sys.time
    ()),cex.main=1)
```

The plots in Figure 12 suggest that a second order differencing have made the series look relatively stationary. However, the ACF and PACF plots suggest that there still is some hidden information in the data. We carry out a unit root test to determine more objectively if the data is stationary, using the following R command.

```
adf.test(diff(diff(hsales,12)), alternative = "stationary")
```

From the test we obtain a p-value less than 0.01. Using a 5 % confidence level, this test suggests that the data is stationary.

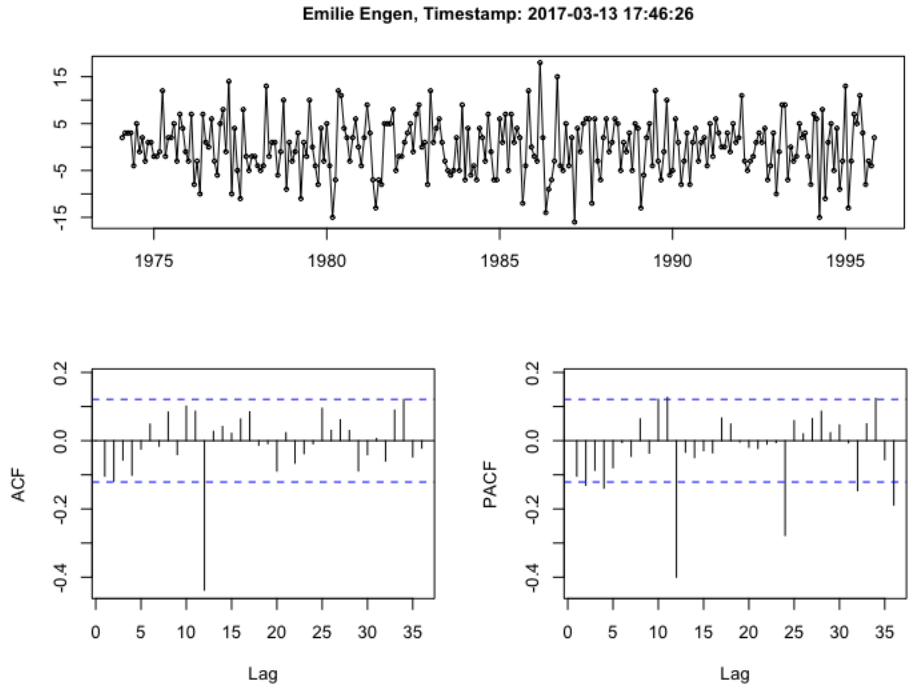

Figure 12: A time series, ACF and PACF plots the second order differentiated data

(c) Identify and evaluate different ARIMA models

We now try to identify some ARIMA models that might be useful in describing the time series. From the ACF plot in Figure 12 we have a significant spike in lag 12 and for the PACF plot we have significant but decreasing spikes in lag 12, 24 and 36. The plots suggest a seasonal component (0,1,1). Further the significant spikes in lag 4 in the PACF suggest a non-seasonal component (4,1,0). The ACF plot show a close to significant spike in lag 2, which suggest a non-seasonal component of (0,1,2). We try these possible models, along with some variation of these, to try to identify the ARIMA model that provide the best fit for our data. The different ARIMA models investigated is presented in the R code below.

```
fit1  <- Arima(hsales, order = c(0, 1, 2), seasonal = c(0, 1, 1))
fit2  <- Arima(hsales, order = c(0, 1, 1), seasonal = c(0, 1, 1))
fit3  <- Arima(hsales, order = c(0, 1, 3), seasonal = c(0, 1, 1))
fit4  <- Arima(hsales, order = c(1, 1, 2), seasonal = c(0, 1, 1))
fit5  <- Arima(hsales, order = c(1, 1, 1), seasonal = c(0, 1, 1))
fit6  <- Arima(hsales, order = c(1, 1, 3), seasonal = c(0, 1, 1))
fit7  <- Arima(hsales, order = c(4, 1, 0), seasonal = c(0, 1, 1))
fit8  <- Arima(hsales, order = c(3, 1, 0), seasonal = c(0, 1, 1))
fit9  <- Arima(hsales, order = c(5, 1, 0), seasonal = c(0, 1, 1))
fit10 <- Arima(hsales, order = c(4, 1, 1), seasonal = c(0, 1, 1))
fit11 <- Arima(hsales, order = c(3, 1, 1), seasonal = c(0, 1, 1))
fit12 <- Arima(hsales, order = c(5, 1, 1), seasonal = c(0, 1, 1))
```

In order to determine the most appropriate ARIMA model, we compare the AIC_c values for all the suggested models. This is done by applying the following R command.

