APPLIED TIME SERIES ANALYSIS

Assignment 1

Group Number：07

Group Member：

He Guanzhou	A0153091M
Hu Hao	A0152988L
Hu Jiongyi	A0153190M
Jiang Zeyu	A0152953B
Ye Rong	A0153626E

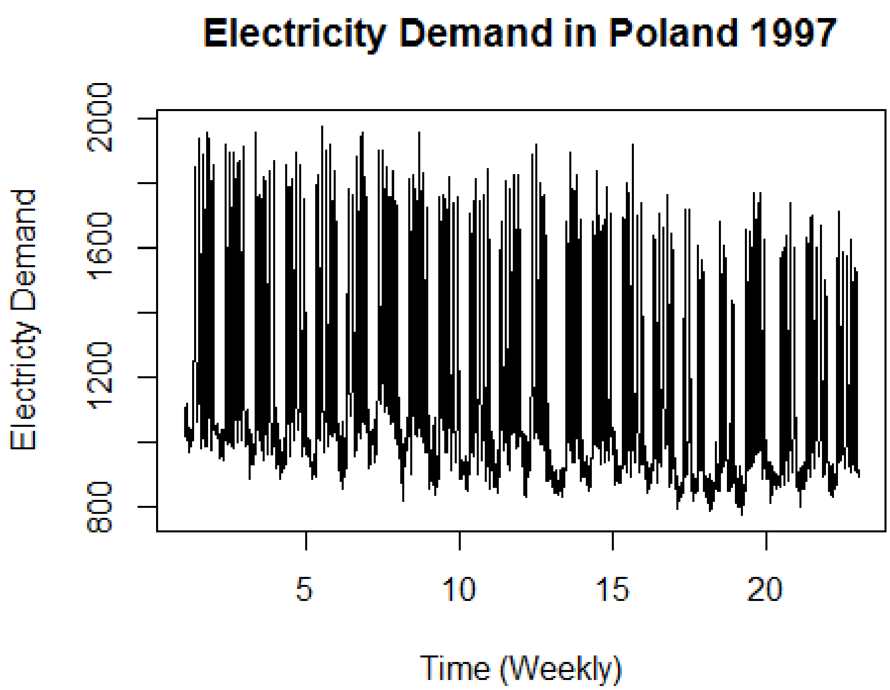
Exercise 1 (Electricity Forecast)

1. Load the data and use the command ts(…) to convert it into a time series object:

```
electricity_load <- read.table("E:/StatSoft/electricity_load.dat", quot
e="\"", comment.char="")
head(electricity_load)

##     V1
## 1 1087
## 2 1032
## 3 1018
## 4 1031
## 5 1057
## 6 1024

electricity_ts<-ts(electricity_load$V1, start = 1, frequency = 24*7)
head(electricity_ts)

## [1] 1087 1032 1018 1031 1057 1024
```

2. Display the time series:

```
plot(electricity_ts, ylab="Electricty Demand", xlab = "Time (Weekly)",
main="Electricity Demand in Poland 1997")
```



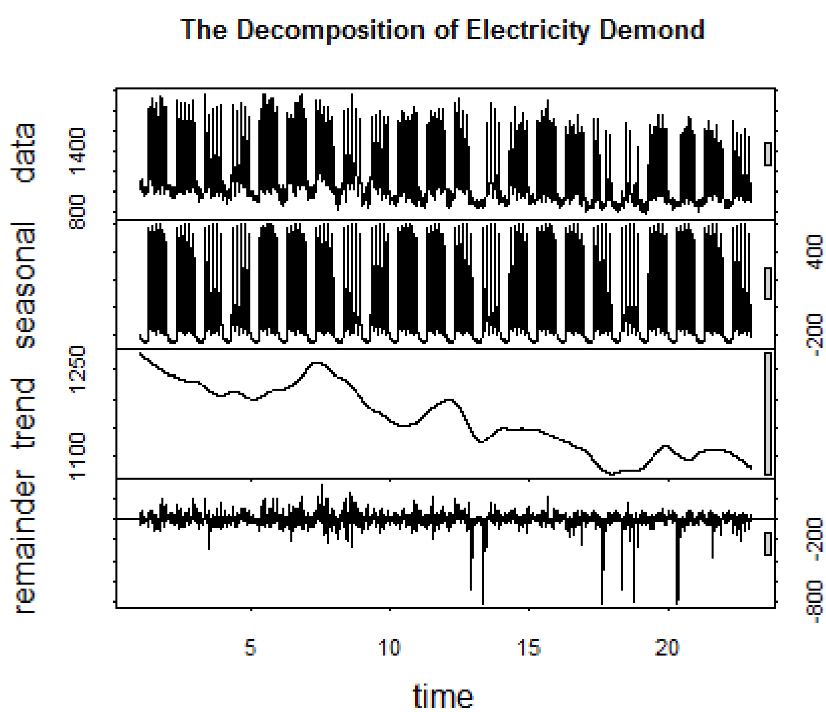
As we can see in the graph, an appropriate seasonal period is ONE unit on the Y-axis. Because we choose "24 * 7" as the frequency, the period should be ONE WEEK according to the data.

