PREDICTIVE ANALYTICS

MSC INFORMATION SYSTEMS AND SERVICES

SPECIALIZATION: BIG DATA AND ANALYTICS

R EXERCISES
PART 1

STUDENT

PANAGIOTAKOPOULOS GEORGIOS (ME2030)

SUPERVISOR

FILIPPAKIS MICHAEL

Deadline: 08/07/2021

Table of Contents

Exercise 1 .... **4**
Solution 4
Exercise 2 .... **7**
Solution 7
Exercise 3 .... **10**
Solution 10
3.1) 10
3.2) 11
3.3) 12
3.4) 15
3.5) 17
Exercise 4 .... **18**
Solution 19
4.I) 19
4.II) 20
4.III) 21
Exercise 5 .... **22**
Solution 23
I) 23
II) 23
Exercise 6 .... **24**
Solution 24
I) 24
6.II) 25
6.III) 26
6.IV) 26
6.V) 28
6.VI) 30
6.VII) 31
6.VIII) 31
6.IX) 32

6.X) 33

6.XI) 34

6.XII) 34

R CODE APPENDIX 36

Exercise 1

Pima.te from the MASS package

We want to study the factors that cause diabetes. Create the appropriate forecast model and analyze each direction, which variables are important? interpret the results.
Predict the results for the next 4 people.

	Npreg	Glu	BP	BMI	SKIN	PED	age
1	5	140	76	70	20	0.6	40
2	1	80	70	25	45	0.56	25
3	8	120	60	27	30	0.5	44
4	2	91	50	68	23	.7	34

Solution

We divide the data into training & test sets, adjust all the variables (model.all) and then remove the insignificant variables one by one and end up with the final model (model4):

Y = -9.02 + 0.195*npreg + 0.034*glu + 0.079*bmi + 0.103*ped

We partially removed the variables: **bp, age & skin.**

In the final model all variables are important with positive coefficients. That is, when the important variables increase by a few points then the probability of a woman getting diabetes increases, for example, the number of pregnancies, the plasma glucose concentration in the oral glucose tolerance test, the body mass index, and the function of genealogic diabetes.

The forecasts for the 4 new people are as follows:

```
         1          2          3          4
0.94989038 0.02938887 0.33618511 0.65521555
```

The above results analyzed are displayed in the following code:

The data file contains the following columns:

Npreg : number of pregnancies.
Glu: plasma glucose concentration in an oral glucose tolerance test.
Bp: diastolic blood pressure (mm Hg).
Skin: triceps skin fold thickness (mm).

```
Bmi: body mass index (weight in kg/(height in m)\^2).
Ped: diabetes pedigree function.
Age: age in years.
```

Through the MASS library we read the Pima.te data file.

```
library(MASS) #pima.te - diabetes dataset is in MASS library
data <- Pima.te
set.seed(789)
training = sample(nrow(data),265,replace = FALSE)
train = data[training, ]
test = data[-training, ]
model.all <- glm(type ~.,data = train,family = "binomial")
summary(model.all)
Call:
glm(formula = type ~ ., family = "binomial", data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7842  -0.6641  -0.3474   0.5964   2.4884

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -9.565870   1.356856  -7.050 1.79e-12 ***
npreg        0.149933   0.062464   2.400   0.0164 *
glu          0.033422   0.006139   5.444 5.22e-08 ***
bp           0.003671   0.015216   0.241   0.8094
skin         0.028826   0.022630   1.274   0.2027
bmi          0.054201   0.032263   1.680   0.0930 .
ped          1.015078   0.500696   2.027   0.0426 *
age          0.016210   0.019607   0.827   0.4084
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 329.47  on 264  degrees of freedom
Residual deviance: 227.71  on 257  degrees of freedom
AIC: 243.71

Number of Fisher Scoring iterations: 5



model2 <- glm(type ~ npreg+glu+skin+bmi+ped+age, data = train,family
= "binomial")
summary(model2)
Call:
glm(formula = type ~ npreg + glu + skin + bmi + ped + age, family =
"binomial",
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7916  -0.6580  -0.3505   0.6162   2.4645
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)    
(Intercept) -9.438374   1.246204  -7.574 3.63e-14 ***
npreg        0.149450   0.062507   2.391   0.0168 *  
glu          0.033531   0.006129   5.471 4.49e-08 ***
skin         0.028329   0.022505   1.259   0.2081    
bmi          0.057160   0.029803   1.918   0.0551 .  
ped          1.014956   0.500687   2.027   0.0426 *  
age          0.017564   0.018835   0.933   0.3511    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 329.47  on 264  degrees of freedom
Residual deviance: 227.77  on 258  degrees of freedom
AIC: 241.77

Number of Fisher Scoring iterations: 5


model3 <- glm(type ~ npreg+glu+skin+bmi+ped, data = train,family =
"binomial")
summary(model3)
Call:
glm(formula = type ~ npreg + glu + skin + bmi + ped, family =
"binomial",
    data = train)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-2.8659  -0.6483  -0.3524   0.6249   2.4505  

Coefficients:
             Estimate Std. Error z value Pr(>|z|)    
(Intercept) -9.101224   1.177409  -7.730 1.08e-14 ***
npreg        0.186867   0.048684   3.838 0.000124 ***
glu          0.034478   0.006073   5.677 1.37e-08 ***
skin         0.027342   0.022418   1.220 0.222599    
bmi          0.056719   0.029804   1.903 0.057034 .  
ped          1.075005   0.496387   2.166 0.030337 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 329.47  on 264  degrees of freedom
Residual deviance: 228.63  on 259  degrees of freedom
AIC: 240.63

Number of Fisher Scoring iterations: 5


model4 <- glm(type ~ npreg+glu+bmi+ped, data = train,family =
"binomial")
```

```
summary(model4)
Call:
glm(formula = type ~ npreg + glu + bmi + ped, family = "binomial",
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.8168  -0.6698  -0.3836   0.5957   2.4859

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -9.017650   1.168199  -7.719 1.17e-14 ***
npreg        0.194749   0.048470   4.018 5.87e-05 ***
glu          0.034025   0.006022   5.651 1.60e-08 ***
bmi          0.079440   0.023930   3.320 0.000901 ***
ped          1.102782   0.494459   2.230 0.025729 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 329.47  on 264  degrees of freedom
Residual deviance: 230.13  on 260  degrees of freedom
AIC: 240.13

Number of Fisher Scoring iterations: 5

newdata = read.csv(''meros1_exc1_new_data.csv')
pred_new <- predict(model4, newdata = newdata, type = "response")
pred_new
         1          2          3          4
0.94989038 0.02938887 0.33618511 0.65521555
```

Exercise 2

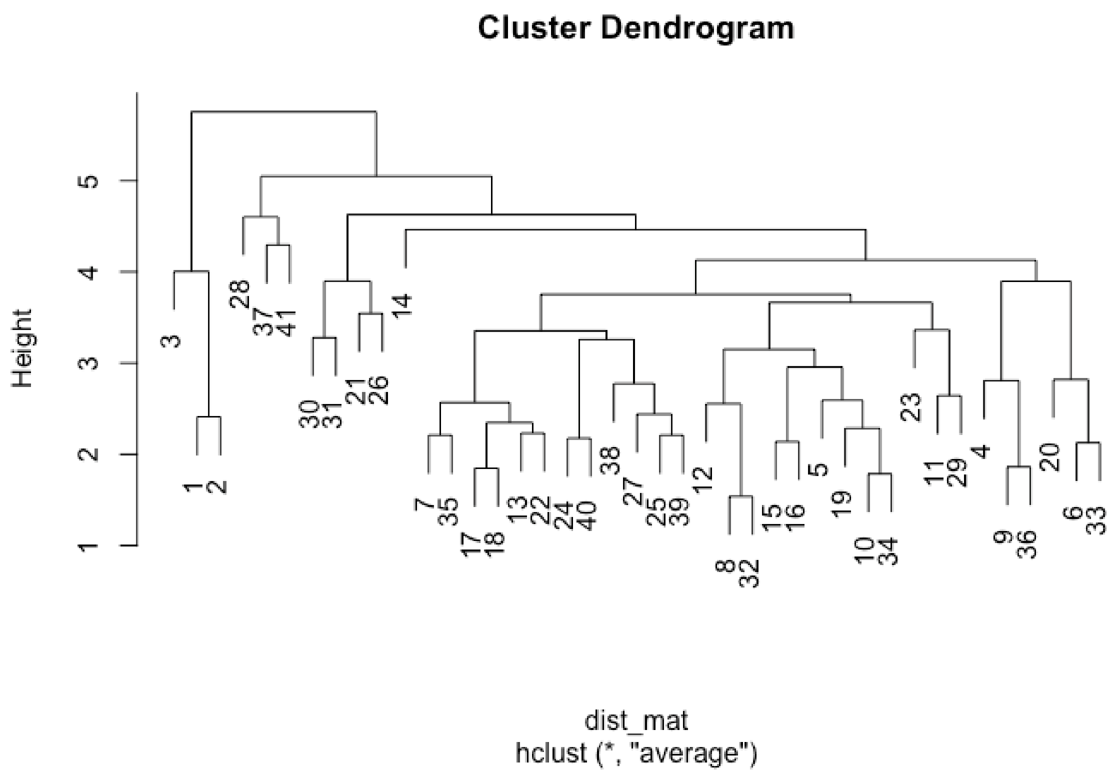
We consider the data "decathlon". The data refer to the performance of decathlon games at the Athens Olympic Games (23 to 24 August 2004). On the first day the athletes compete in 5 events (100m, long jump, shot put, high jump, 400m) and in the other five on the second day in (110m hurdles, discus, pole vault, javelin, 1500m). Build a dendrogram that shows us pictures of their performance. You will standardize the data where you deem necessary and pruning the dendrogram. Finally, characterize the resulting clusters. Also, do clustering with the kmeans method

Solution

First, we standardize the data. We present below dendrogram that shows the performance images. Then we do grouping with the kmeans method which divides the athletes into 7 groups based on the performance in the sports.