```
c(fit1$aicc, fit2$aicc, fit3$aicc, fit4$aicc, fit5$aicc, fit6$aicc, fit7$aicc,
    fit8$aicc, fit9$aicc, fit10$aicc, fit11$aicc, fit12$aicc)
```

The different AIC_c values are presented in Table 1. From the table we have that *fit5*, which corresponds to ARIMA(1,1,1)(0,1,1), provides the lowest AIC_c values. Hence we proceed by investigating this model further.

ARIMA Model	AIC_c
ARIMA(0,1,2)(0,1,1)	1568.031
ARIMA(0,1,1)(0,1,1)	1568.829
ARIMA(0,1,3)(0,1,1)	1567.288
ARIMA(1,1,2)(0,1,1)	1567.716
ARIMA(1,1,1)(0,1,1)	**1566.055**
ARIMA(1,1,3)(0,1,1)	1568.860
ARIMA(4,1,0)(0,1,1)	1568.719
ARIMA(3,1,0)(0,1,1)	1569.038
ARIMA(5,1,0)(0,1,1)	1569.602
ARIMA(4,1,1)(0,1,1)	1569.806
ARIMA(3,1,1)(0,1,1)	1567.821
ARIMA(5,1,1)(0,1,1)	1569.208

Table 1: AIC_c values for the different ARIMA models

(d) Parameter estimation and residual diagnostics

In order to estimate the parameters of ARIMA(1,1,1)(0,1,1), we apply the following R command.

```
summary(fit5)
```

The ARIMA(1,1,1)(0,1,1) can be expressed by the following equation.

$$y_t = (1 + \phi_1)y_{t-1} + e_t + \theta_1 e_{t-1} \tag{1}$$

where $phi_1 = 0.6352$ and $\theta_1 = -0.7475$. We proceed by investigate the residuals of the chosen ARIMA model. We plot the histogram of the residuals, including the residuals plot, ACF and PACF, by including the following R code.

```
res <- residuals(fit5)
tsdisplay(res, main=paste('Emilie Engen, Timestamp:',Sys.time()),cex.main=1)
hist(res, xlab='Residuals',main='', breaks=20, col='light blue')
```

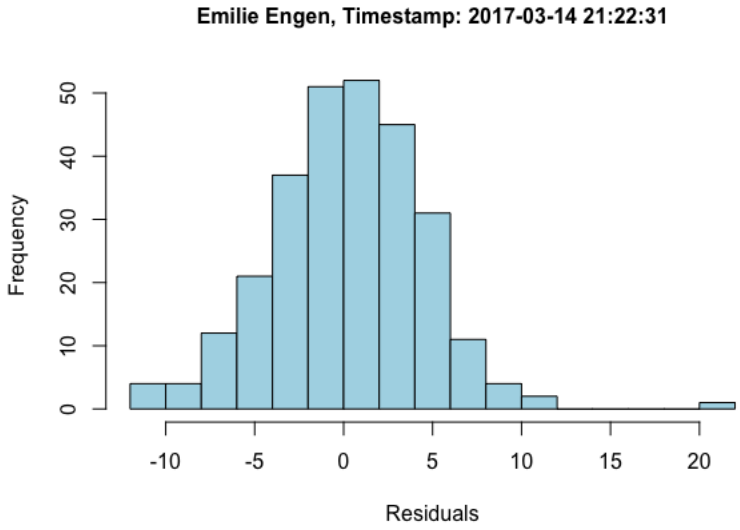

Figure 13: A historgram showing the distribution of the residuals

The histogram in Figure 13 suggest that the residuals follow an approximately normal distribution.

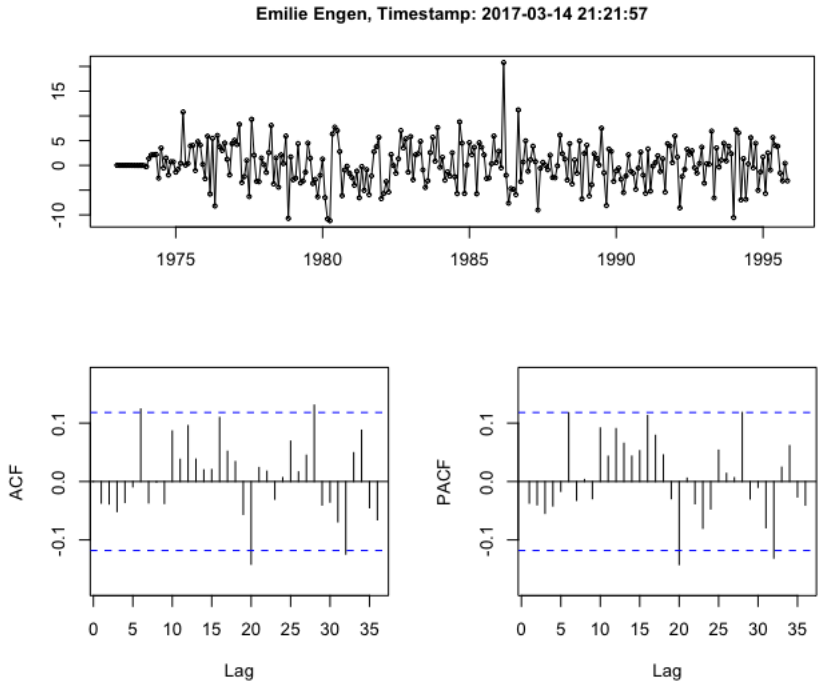

Figure 14: A plot of the residuals, ACF and PACF for the second order differentiated data

According to the ACF and PACF plots all spikes, except one in the PACF plot is within the confidence level. From the Ljung-Box test below we obtain a p-value of 0.1966. Hence the evidence against autocorrelation is not very strong. However the requirements of the residuals are obtained and we therefore proceed, assuming that the residuals remaining are white noise.