3. Decompose the time series into the superposition of a trend / seasonal / remainder pattern:

```
## Loading required package: forecast

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: timeDate

## This is forecast 7.1

## Loading required package: fma

## Loading required package: tseries

## Loading required package: expsmooth

## Loading required package: lmtest
```

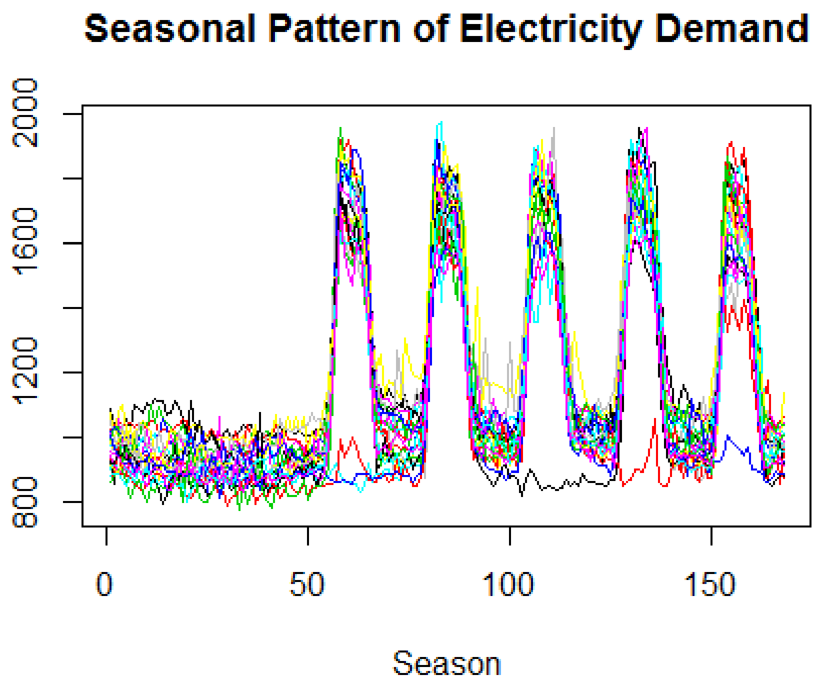
We use package "fpp" to decompose the time series.

```
fit<-stl(electricity_ts, s.window="periodic", robust=T)
plot(fit, main="The Decomposition of Electricity Demond")
```