The sizes of the 7 clusters are 5, 6, 7, 9, 8, 3, 3,
and the 41 athletes belong in order of appearance to the following clusters:
7 7 7 5 4 5 5 4 5 4 2 4 5 5 4 4 1 1 4 6 2 1 3 1 3 2 3 6 2 2 2 4 6 4 5 5 3 3 3 1 3.

Το δενδρόγραμμα το κλαδεύουμε ώστε να γίνουν οι συστάδες 7 με την εντολή prune(hclust_avg).



K-means clustering with 7 clusters of sizes 5, 6, 7, 9, 8, 3, 3

```
Cluster means:
       X100m   Long.jump    Shot.put   High.jump       X400m
X110m.H      Discus   Pole.vault
1  0.3571977 -0.20859563 -1.0881161 -0.66137070 -0.3765583
0.6785795 -0.8844722 -0.78575238
2 -0.2777277  0.02107027  0.4159574  0.09185704  0.1967937 -
0.1819747  0.6787534  0.77060352
3  1.2783556 -0.68628865 -0.7554178 -0.59070541  0.9568319
0.6749459 -0.3772172 -0.02932847
4 -0.6135666  0.62157284 -0.2336514 -0.57642960 -0.4572822 -
0.3350563 -0.7894741  0.87052364
5 -0.2729753 -0.15407632  0.3128556  0.83665313 -0.3782922 -
0.5264083  0.4720141 -0.97100430
6  1.0719641 -1.47491859  0.9577473  0.52280824  1.6793017
1.2452169  0.3792527 -0.44042842
```

```
7 -1.5260343  1.92792930  1.6531791  1.27228858 -1.2972738 -
1.1781827  1.7272523  0.25501566
    Javeline     X1500m
1  0.2733507 -1.0246402
2 -0.6605284  1.6310904
3 -0.4840838  0.2215793
4 -0.2550395 -0.2410441
5  0.1610097 -0.6815694
6  0.8929442  0.3822234
7  1.4378164  0.0869614

Clustering vector:
 [1] 7 7 7 5 4 5 5 4 5 4 2 4 5 5 4 4 1 1 4 6 2 1 3 1 3 2 3 6 2 2 2 4
6 4 5 5 3 3 3 1 3

Within cluster sum of squares by cluster:
[1] 14.71159 33.30038 34.05495 34.33803 35.02823 13.86263 12.62547
 (between_SS / total_SS =  55.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
```

Exercise 3

library(astsa)
library(TTR)

1. Let x_t be the time series (cmort). Adjust the data to an AR (2) model using linear regression. Then with the custom model you found make a 4-week forecast and find a 95% confidence interval for the forecast.
2. Generate n = 500 observations from an ARMA model with the formula:

$$X_t = 0.9x_{t-1} + w_t - 0.9w_{t-1},\ w_t \sim iidN(0,1)$$

Graph the simulated time series and calculate ACF, PACF and customize your data with an ARMA (1,1). What do you notice?

3. Consider the **oil** data. Adjust the data with an ARIMA (p, d, q) suitable model and perform the diagnostic test of the model as well as its evaluation.
4. Consider the data **globtemp**. Customize data with an ARIMA (p, d, q) suitable model and do the model diagnostics as well as evaluation of this. Then make a forecast for the next 10 years.
5. Consider the chicken data. Adjust these to a suitable ARIMA model and make a forecast for the next 12 months.

Solution

3.1)

We customize an AR model (2) and build 95% confidence interval for the next 4 weeks are given as follows:

```
76.5     98.8
74.7     99.0
73.5    101.5
72.5    102.3
```

The R commands are given below:

```
# Loading the TTR library.
> library(TTR)
> ts = cmort
> model_ts <- arima(ts,c(2,0,0))
> ts_forecast <- predict(model_ts, n.ahead = 4)
> msft_forecast_values <- ts_forecast$pred
> msft_forecast_se <- ts_forecast$se
> lower_bound = msft_forecast_values - 1.96*msft_forecast_se
> upper_bound = msft_forecast_values + 1.96*msft_forecast_se
> confidence_int = cbind(lower_bound, upper_bound)
> msft_forecast_values
Time Series:
Start = c(1979, 41)
End = c(1979, 44)
Frequency = 52
```

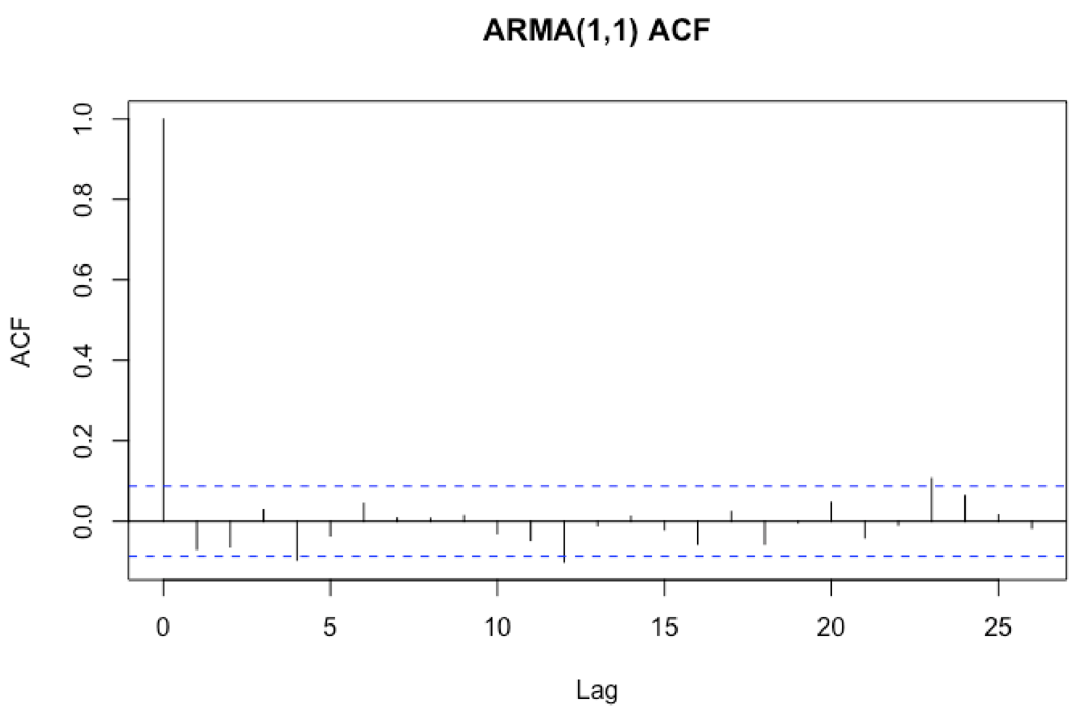
```
[1] 87.66207 86.85311 87.46615 87.37190
> confidence_int
Time Series:
Start = c(1979, 41)
End = c(1979, 44)
Frequency = 52
         lower_bound upper_bound
1979.769    76.51057    98.81358
1979.788    74.71407    98.99214
1979.808    73.45539   101.47690
1979.827    72.45134   102.29246
```

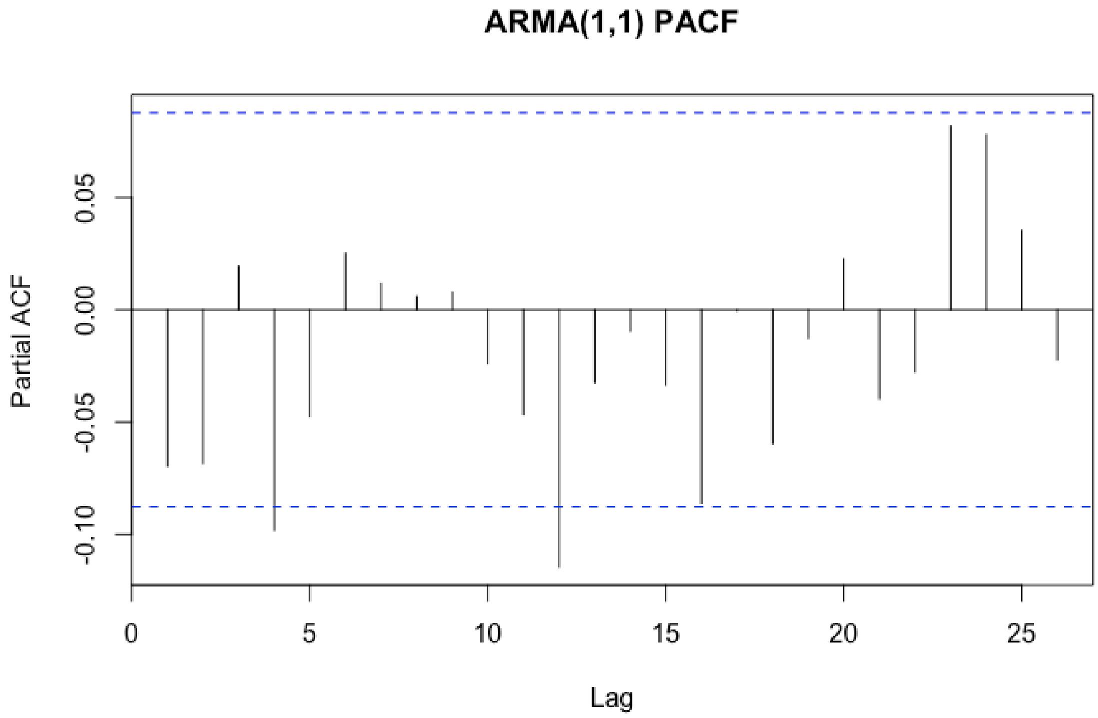
3.2)