```
Box.test(residuals(fit5), lag=24, fitdf=4, type="Ljung")
```

(e) Forecasting

Now that we have an ARIMA model that passes the required test, we proceed by forecasting the next two years of house sales. The forecast is obtained by applying the following R command.

```
fcast <- forecast(fit5, h=24)
plot(fcast,ylab='House Sales',xlab='Year',main='')
```

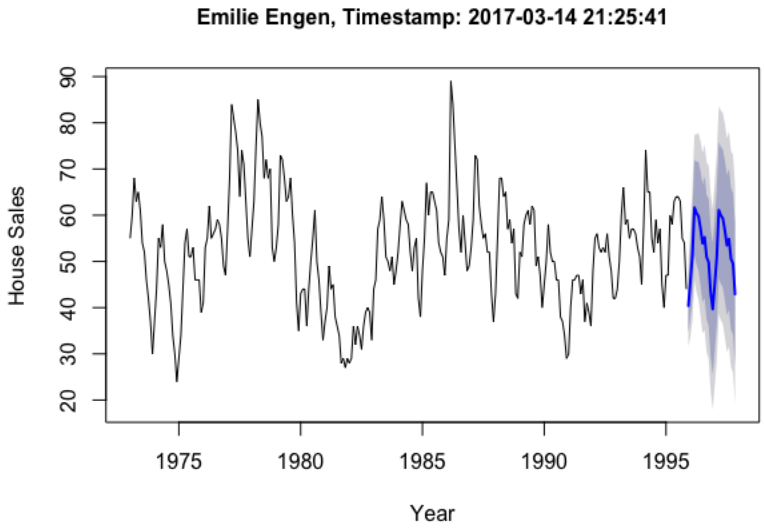

Figure 15: A plot of two year forecast of monthly sales of new one-family houses sold in the USA

The forecast is plotted in Figure 15. Notice that the forecast follow the seasonal pattern. However the trend cycle is not captured. The large and increasing prediction intervals allow the data to trend both upwards or downwards during the forecast period. The increasing prediction interval indicate that the first year forecast is more likely to be close to the previous year, but that the second year forecast might be more affected by the trend.

(f) Apply an Exponential Smoothing Model

We continue the analysis of the ARIMA model by comparing the obtained forecast to a ETS model. The ETS model is obtained by applying the following R code.

```
fit_ets<- ets(hsales)
fcast_ets = forecast(fit_ets, h=24)
```

From the Ljung-Box test we obtain a p-value of 0.4995. This suggest that the ETS model provides a better fit than the ARIMA model.

```
Box.test(residuals(fit_ets), lag=24, fitdf=4, type="Ljung")
```

We plot the two forecast, by applying the following R code.