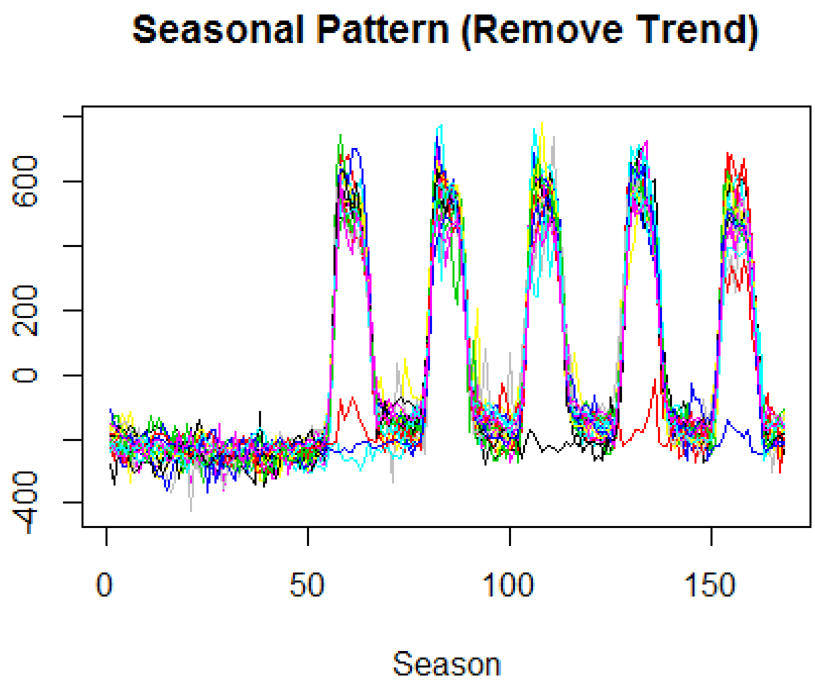


4. Display the seasonal pattern:

```
library(forecast)
seasonplot(electricity_ts, s=24*7,col=1:23,type="l", main="Seasonal Pat
tern of Electricity Demand")
```

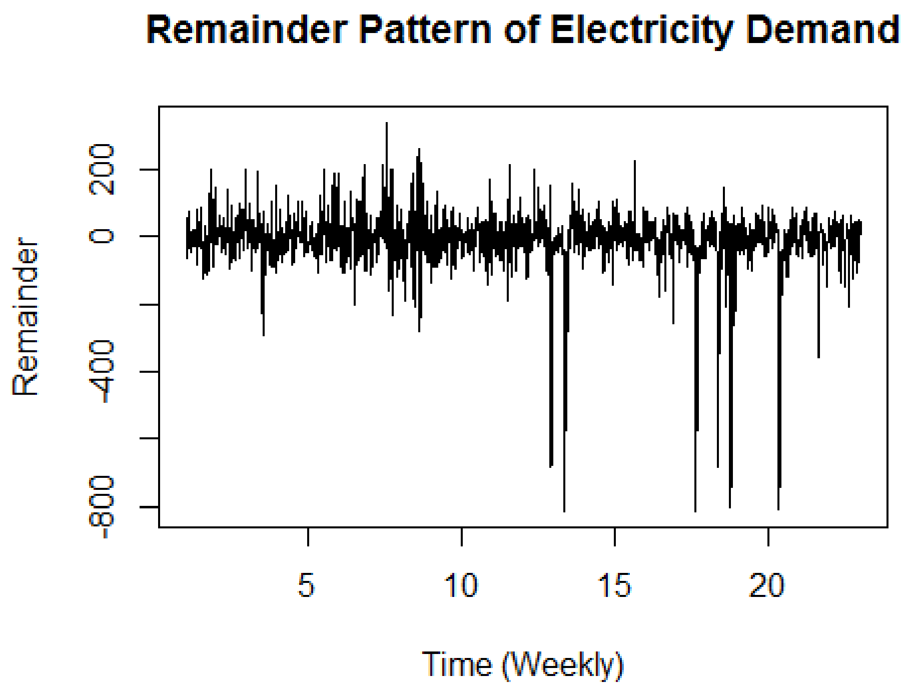



```
seasonplot(electricity_ts-fit$time.series[,"trend"], s=24*7,col=1:23,ty
pe="l", main="Seasonal Pattern (Remove Trend)")
```



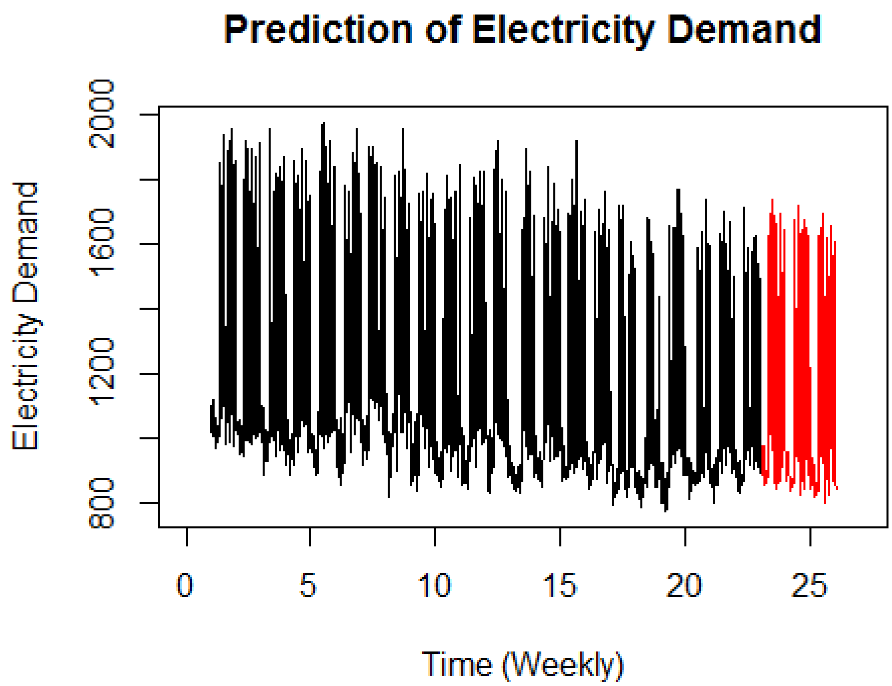
In the remainder pattern, we can see there are some huge residual value at some point. Thus, the time series is not perfectly seasonal.

```
plot(fit$time.series[,"remainder"], main="Remainder Pattern of Electric
ity Demand", ylab="Remainder", xlab="Time (Weekly)")
```