We create the data, n = 500 observations from an ARMA model with the formula:

$$X_t = 0.9x_{t-1} + w_t - 0.9w_{t-1},\ w_t \sim \text{iidN}(0,1)$$

and calculate ACF and PACF. From ACF we see that the time series is 1st class AR (1) and / or MA (1). From the PACF we observe that classes 4 and 12 of the partial correlations are important but this does not make sense since the time series is ARMA (1,1) and occurs due to statistical error.






```
ts = arima.sim( n = 500, list(ar = 0.9, ma = -0.9, sd = 1) )
acf(ts, main = 'ARMA(1,1) ACF')
pacf(ts, main = 'ARMA(1,1) PACF')
model_ts <- arima(ts,c(1,0,1))
> model_ts$coef
        ar1         ma1   intercept
 0.93712095 -0.99999201  0.07408236
```

3.3)

To the oil data we adapt ARIMA models (p, d, q) and with the command auto.arima we select the model **ARIMA(1,1,3)**.

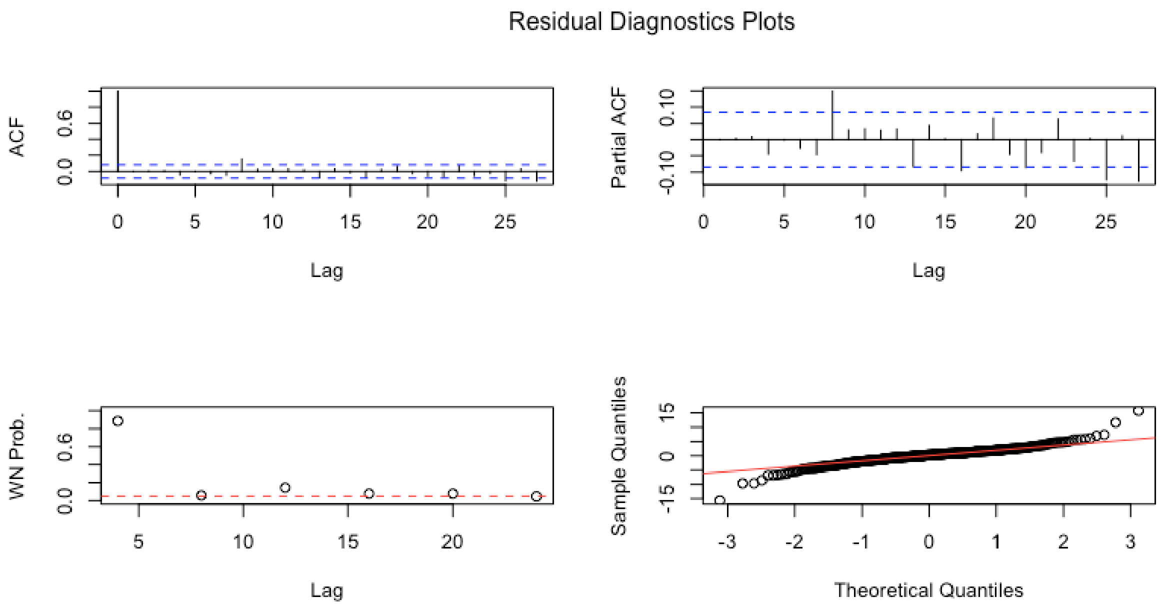
We use AIC and BIC as evaluation criteria in selecting the most suitable model. The smaller these criteria are, the more appropriate is the model (AIC=2560.66   AICc=2560.82   BIC=2586.46).

We perform a diagnostic test with the command**ts.diag(estimate(ts,p=1,d=1,q=3))**
and we notice that the errors are not correlated with the selected model

```
Autocorrelation Check of Residuals
     lag    LB p.value
[1,]   4  1.16  0.8846 > 0.05
```

```
> ts = oil
> fit = auto.arima(ts)
> fit
Series: ts
ARIMA(1,1,3)(0,0,1)[52]


Coefficients:
         ar1     ma1      ma2    ma3     sma1
      0.8793  -0.725  -0.1178  0.066  -0.0738
s.e.  0.0539   0.069   0.0552  0.044   0.0436

sigma^2 estimated as 6.396:  log likelihood=-1274.33
AIC=2560.66   AICc=2560.82   BIC=2586.46
```

```
#Load the atsa library.
> library(atsa)
> ts.diag(estimate(ts,p=1,d=1,q=3))
ARIMA(1,1,3) model is estimated for variable: ts

Conditional-Sum-of-Squares & Maximum Likelihood Estimation
      Estimate    S.E t.value  p.value Lag
AR 1     0.061 0.0445    1.37 1.71e-01   3
MA 1     0.884 0.0521   16.98 0.00e+00   1
MA 2    -0.726 0.0677  -10.73 1.69e-24   1
MA 3    -0.119 0.0556   -2.13 3.34e-02   2
-----
n = 545; 'sigma' = 2.524602; AIC = 2561.507; SBC = 2578.71
------------------------------
Correlation of Parameter Estimates
         AR 1    MA 1   MA 2    MA 3
AR 1   1.0000 -0.7714 -0.146 -0.0977
MA 1  -0.7714  1.0000 -0.249  0.0419
MA 2  -0.1460 -0.2491  1.000 -0.6164
MA 3  -0.0977  0.0419 -0.616  1.0000
------------------------------
Autocorrelation Check of Residuals
     lag    LB p.value
[1,]   4  1.16  0.8846
[2,]   8 15.07  0.0578
[3,]  12 17.15  0.1441
[4,]  16 24.58  0.0776
[5,]  20 29.55  0.0775
[6,]  24 36.71  0.0468
------------------------------
Model for variable: ts
Period(s) of Differencing: ts(1,0)

AR factors: 1 + 0.8844 B**(1)
MA factors: 1 - 0.7262 B**(1) - 0.1186 B**(2) + 0.061 B**(3)
```


Residual Diagnostics Plots
ACF
0.0
0.6
0
5
10
15
20
25
Lag
Partial ACF
-0.10
0.10
0
5
10
15
20
25
Lag
WN Prob.
0.0
0.6
5
10
15
20
Lag
Sample Quantiles
-15
0
15
-3
-2
-1
0
1
2
3
Theoretical Quantiles

3.4)

To the **globtemp** data we adapt ARIMA models (p, d, q) and with the auto.arima command we select the model **ARIMA(1,1,3)**.

We use AIC and BIC as evaluation criteria in selecting the most suitable model. The smaller these criteria are, the more appropriate is the model (AIC=-234.12   AICc=-233.47   BIC=-216.69).

We perform a diagnostic test with the following command:
ts.diag(estimate(ts,p=1,d=1,q=3))
and we notice that the errors are not correlated with the selected model
Autocorrelation Check of Residuals

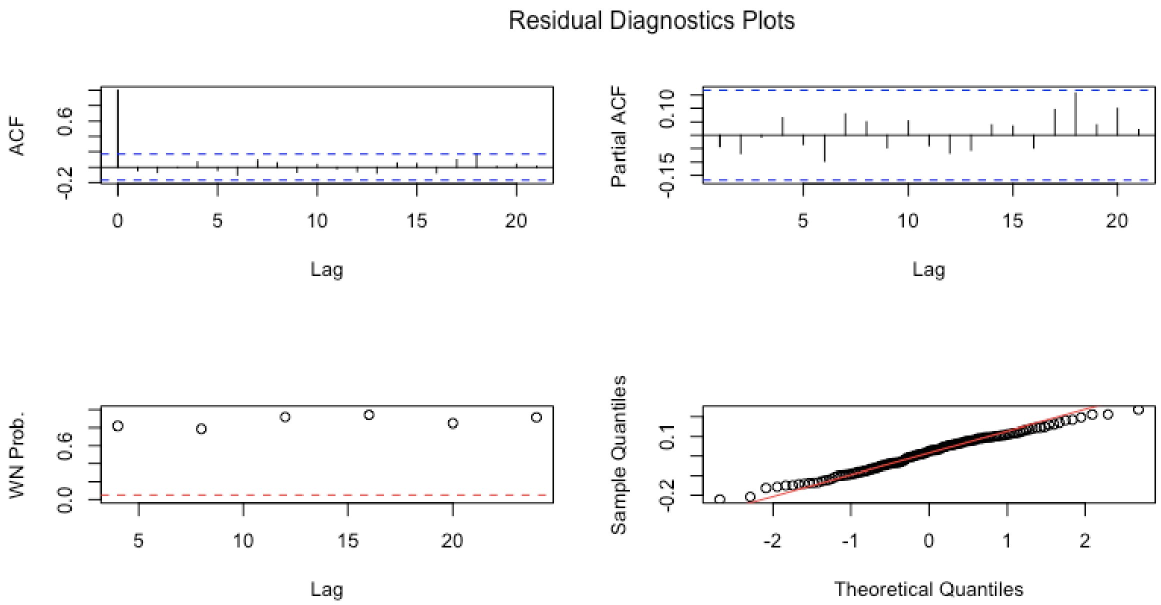
```
     lag    LB p.value
[1,]   4  1.54   0.819
```