```
Box.test(residuals(fit_ets), lag=24, fitdf=4, type="Ljung")
```

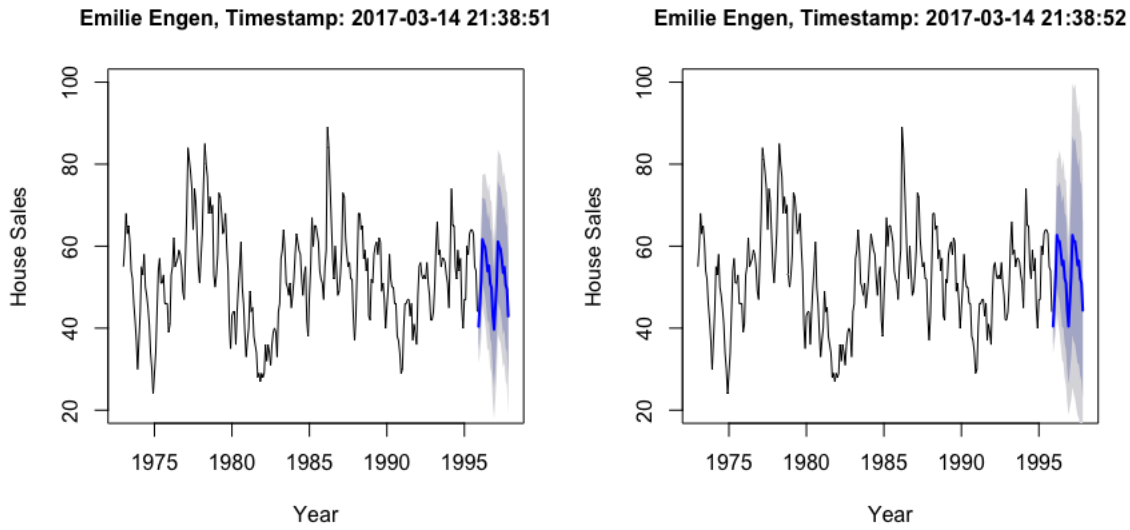

Figure 16: A plot of two year forecast of monthly sales of new one-family houses with the ARIMA model to the left and the ETS to the right

The two plots are presented in Figure 16. The two methods provide very similar forecasts. However, the ETS model provide a wider prediction interval, which suggest that ARIMA model have a higher level of certainty in the forecast.

(g) Apply a non-seasonal model to seasonal adjusted data

We now try using a non-seasonal model applied to the seasonally adjusted data obtained from STL. The forecasts obtained from this model is given in the R code below.

```
fit_stl <- stlf(hsales, robust=TRUE, method=c('arima'))
fcast_stl <- forecast(fit_stl, h=24)
```

The Ljung-Box test suggest the same p-value as for the ETS model. Hence, the test indicate that the two models are considered more appropriate than the ARIMA model.

```
Box.test(residuals(fit_ets), lag=24, fitdf=4, type="Ljung")
```

We plot the three forecasts by applying the following R code.

```
par(mfrow=c(1,3))

plot(fcast, ylim = c(20,100),ylab='House Sales',xlab='Year',main='')
plot(fcast_ets, ylim = c(20,100),ylab='House Sales',xlab='Year',main='')
plot(fcast_stl, ylim = c(20,100),ylab='House Sales',xlab='Year',main='')
```

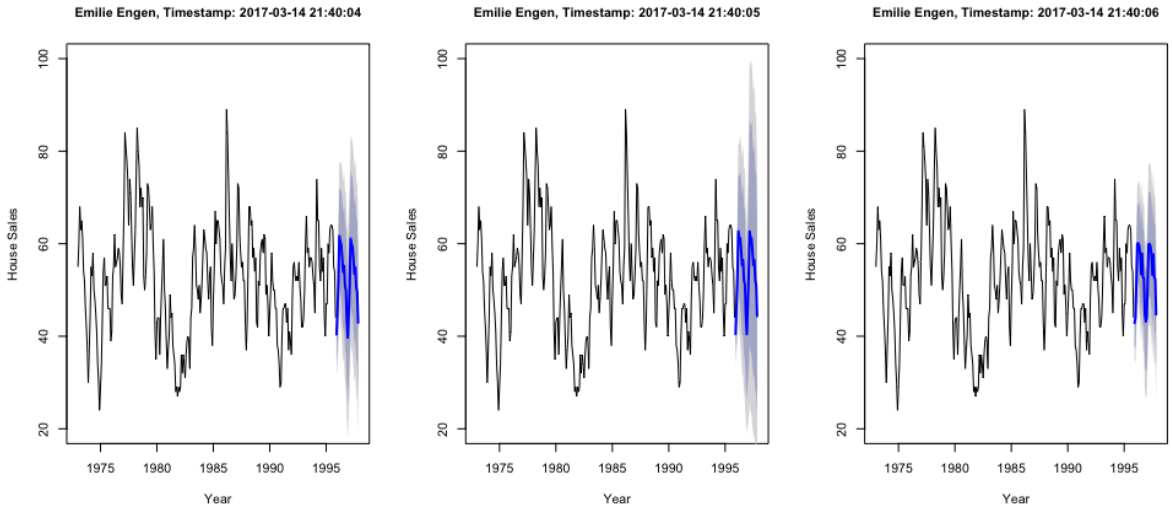

Figure 17: A plot of two year forecast of monthly sales of new one-family houses with the ARIMA, the ETS and the STL model

The plots are presented in Figure 17. The plots suggest that the forecasts obtained from the STL decomposition is slightly different from the two previous models. The ARIMA and ETS forecasts also provide a wider prediction interval than the STL model. The Ljung-Box test suggest that the ETS or STL should be selected. When comparing the three models by looking at the AIC_c values of the different models we have that the AIC_c values are 1566.055, 2377.106 and 1544.16 for the ARIMA, the ETS and the STL model, respectively. This suggest that the STL model provides the best fit for this dataset. From looking at the forecasts we see that the forecasts are quit similar but that the prediction interval from the STL model is more narrow than the two other models. The characteristics of the three models are obtained by applying the following R code.