5. Make a forecast for the next 3 weeks:

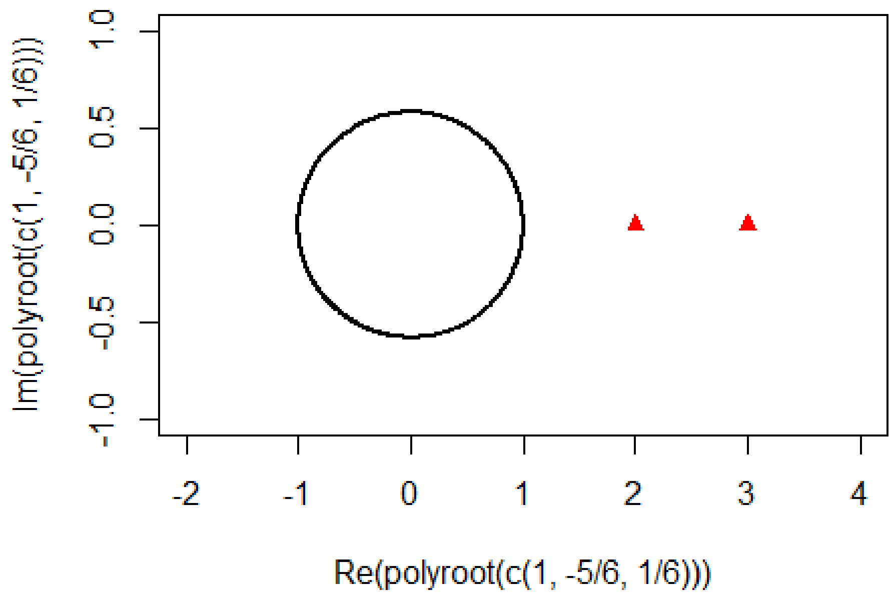
```
ele_optim<-hw(electricity_ts, initial = "simple", seasonal = "additive
", h=24*7*3)
plot(electricity_ts, main="Prediction of Electricity Demand", ylab="Ele
ctricity Demand", xlab="Time (Weekly)", xlim=c(0,27))
lines(ele_optim$mean,col="red", type="l", lwd=1)
```



Exercise 2 (Bootstrap Estimate)

1. The MA(2) Process is invertible. This is because all the roots of the characteristic polynomial are greater than one.

```
library(shape)
plot(polyroot(c(1,-5/6,1/6)), pch=17, col="red", xlim=c(-2,4), ylim=c(-
1,1))
plotcircle(r=1, mid=c(0,0), lwd=2)
```



2. Simulate a trajectory and double check:

```
MA2<-arima.sim(n=1000, list(ma=c(-5/6,1/6)), sd=1)
MA2_test<-arima(MA2, order=c(0,0,2))
summary(MA2_test)

##
## Call:
## arima(x = MA2, order = c(0, 0, 2))
##
## Coefficients:
##           ma1     ma2  intercept
##       -0.8588  0.1915     0.0071
## s.e.   0.0309  0.0305     0.0106
##
## sigma^2 estimated as 1.014:  log likelihood = -1426.35,  aic = 2860.
71
##
## Training set error measures:
##                         ME      RMSE       MAE       MPE     MAPE
  MASE
## Training set -9.175537e-05 1.007037 0.8061325 -82.80622 355.2077 0.4
182559
##                      ACF1
## Training set -0.002422805
```