The forecasts for the next 10 years are:

[1] 0.7941763 0.7597491 0.7511044 0.7592924 0.7515370 0.7588827 0.7519251 0.7585151 0.7522732
[10] 0.7581853

```
> ts = globtemp
> fit = auto.arima(ts)
> fit
Series: ts
ARIMA(1,1,3) with drift

Coefficients:
          ar1     ma1      ma2      ma3   drift
      -0.9449  0.6081  -0.5680  -0.3091  0.0072
s.e.   0.0562  0.0971   0.0856   0.0804  0.0032

sigma^2 estimated as 0.009775:  log likelihood=123.06
AIC=-234.12   AICc=-233.47   BIC=-216.69



> ts.diag(estimate(ts,p=1,d=1,q=3))
ARIMA(1,1,3) model is estimated for variable: ts

Conditional-Sum-of-Squares & Maximum Likelihood Estimation
      Estimate    S.E t.value  p.value Lag
AR 1    -0.278 0.0792   -3.51 6.19e-04   3
MA 1    -0.947 0.0539  -17.59 2.11e-36   1
MA 2     0.645 0.0967    6.67 6.23e-10   1
MA 3    -0.506 0.0843   -5.99 1.83e-08   2
-----
n = 136; 'sigma' = 0.09856878; AIC = -231.9642; SBC = -220.3136
------------------------------
Correlation of Parameter Estimates
```

```
       AR 1   MA 1  MA 2   MA 3
AR 1   1.000 -0.500 0.257 -0.025
MA 1  -0.500  1.000 0.282 -0.383
MA 2   0.257  0.282 1.000  0.413
MA 3  -0.025 -0.383 0.413  1.000
------------------------------
Autocorrelation Check of Residuals
     lag    LB p.value
[1,]   4  1.54   0.819
[2,]   8  4.73   0.786
[3,]  12  5.98   0.917
[4,]  16  8.24   0.942
[5,]  20 13.63   0.849
[6,]  24 15.25   0.913
------------------------------
Model for variable: ts
Period(s) of Differencing: ts(1,0)

AR factors: 1 - 0.9472 B**(1)
MA factors: 1 + 0.6454 B**(1) - 0.5055 B**(2) - 0.2777 B**(3)
```




```
> ts_forecast <- predict(model_ts, n.ahead = 10)
> ts_forecast$pred
Time Series:
Start = 2016
End = 2025
Frequency = 1
 [1] 0.7941763 0.7597491 0.7511044 0.7592924 0.7515370 0.7588827
0.7519251 0.7585151 0.7522732
[10] 0.7581853
```

3.5)

In the **chicken** data file we adapt ARIMA models (p, d, q) and with the auto.arima command we select the model **ARIMA(2,1,1)**.

We use AIC and BIC as evaluation criteria in selecting the most suitable model. The smaller these criteria are, the more appropriate is the model (AIC=351.01 AICc=351.5 BIC=370.14).

Then we perform a diagnostic test with the command:
ts.diag(estimate(ts,p=2,d=1,q=1))
and we notice that the errors are not correlated with the selected model

```
Autocorrelation Check of Residuals
     lag    LB  p.value
[1,]   4  2.84 0.584126
```

The forecasts for the next 12 months are:

```
Jan      Feb      Mar      Apr      May      Jun      Jul
111.1112 110.8680 110.7415 110.7072 110.7290 110.7738 110.8189
Aug      Sep      Oct      Nov      Dec
110.8526 110.8717 110.8785 110.8772 110.8721
```

```
> ts = chicken
> fit = auto.arima(ts)
> fit
Series: ts
ARIMA(2,1,1)(0,0,1)[12] with drift

Coefficients:
         ar1      ar2      ma1    sma1   drift
      1.2933  -0.5375  -0.4019  0.2756  0.2518
s.e.  0.2220   0.1542   0.2569  0.0692  0.1428

sigma^2 estimated as 0.396:  log likelihood=-169.51
AIC=351.01   AICc=351.5   BIC=370.14
```

```
> library(aTSA)
> ts.diag(estimate(ts,p=2,d=1,q=1))
ARIMA(2,1,1) model is estimated for variable: ts

Conditional-Sum-of-Squares & Maximum Likelihood Estimation
      Estimate   S.E t.value  p.value Lag
AR 1    -0.321 0.213   -1.50 1.34e-01   1
AR 2     1.255 0.189    6.64 3.76e-10   1
MA 1    -0.512 0.139   -3.69 2.99e-04   2
-----
n = 180; 'sigma' = 0.6568873; AIC = 366.5547; SBC = 376.1335
------------------------------
Correlation of Parameter Estimates
```

```
        AR 1   AR 2   MA 1
AR 1   1.000 -0.964 -0.939
AR 2  -0.964  1.000  0.884
MA 1  -0.939  0.884  1.000
------------------------------
Autocorrelation Check of Residuals
     lag    LB  p.value
[1,]   4  2.84 0.584126
[2,]   8 12.35 0.136106
[3,]  12 32.79 0.001043
[4,]  16 35.06 0.003906
[5,]  20 44.17 0.001429
[6,]  24 51.32 0.000959
------------------------------
Model for variable: ts
Period(s) of Differencing: ts(1,0)

AR factors: 1 + 1.2546 B**(1) - 0.5118 B**(2)
MA factors: 1 - 0.321 B**(1)
```

```
> ts_forecast <- predict(model_ts, n.ahead = 12)
> ts_forecast$pred
          Jan      Feb      Mar      Apr      May      Jun      Jul
Aug      Sep      Oct      Nov
2016
111.1112 110.8680 110.7415 110.7072
2017 110.7738 110.8189 110.8526 110.8717 110.8785 110.8772 110.8721
          Dec
2016 110.7290
2017
```

Exercise 4

The data in the table show the sales of beer of an industry (in millions of bottles) per region from 2016 to 2019.

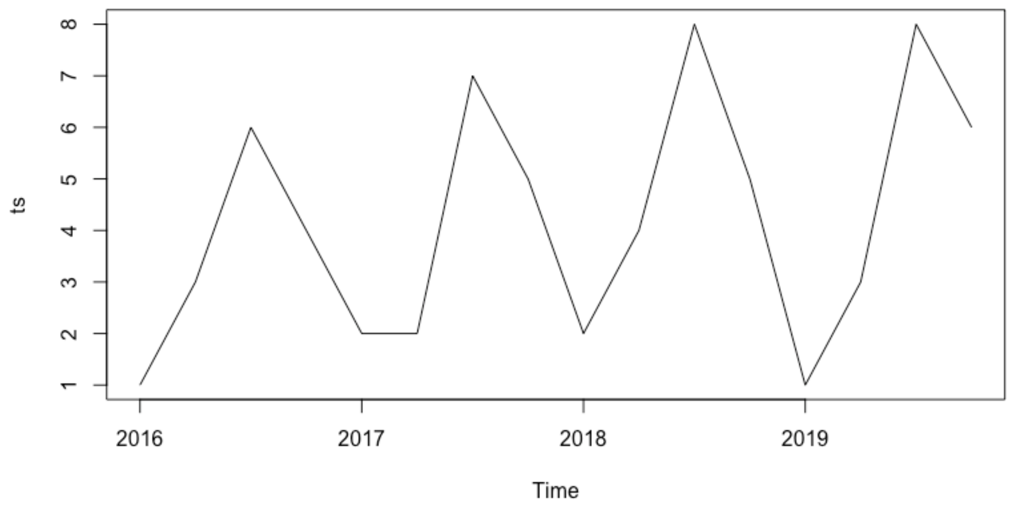
i. Eliminate seasonality and calculate the seasonal indexes of this data.
ii. Build the trend line.
iii. Determine the cyclic change by the method of the relative cyclic residues.

Year	Winter	Spring	Summer	Autumn
2016	1	3	6	4
2017	2	2	7	5
2018	2	4	8	5
2019	1	3	8	6

Solution

To show the time series of beer sales, we apply the following graph:

```
dat = c(1,3,6,4,
       2,2,7,5,
       2,4,8,5,
       1,3,8,6)
ts <- ts(dat, start=c(2016, 1), end=c(2019, 4), frequency=4)
plot.ts(ts)
```



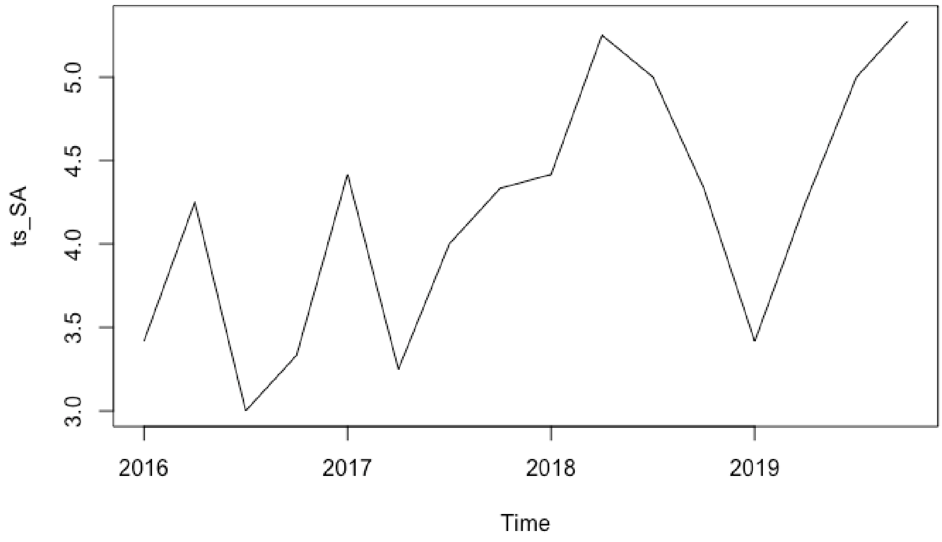
4.I)

Elimination of seasonality can be done by taking the differences between one quarter and the corresponding quarter of the previous year.