```
summary(fcast)
summary(fcast_ets)
summary(fcast_stl)
```

Exercise 3

In this exercise we want to investigate the behavior of some simple ARIMA models by applying it to random normally distributed data.

(a) Generate time series data from AR(1)

The autoregression (AR) model forecast the variable of interest using a linear combination of past values of the variable, which means that it is a regression of the variable against itself. The AR(1) predicts y_t only based on the previous value y_{t-1} and can be expressed by the following equation.

$$y_t = c + \phi_1 y_{t-1} + e_t \tag{2}$$

where c is a constant and e_t is the white noise. We first apply the following code to generate data from an autoregressive model with one parameter, AR(1), for normally distributed random data. Hence the data has zero mean ($\mu = 0$) and variance one ($\sigma^2 = 1$). The initial value $y_0 = 0$ and the parameter $\phi_1 = 0.6$ This is obtained by applying the following R code.

```
set.seed(0)
y <- ts(numeric(100))
e <- rnorm(100)

for(i in 2:100)
    y[i] <- 0.6*y[i-1] + e[i]
```

We use the set.seed(0) function to make sure the white noise is the same for all models investigated in this exercise.

(b) Time series plot from AR(1)

We first plot the time series from AR(1) by applying the following R code.

```
plot(y,main='',ylim=c(-3,3))
```

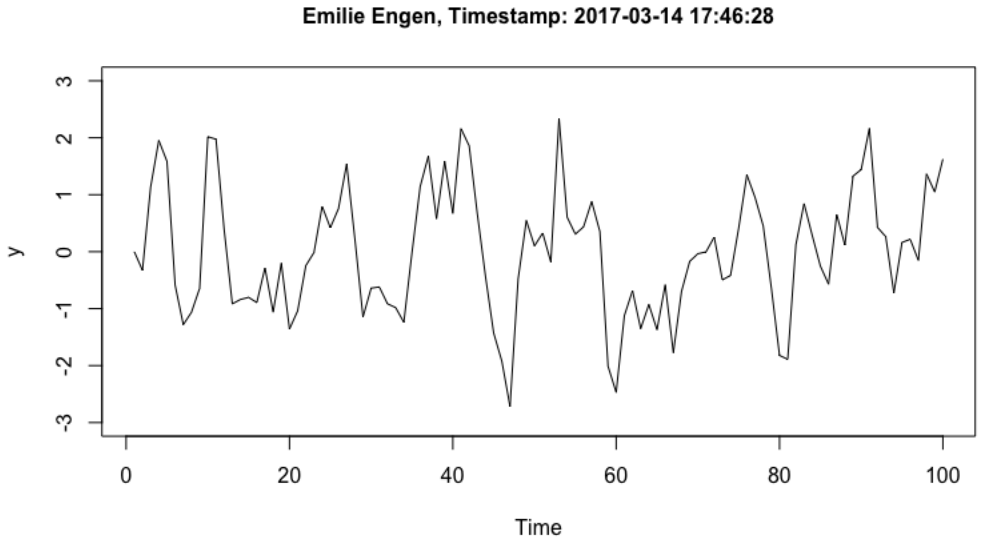

Figure 18: A plot of the data generated from AR(1) with normally distributed random data

To investigate the model for changing ϕ_1 values, we apply the same formula using different values for the ϕ_1 parameter.

```
y1 <- ts(numeric(100))
y2 <- ts(numeric(100))
y3 <- ts(numeric(100))
y4 <- ts(numeric(100))
y5 <- ts(numeric(100))

for(i in 2:100)
    y1[i] <- -0.8*y1[i-1] + e[i]

for(i in 2:100)
    y2[i] <- -0.4*y2[i-1] + e[i]

for(i in 2:100)
    y3[i] <- 0*y3[i-1] + e[i]

for(i in 2:100)
    y4[i] <- 0.4*y4[i-1] + e[i]

for(i in 2:100)
    y5[i] <- 0.8*y5[i-1] + e[i]
```

Here we are fitting the AR(1) model with ϕ_1 ranging from -0.8 to 0.8 with a step lenght of 0.4. We plot the models by applying the following R code.