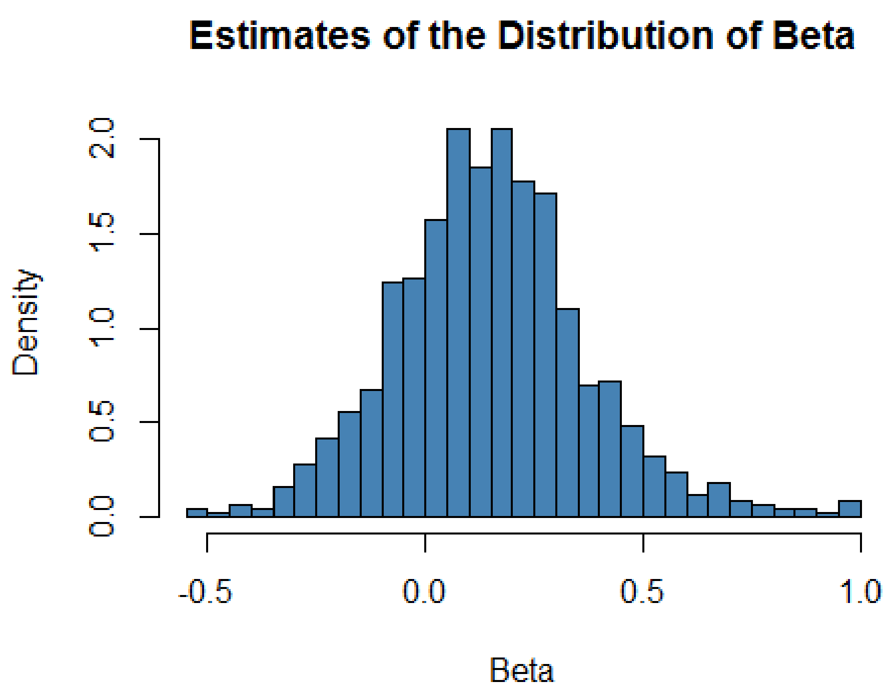
The coefficients of MA1 and MA2 are closed to -5/6 and 1/6, so the estimation procedure provides a good estimate.

3. Estimate the accuracy of the estimation procedure for short time series:

```
beta<-c()
for (i in 1:1000)
{
  MA2_short_time<-arima.sim(n=40, list(ma=c(-5/6,1/6)), sd=1)
  MA2_st_test<-arima(MA2_short_time, order=c(0,0,2))
  beta[i]<-MA2_st_test$coef[2]
}
head(beta)

## [1]  0.240748317  0.183289360  0.144419602  0.578006280 -0.006454145
## [6]  0.102763945
```

4. Summary of the Estimates of Beta:

```
hist(beta, nclass=50, probability = T, col="steelblue", xlab="Beta", ma
in="Estimates of the Distribution of Beta")
```




```
mean(beta)

## [1] 0.1497037

var(beta)

## [1] 0.04911213
```

5. Estimate of Alpha:

```
MA1<-arima.sim(n=10000, list(ma=c(3)), sd=1)
MA1_test<-arima(MA1, order=c(0,0,1))
summary(MA1_test)

##
## Call:
## arima(x = MA1, order = c(0, 0, 1))
##
## Coefficients:
##          ma1  intercept
##       0.3312    -0.0061
## s.e.  0.0094     0.0400
##
## sigma^2 estimated as 9.013:  log likelihood = -25183.04,  aic = 5037
2.08
##
## Training set error measures:
##                        ME     RMSE      MAE      MPE     MAPE      MA
SE
## Training set 3.98034e-05 3.002244 2.397005 70.78541 202.8658 0.80733
23
##                     ACF1
## Training set 0.001553553
```

That is because we use the Autocorrelation Function to estimate the coefficient Alpha, and the model with Alpha = 3 and Alpha=1/3 have the same ACF. However, only the model with Alpha = 1/3 is invertible. So the "arima" function in R shows that Alpha is closed to 1/3.