In the figure below we show the graph of the time series after we eliminate the seasonality.

We calculate the seasonal indexes with the command

```
> ts_comp$figure # Seasonality Indexes
[1] -2.4166667 -1.2500000  3.0000000  0.6666667
```

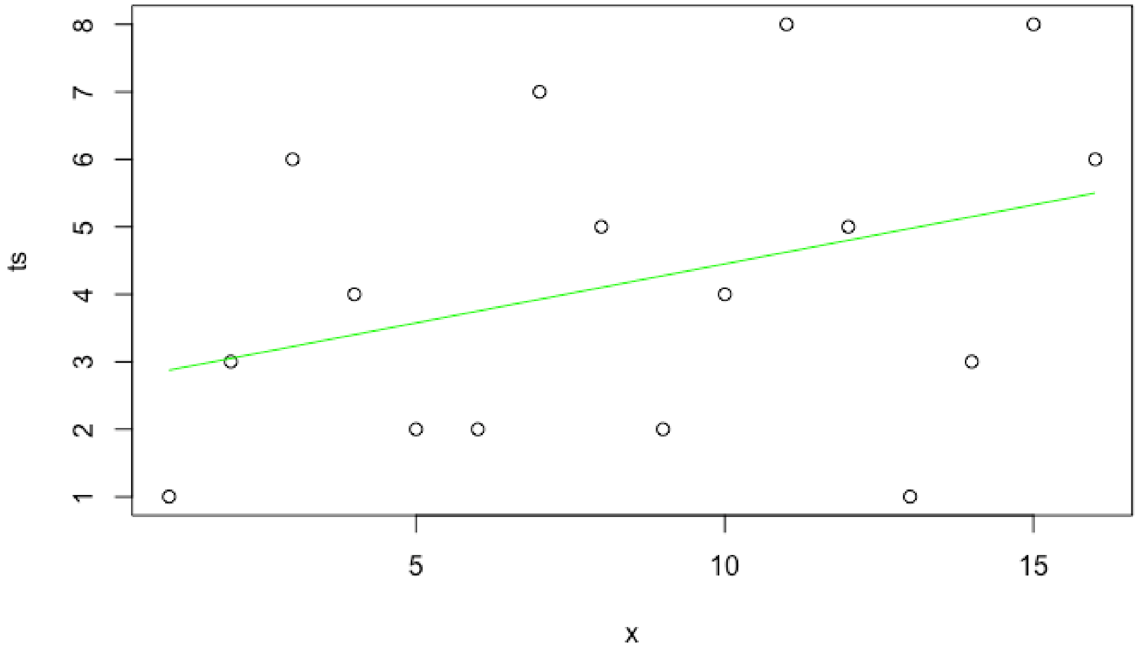



```
> library(fpp)
> ts_comp <- decompose(ts)
> ts_SA <- ts - ts_comp$seasonal
> plot.ts(ts_SA)
> ts_comp$figure # Seasonality Indexes
[1] -2.4166667 -1.2500000  3.0000000  0.6666667
```

4.II)

The trend line is shown in the figure below.

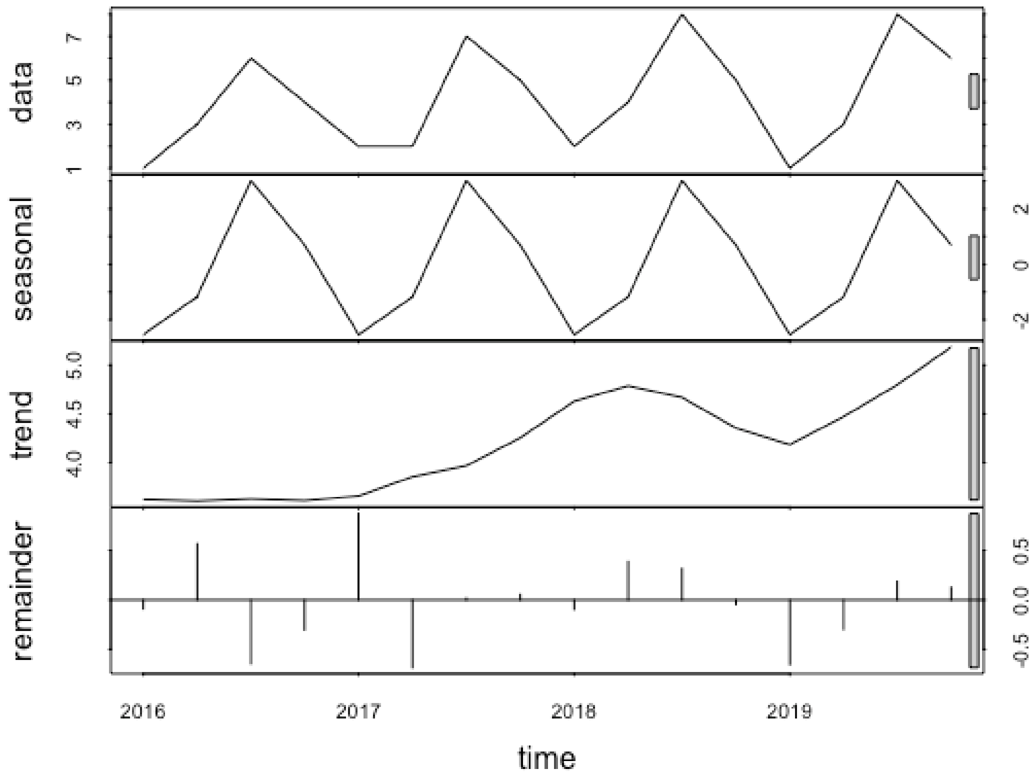
```
# 4.II
x <- (1:length(ts))
plot(ts)
plot(x,ts)
lines(predict(lm(ts~x)),col='green')
```



4.III)

We decompose the time series and observe the cyclical change from the trend of the series. We also see the residues if they show some cyclicity.

```
decomp <- stl(ts, s.window="periodic")
plot(decomp)
```



Exercise 5

The following table shows the receipts from the annual sales (in thousand euros) of an industry:

Year	Revenues
2011	37.44
2012	44.14
2013	46.25
2014	43.99
2015	51.84
2016	49.10
2017	58.56
2018	58.02
2019	70.28

i. Eliminate seasonality and calculate the seasonal indexes of this data.
ii. Build the trend line.
iii. Determine the cyclic change by the method of the relative cyclic residues

Solution

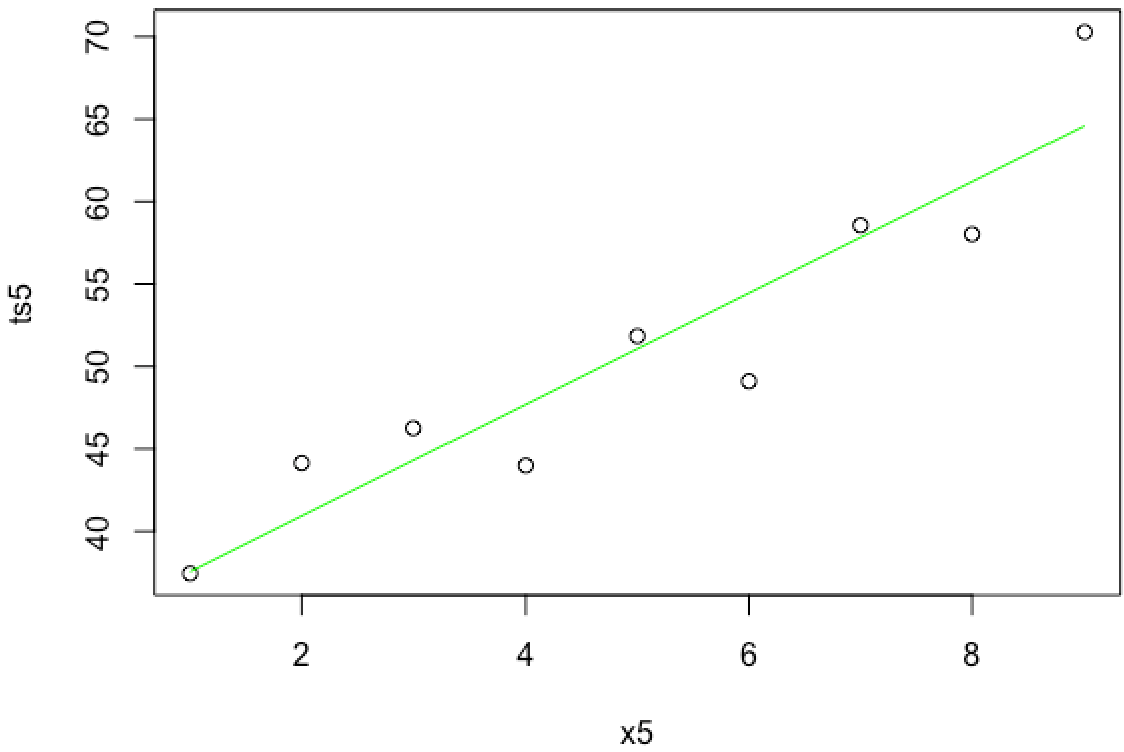
I)

In the present time series we have annual data, so it does not make sense to look for seasonality.