```
plot(y,main='',ylim=c(-3,3))
lines(y1, col=5, lty=2)
lines(y2, col=3, lty=2)
lines(y3, col=4, lty=2)
lines(y4, col=2, lty=2)
lines(y5, col=6, lty=2)
legend('topleft',lty=1, col=c(1,5,3,4,2,6), c(expression(phi == 0.6),
      expression(phi == -0.8), expression(phi == -0.4), expression(phi == 0.0)
      , expression(phi == 0.4), expression(phi == 0.8)), pch=1, cex=0.8)
```

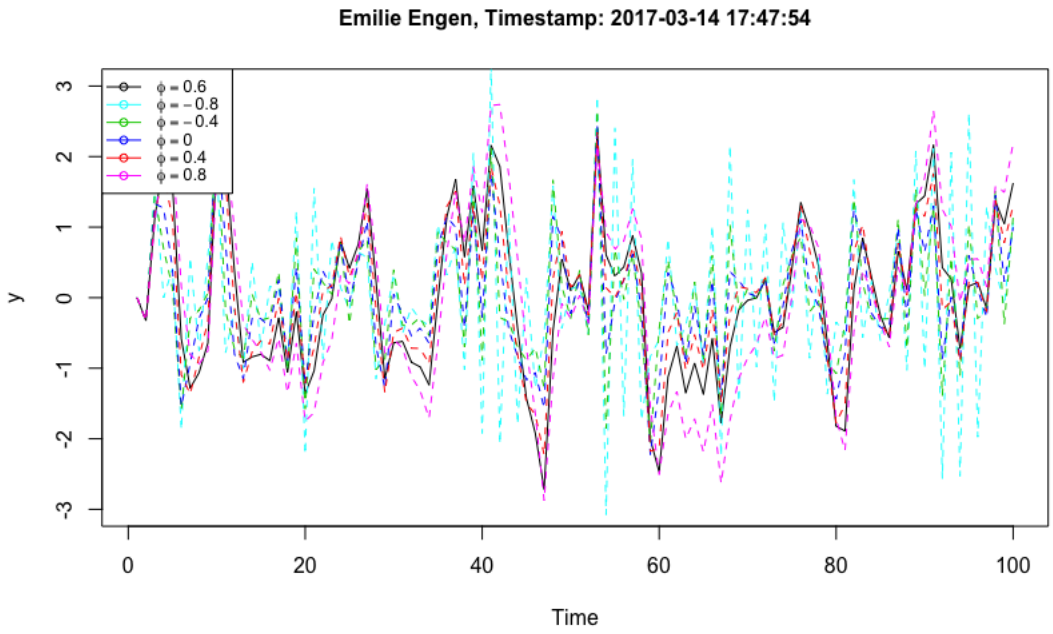

Figure 19: A plot of the data generated from AR(1) for different values of the ϕ_1 parameter

The plots are presented in Figure 19. From the plot we see that the different values for the ϕ_1 parameter change the pattern of the time series. For $\phi_1 = 0$ the model is equivalent to white noise. For increasing ϕ_1 values the model becomes more similar to a random walk. When ϕ_1 decreases below zero the frequency of the oscillations increases.

(c) Generate time series data from MA(1)

Unlike the autoregression model, the moving average (MA) model uses past forecast errors to predict y_t. This is not really a regression, since the white noise is not observed. The MA(1) model can be expressed by the following equation.

$$y_t = c + e_t + \theta e_{t-1} \tag{3}$$

We now generate data from a moving average model with one parameter, MA(1). The initial value $y_0 = 0$ and the parameter $\theta_1 = 0.6$. This is obtained by applying the following R code.

```
y <- ts(numeric(100))

for(i in 2:100)
    y[i] <- e[i] + 0.6*e[i-1]
```

The white noise e is the same as for the previous model.

(d) Time series plot from MA(1)

We first plot the time series from MA(1) by applying the following R code.