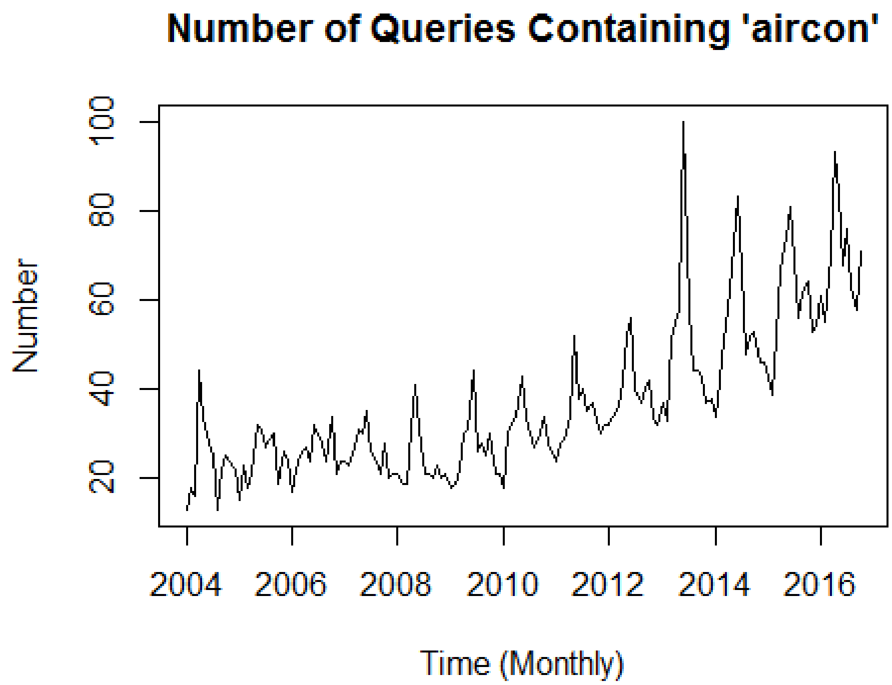
Exercise 3 (Sale Forcast)

1. Get the data and generate the time series: We download the data for Google Trend. The data starts in 2004, counting monthly.

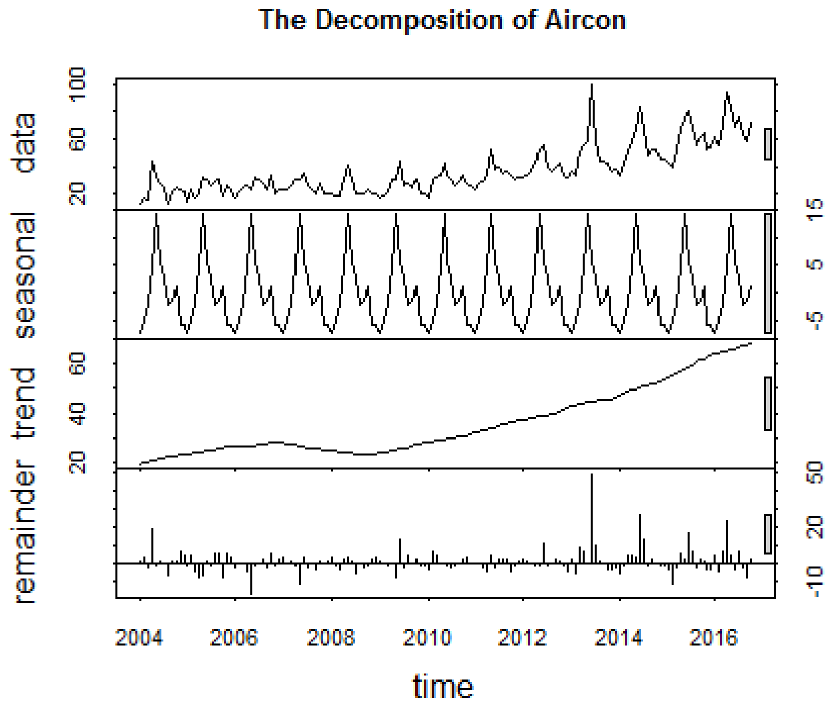
```
aircon_data<-read.csv("E:/StatSoft/air_con.csv", header = FALSE, skip=
3)
head(aircon_data)

##        V1 V2
## 1 2004-01 13
## 2 2004-02 18
## 3 2004-03 16
## 4 2004-04 44
## 5 2004-05 34
## 6 2004-06 28

aircon_ts<-ts(aircon_data$V2, start = c(2004,1), frequency = 12)
head(aircon_ts)

## [1] 13 18 16 44 34 28
```

2. Plot the time series and decompose it:

```r
plot(aircon_ts, main="Number of Queries Containing 'aircon'", ylab="Number", xlab="Time (Monthly)")
```