II)

The trend line is shown in the figure below.

```
dat5 = c(37.44, 44.14, 46.25, 43.99, 51.84, 49.10, 58.56, 58.02,
70.28)
ts5 <- ts(dat5, start=c(2011), end=c(2019), frequency=1)
plot.ts(ts5)
x5 <- (1:length(ts5))
ts.plot(ts5)
plot(x5,ts5)
lines(predict(lm(ts5~x5)),col='green')
```



Exercise 6

https://www.bankofcanada.ca/

the daily USD/CAD exchange rates for the last 2 years

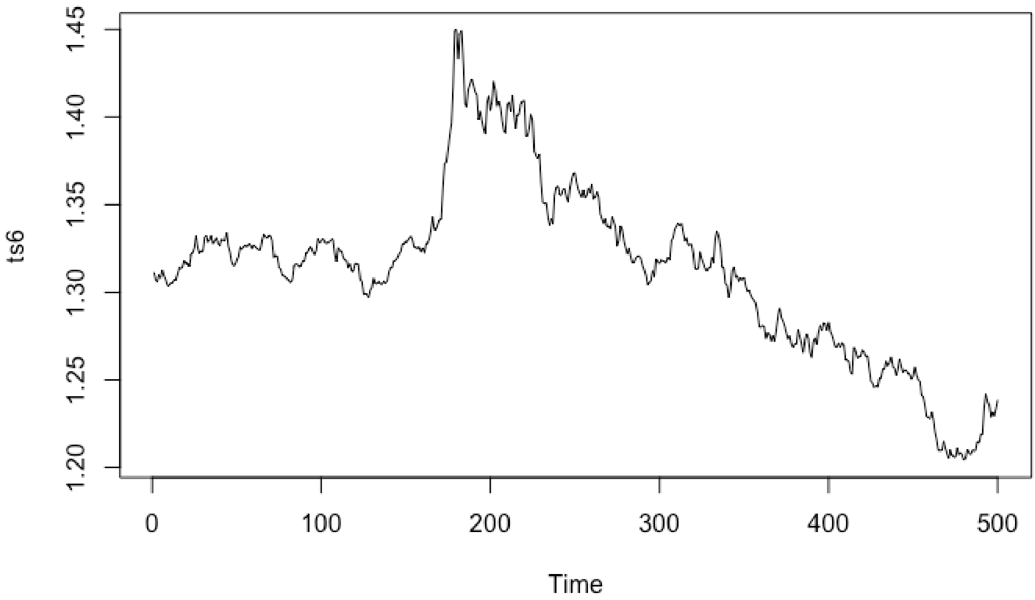
i. Plot the time series.
ii. Find the trend line with linear regression and design the chart.
iii. Find the quadratic trend and do the chart again.
iv. Calculate the first and second differences and make the corresponding charts.
v. Make a prediction with one of the above 4 models for the next 22 days.
vi. Add a seasonal component (i.e. the January effect) to the linear trend (create a binary variable that will show if the exchange rate is from January or not). Fit the model with the effect of January.
(hint: use the format function to convert the dates to strings)
vii. Fit a quadratic trend considering the effect of January.
viii. Scatter plot the exchange rates and the lagged values.
ix. Calculate the self-correlation r_1 lag 1. What do you notice?
x. Make a function (name it cal AC (y, k) that will compute the k-lag autocorrelation in general for a time series.
xi. In the above data make the AR (1) and MA (1) model
AR(1)=ARIMA(1,0,0)
MA(1)=ARIMA(0,0,1)
xii. Compare the above two models with ARIMA (2,1,2)
Calculate the MAE (MEAN ABSOLUTE ERROR), mse, mape.

Solution

```
df6 = read.csv('FXUSDCAD.csv')
library(xts)
ts6 <- xts(df6[,-1], order.by=as.Date(df6[,1], "%d/%m/%Y"))
```

I)

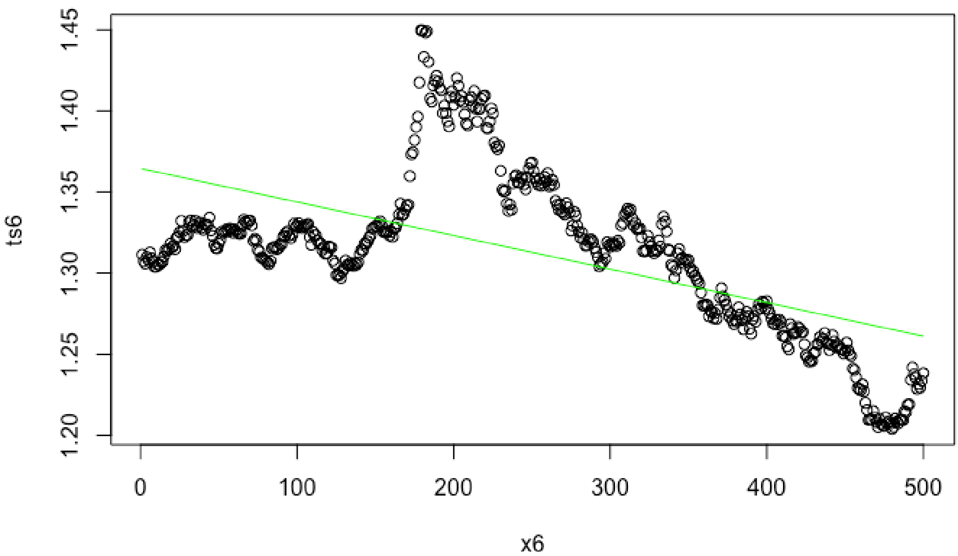
```
Time series graph

plot.ts(ts6)
```



6.II)

We represent in a diagram the line trend of the time series:

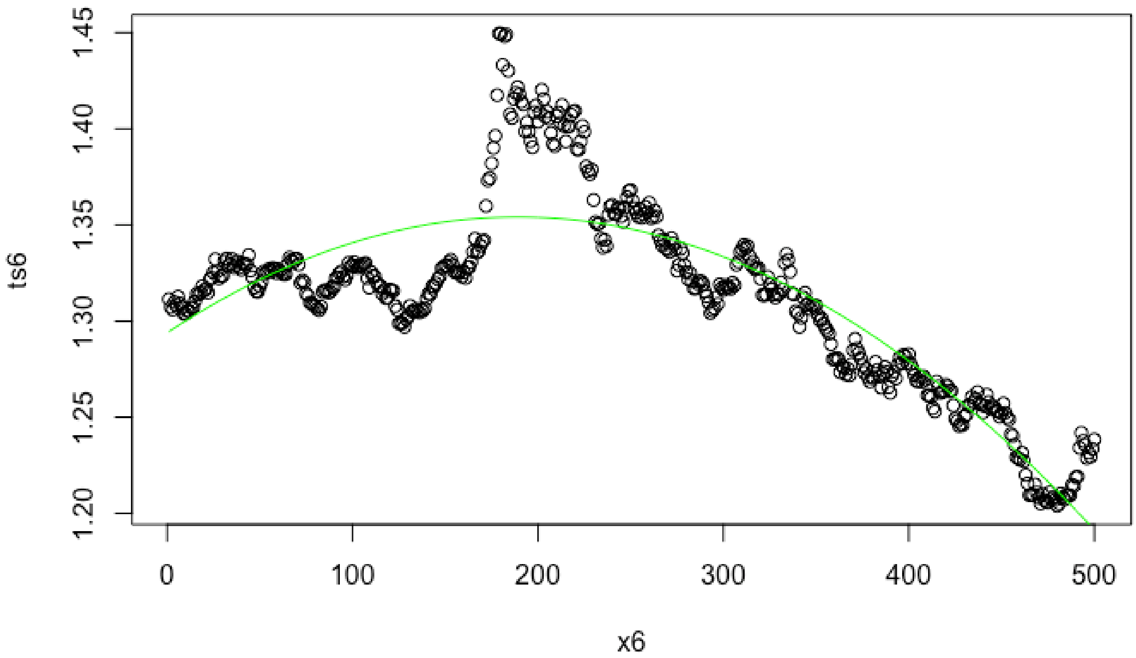
```
x6 <- (1:length(ts6))
plot.ts(ts6)
plot(x6,ts6)
lines(predict(lm(ts6~x6)),col='green')
```



6.III)

We represent graphically with the following code the square trend of the time series

```
plot.ts(ts6)
plot(x6,ts6)
lines( predict( lm( ts6 ~ x6 + I(x6^2) ) ), col='green' )
```