```
plot(y,main='',ylim=c(-3,3))
```

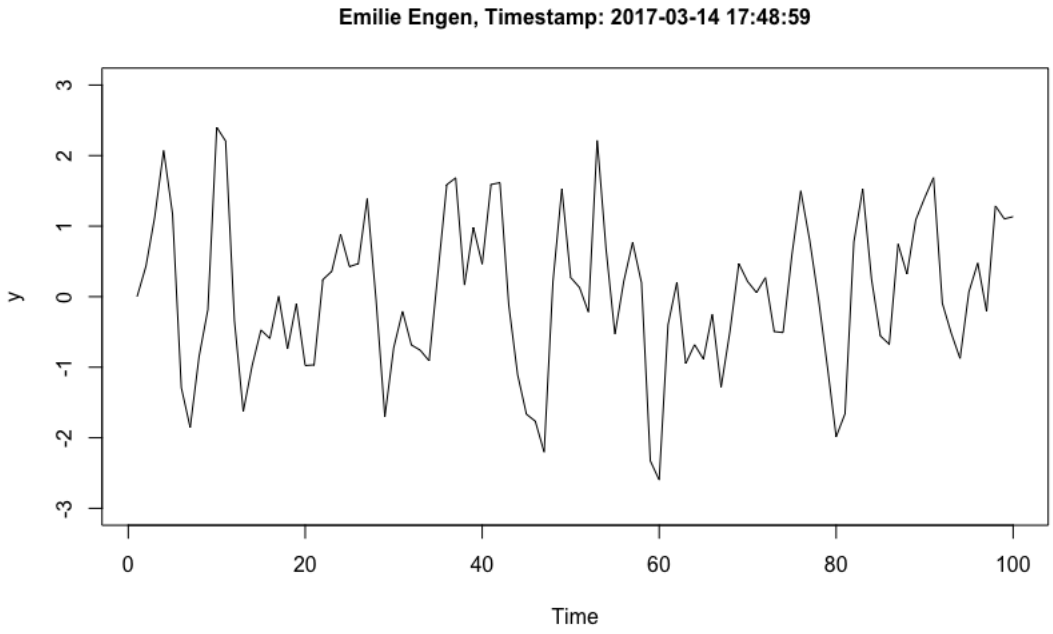

Figure 20: A plot of the data generated from MA(1) with normally distributed random data

To investigate the model for changing θ_1 values, we apply the same formula using different values for the θ_1 parameter.

```
y1 <- ts(numeric(100))
y2 <- ts(numeric(100))
y3 <- ts(numeric(100))
y4 <- ts(numeric(100))
y5 <- ts(numeric(100))

for(i in 2:100)
    y1[i] <- e[i] - 0.8*e[i-1]
```

```
for(i in 2:100)
    y2[i] <- e[i] - 0.4*e[i-1]

for(i in 2:100)
    y3[i] <- e[i] + 0*e[i-1]

for(i in 2:100)
    y4[i] <- e[i] + 0.4*e[i-1]

for(i in 2:100)
    y5[i] <- e[i] + 0.8*e[i-1]
```

Here we are fitting the MA(1) model with θ_1 ranging from -0.8 to 0.8 with a step lenght of 0.4. We plot the models by applying the following R code.

```
plot(y,main='',ylim=c(-3,3))
lines(y1, col=5, lty=2)
lines(y2, col=3, lty=2)
lines(y3, col=4, lty=2)
lines(y4, col=2, lty=2)
lines(y5, col=6, lty=2)
legend('topleft',lty=1, col=c(1,5,3,4,2,6), c(expression(theta == 0.6),
      expression(theta == -0.8), expression(theta == -0.4),
      expression(theta == 0.0), expression(theta == 0.4),
      expression(theta == 0.8)), pch=1, cex=0.8)
```

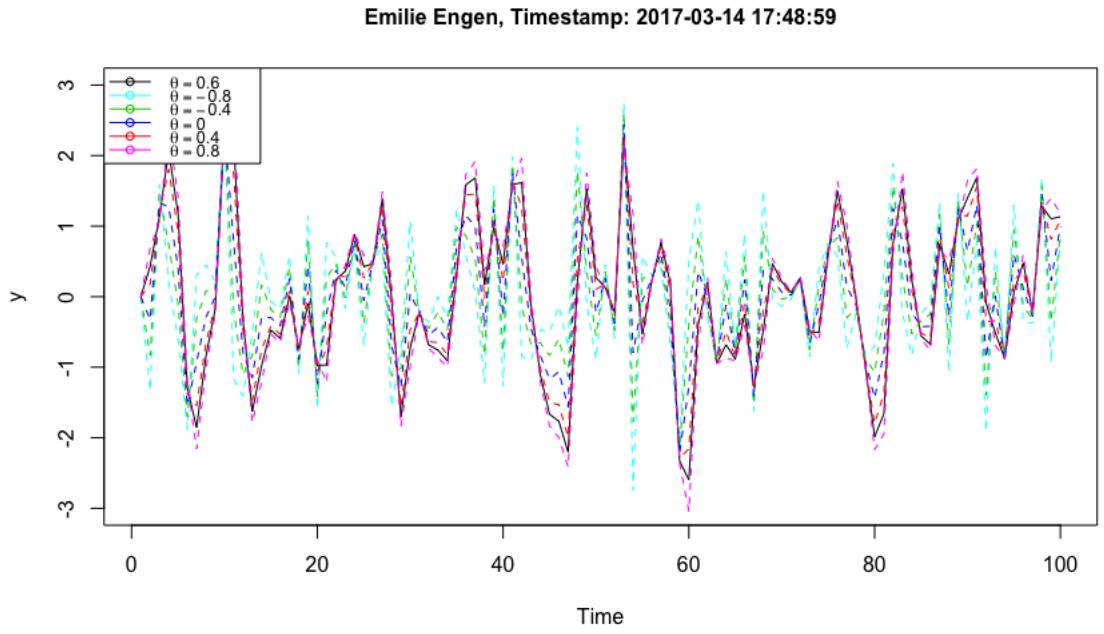

Figure 21: A plot of the data generated from MA(1) for different values of the θ_1 parameter

The plots are presented in Figure 21. From the plot we see that changing the value θ_1 parameter have similar effects in the MA model. Changing the parameter only change the

pattern and not the scale of the time series. Changes in the white noise would result in a different pattern.

(e) Generate time series data from ARMA(1,1)

If we combine an autoregression and a moving average model, we obtain a autoregressive moving average model (ARMA). In the ARMA model the forecasts are based both on previous values of the variable and the errors. We now generate data from ARMA(1,1) with one parameter, ϕ_1, for the autoregressive part and one parameter, θ_1 for the moving average part. The initial value $y_0 = 0$ and the parameters $\phi_1 = \theta_1 = 0.6$. This is obtained by applying the following R code.