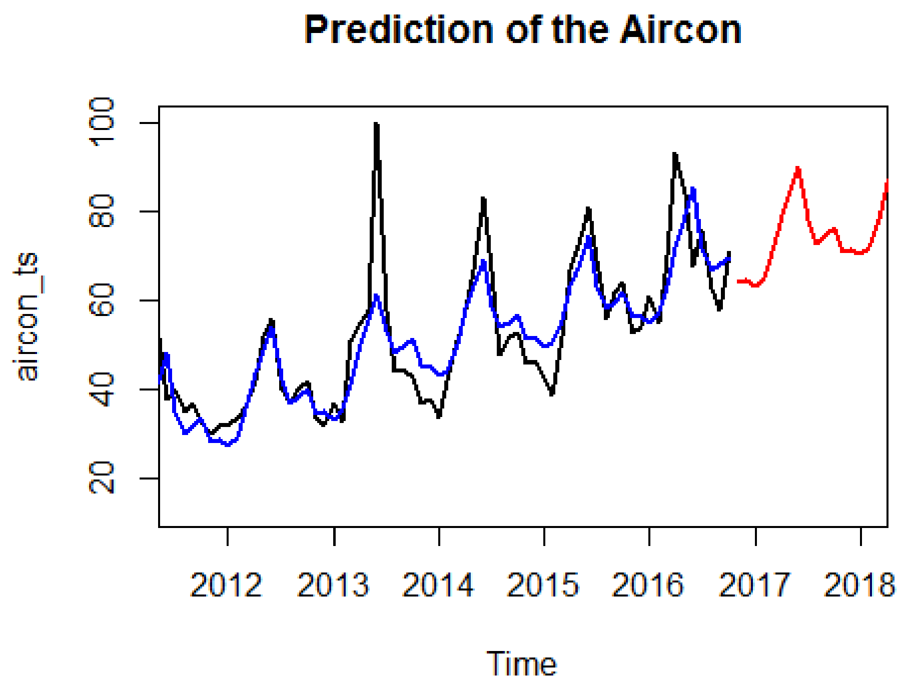

```r
fit_aircon<-stl(aircon_ts, s.window = "periodic", robust=T)
plot(fit_aircon, main="The Decomposition of Aircon")
```



As we can see in the graph, the time series has a trend and seasonal pattern. Therefore, we choose the Triple Exponential Smoothing method.

3. Fit and predict the time series:

```
aircon_optim<-hw(aircon_ts, initial = "optimal", seasonal = "additive",
 h=20)
plot(aircon_ts, lwd=2, xlim=c(2011.6,2018), main="Prediction of the Air
con")
lines(fitted(aircon_optim), col="blue", type="l", lwd=2)
lines(aircon_optim$mean, col="red", type="l", lwd=2)
```



Then we find the number in June 2016, and the prediction in June 2017:

```
aircon_ts

##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 2004  13  18  16  44  34  28  25  13  22  25  24  22
## 2005  15  23  18  21  32  31  27  29  30  19  26  24
## 2006  17  22  25  27  24  32  30  28  24  34  21  24
## 2007  24  23  27  31  30  35  26  24  21  28  20  21
## 2008  21  19  19  31  41  31  21  21  20  23  20  21
## 2009  18  19  22  30  31  44  26  28  25  30  21  21
## 2010  18  30  32  34  43  34  30  27  29  34  28  26
## 2011  24  28  29  34  52  38  40  35  37  33  30  32
## 2012  32  34  36  42  52  56  40  37  40  42  34  32
## 2013  37  33  51  55  58 100  58  44  44  43  37  38
## 2014  34  42  51  58  67  83  67  48  52  53  46  46
## 2015  43  39  52  67  74  81  69  56  62  64  53  54
## 2016  61  55  70  93  84  68  76  63  58  71

aircon_optim

##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Nov 2016       64.16361 55.74601 72.58121 51.29000 77.03723
## Dec 2016       64.63915 56.17921 73.09909 51.70079 77.57751
## Jan 2017       63.41915 54.91170 71.92659 50.40813 76.43016
## Feb 2017       65.09782 56.53746 73.65819 52.00587 78.18977
## Mar 2017       71.10588 62.48697 79.72480 57.92440 84.28737
```

```
## Apr 2017       79.28684 70.60355  87.97014 66.00689  92.56679
## May 2017       84.47146 75.71776  93.22516 71.08384  97.85909
## Jun 2017       90.08184 81.25155  98.91213 76.57707 103.58660
## Jul 2017       78.31734 69.40412  87.23057 64.68574  91.94894
## Aug 2017       72.88235 63.87973  81.88498 59.11402  86.65068
## Sep 2017       74.50885 65.41024  83.60747 60.59372  88.42398
## Oct 2017       76.47316 67.27176  85.67457 62.40083  90.54550
## Nov 2017       71.10176 61.79093  80.41258 56.86208  85.34143
## Dec 2017       71.57729 62.15024  81.00435 57.15986  85.99473
## Jan 2018       70.35729 60.80716  79.90743 55.75163  84.96296
## Feb 2018       72.03597 62.35588  81.71605 57.23155  86.84038
## Mar 2018       78.04403 68.22711  87.86096 63.03034  93.05772
## Apr 2018       86.22499 76.26435  96.18563 70.99151 101.45847
## May 2018       91.40961 81.29840 101.52082 75.94585 106.87337
## Jun 2018       97.01998 86.75138 107.28859 81.31551 112.72446
```

So the number of queries in June 2016 is 68, and the prediction in June 2017 is 90.08, with the 95% prediction interval [76.58,103.59].