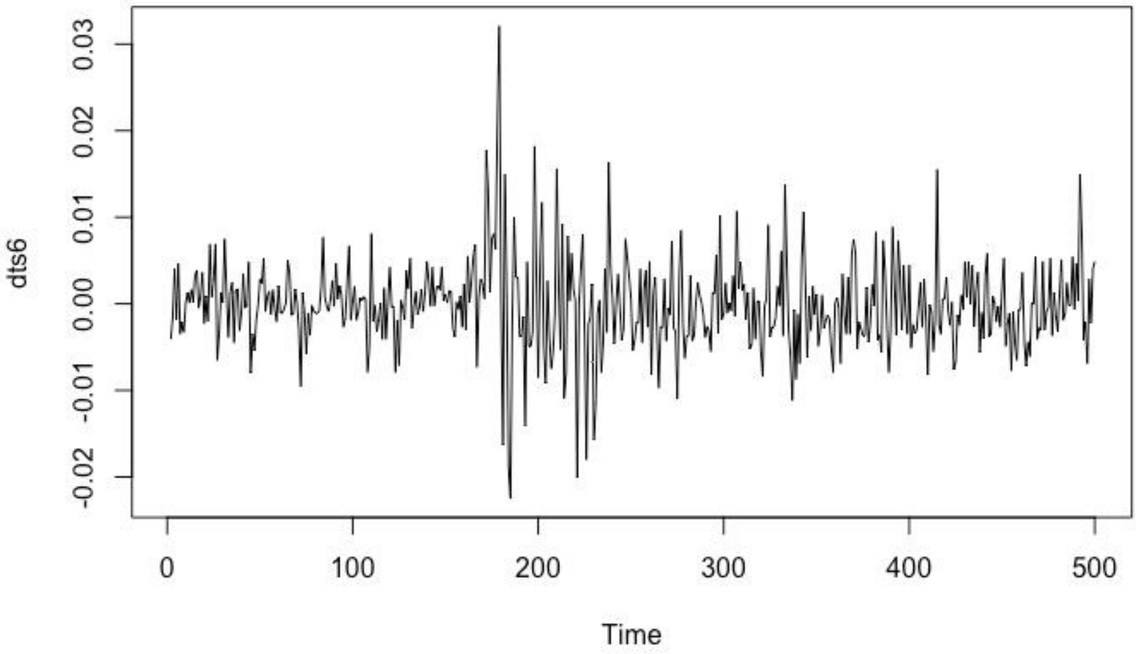


6.IV)

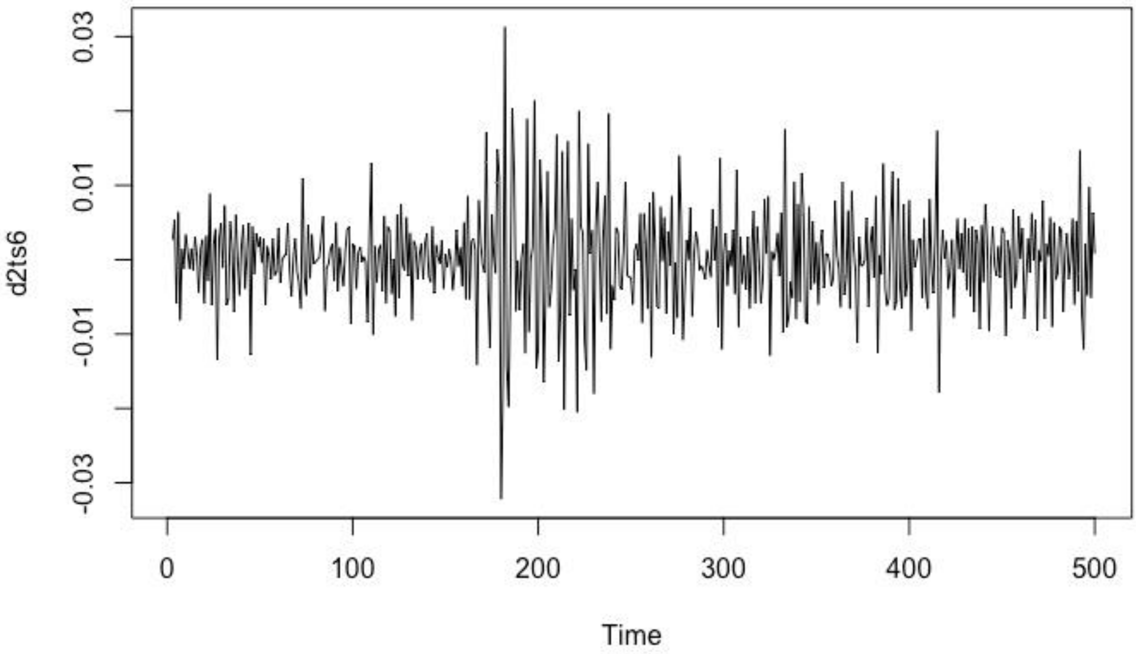
We graphically represent the time series of the first and second differences.

```
dts6 = diff(ts6, differences = 1)
plot.ts(dts6)
d2ts6 = diff(ts6, differences = 2)
plot.ts(d2ts6)
```

First Differences



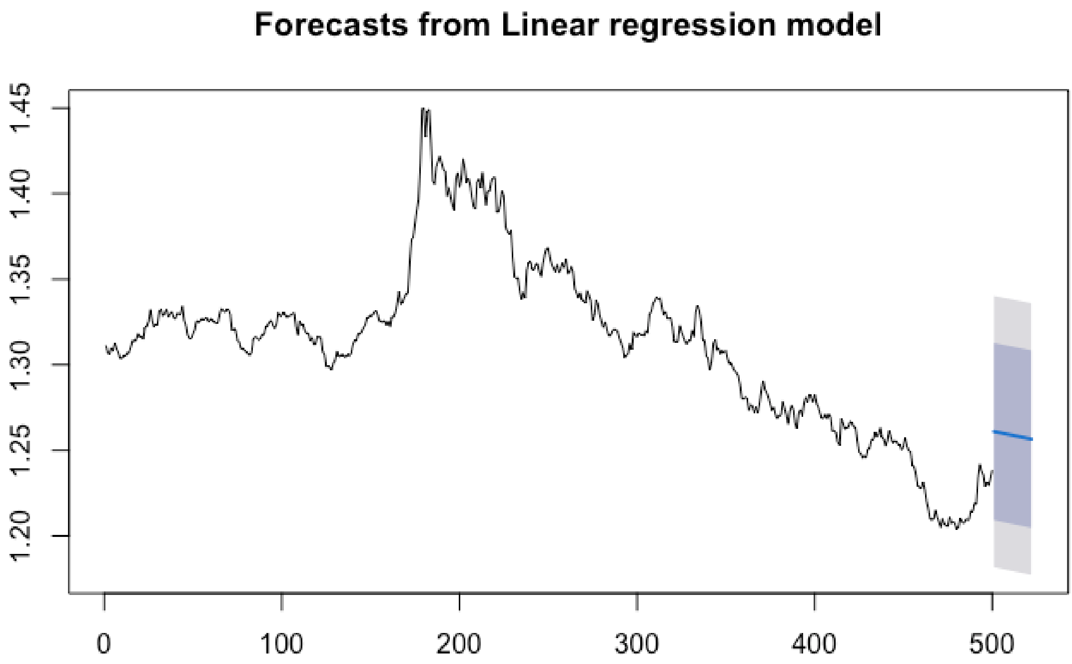
Second Differences



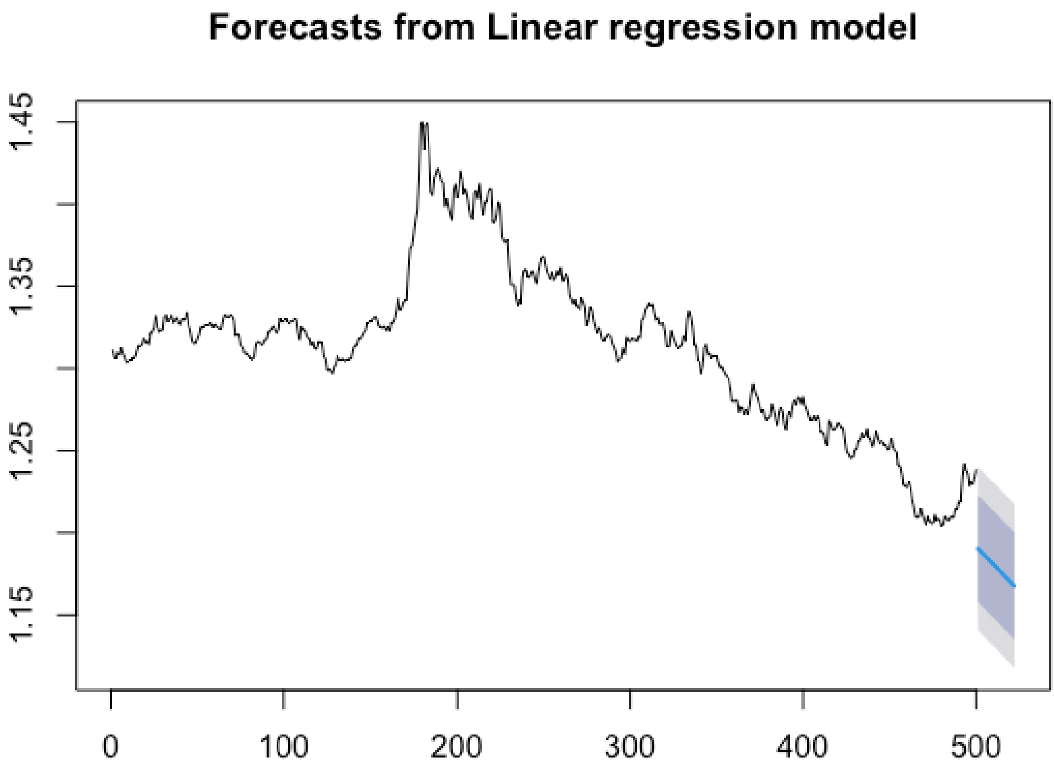
6.V)

The 4 figures below show the forecasts for the next 22 days of the four models.