```
y_arma <- ts(numeric(100))

for(i in 2:100)
    y_arma[i] <- e[i] + 0.6*e[i-1] + 0.6*y_arma[i-1]}
```

(f) Generate time series data from AR(2)

We now generate data from a autoregressive model with two parameters, AR(2). The initial value $y_0 = 0$ and the parameters $\phi_1 = -0.8$ and $\phi_2 = 0.3$. This is obtained by applying the following R code.

```
y_ar <- ts(numeric(100))

for(i in 3:100)
    y_ar[i] <- -0.8*y_ar[i-1] + 0.3*y_ar[i-2] + e[i]
```

(g) Time series plot from ARMA(1,1) and AR(2)

The time series from the ARMA(1,1) and AR(2) models are plotted by applying the following R code.

```
par(mfrow=c(1,2))

plot(y_arma,ylab='ARMA(1,1)',main='')
plot(y_ar,ylab='AR(2)',main='')
```

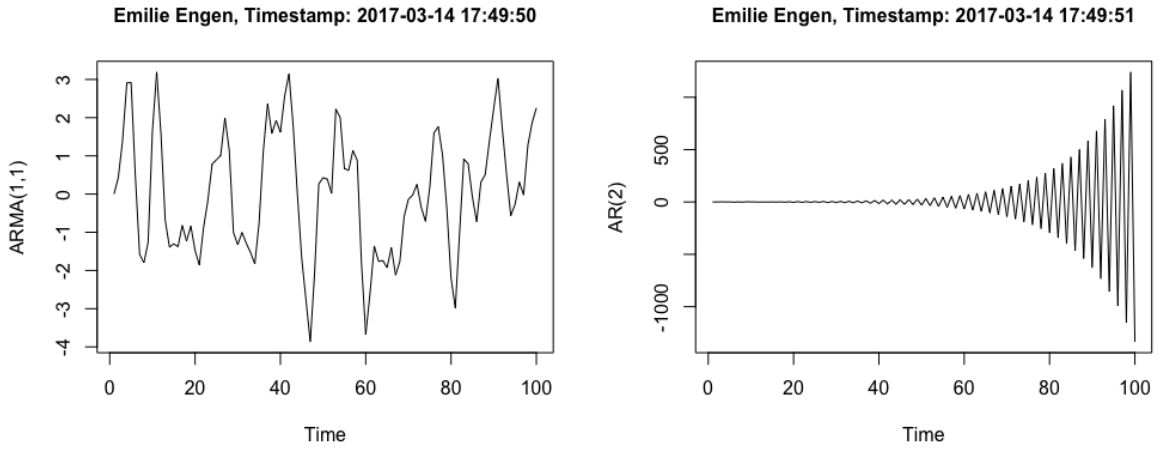

Figure 22: A plot of the data generated from the ARMA(1,1) and AR(2) model

The two plots are presented in Figure 22. From the plots we see that both models have a mean equal to zero, but for the AR(2) the variance is increasing. Thus the ARMA(1,1) appears to be stationary, while the AR(1) is non-stationary as the variance is increasing exponentially.