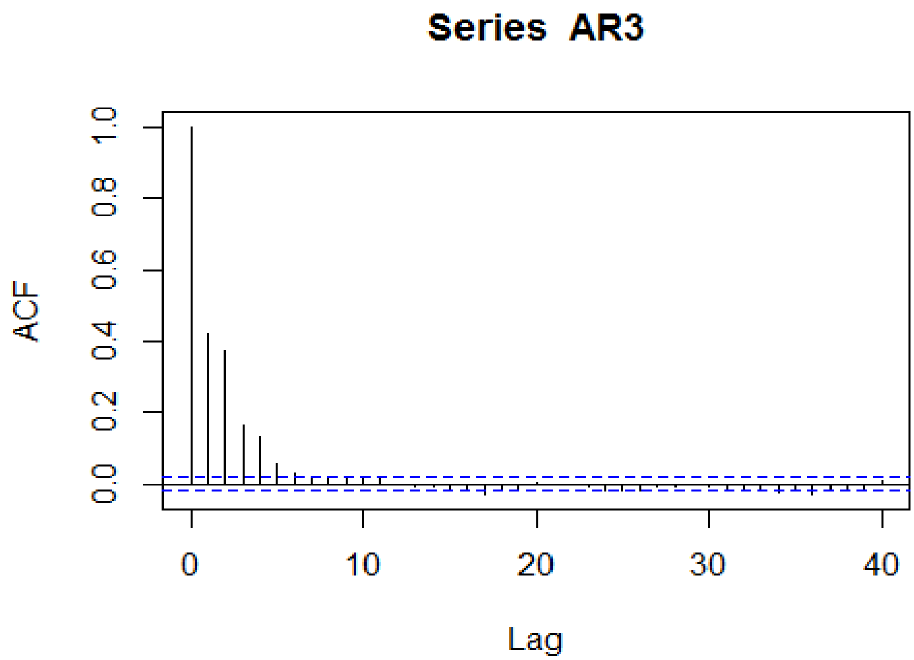
We know that the retailer sold 51 devices in June 2016, so the demand in June 2017 could be 51 / 68 * 90.08 = 67.56, with the 95% prediction interval [57.435,77.69].

Exercise 4

1. Estimate the first three autocorrelation coefficients.

First of all, we load the data and generate the time series.

```
asg_1_AR3 <- read.table("E:/StatSoft/asg_1_MA3.dat", quote="\"", commen
t.char="")
AR3<-ts(asg_1_AR3$V1)
ACFAR3<-acf(AR3)
```




```r
head(ACFAR3$acf)
```

```
## [1] 1.00000000 0.41980461 0.37193302 0.16221226 0.13283237 0.0594732
7
```

So the first three autocorrelation coefficients are 0.4198, 0.3719, 0.1622.

2. The associated Yule-Walker equations:

   $$\rho(k) = \alpha\rho(k-1) + \beta\rho(k-2) + \gamma\rho(k-3)$$

3. Solve the Yule-Walker equations:

   $$\begin{pmatrix} \rho(2) & \rho(1) & \rho(0) \\ \rho(1) & \rho(0) & \rho(1) \\ \rho(0) & \rho(1) & \rho(2) \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} \rho(3) \\ \rho(2) \\ \rho(1) \end{pmatrix}$$

```r
coeffi<-matrix(c(0.3719,0.4198,1,0.4198,1,0.4198,1,0.4198,0.3719),nrow
= 3)
rightside<-matrix(c(0.1622,0.3719,0.4198),nrow=3)
solution<-solve(coeffi)%*%rightside; solution
```

```
##             [,1]
## [1,]  0.33736780
## [2,]  0.26081680
## [3,] -0.07275798
```

Thus, $\hat{\alpha} = 0.34, \hat{\beta} = 0.26, \hat{\gamma} = -0.07$.