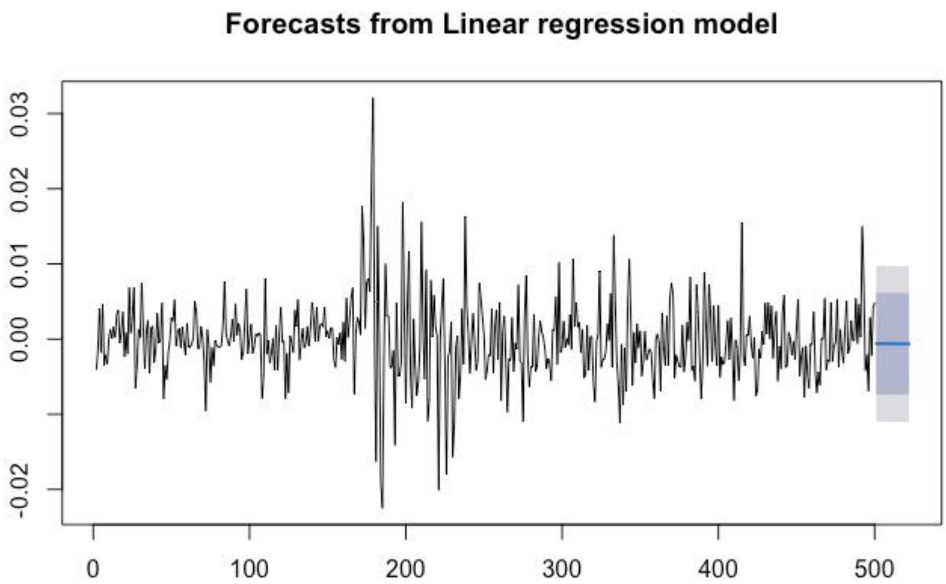
```
library(xts)
library("forecast")
model1 <- tslm( ts(ts6) ~ trend )
model1_f22 = forecast(model1, h=22)
model1_f22
plot(model1_f22)
```



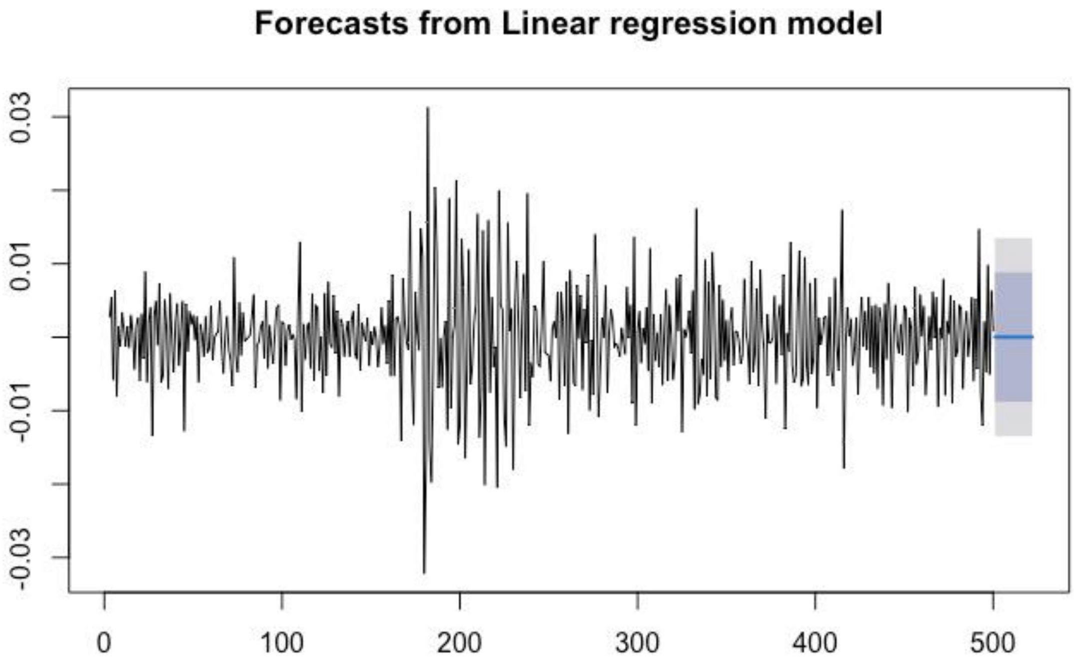

```
model2 <- tslm( ts(ts6) ~ trend + I(trend^2) )
model2_f22 = forecast(model2, h=22)
model2_f22
plot(model2_f22)
```




```
model3 <- tslm( ts(dts6) ~ 1 )
model3_f22 = forecast(model3, h=22)
model3_f22
plot(model3_f22)
```



```
model4 <- tslm( ts(d2ts6) ~ 1 )
model4_f22 = forecast(model4, h=22)
model4_f22
plot(model4_f22)
```



6.VI)

We add a seasonal component (January effect. The dummy-variable is statistically significant and therefore there is a significant deviation of January from the other months.

```
> df6$dates = as.Date(df6$date)
> df6$months = months(df6$dates)
> df6$Jan = ifelse(df6$months == "January", 1, 0)
> model6VI <- tslm( ts(df6$FXUSDCAD) ~  df6$Jan)
> model6VI

Call:
tslm(formula = ts(df6$FXUSDCAD) ~ df6$Jan)

Coefficients:
(Intercept)      df6$Jan
    1.31467     -0.02327
```

6.VII)

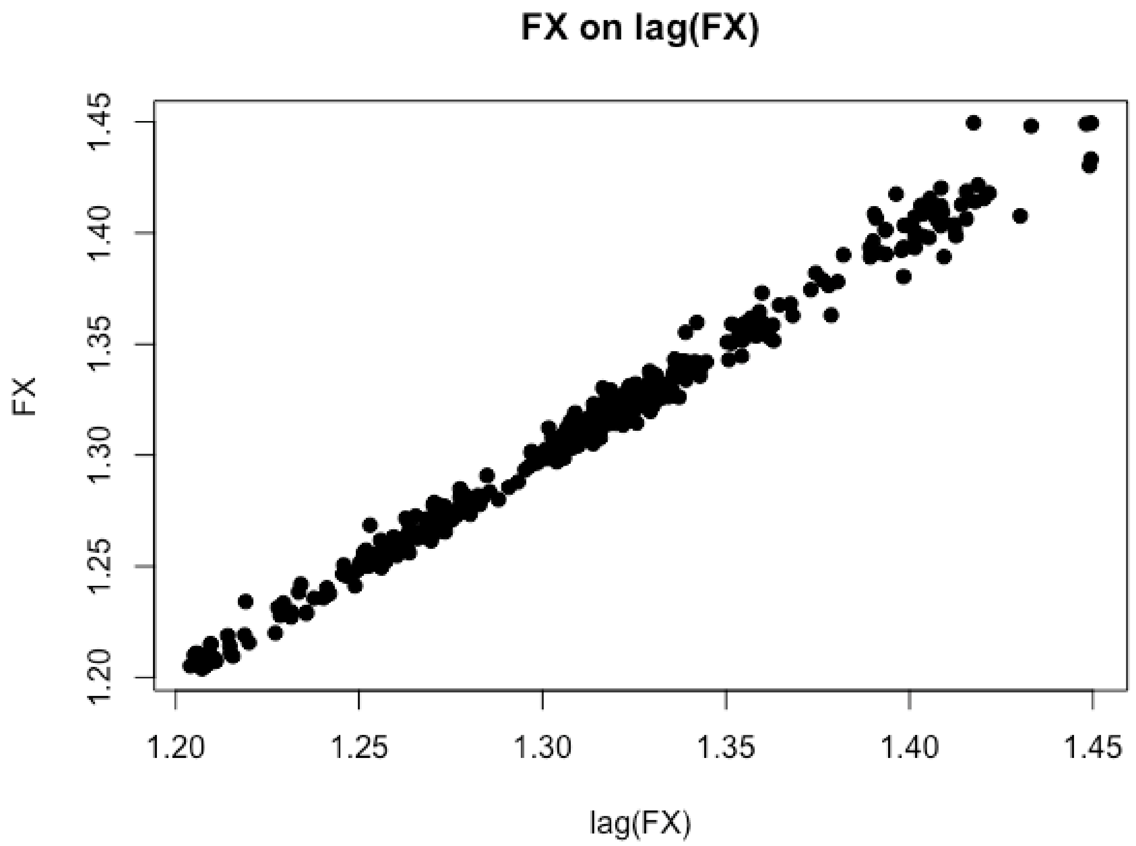
We adjust the square trend and the effect of January. All variables are statistically significant.

```
> model6VII <- tslm( ts(ts6) ~ trend + I(trend^2) + df6$Jan)
> summary(model6VII)

Call:
tslm(formula = ts(ts6) ~ trend + I(trend^2) + df6$Jan)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.051219 -0.015743 -0.003511  0.009685  0.092362 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.295e+00  3.166e-03  408.95  < 2e-16 ***
trend        6.579e-04  2.927e-05   22.47  < 2e-16 ***
I(trend^2)  -1.724e-06  5.658e-08  -30.48  < 2e-16 ***
df6$Jan     -3.189e-02  3.801e-03   -8.39 5.11e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.02349 on 496 degrees of freedom
Multiple R-squared:  0.781,Adjusted R-squared:  0.7797 
F-statistic: 589.7 on 3 and 496 DF,  p-value: < 2.2e-16
```

6.VIII)

Scatter plot diagram between exchange rates and the lagged values.

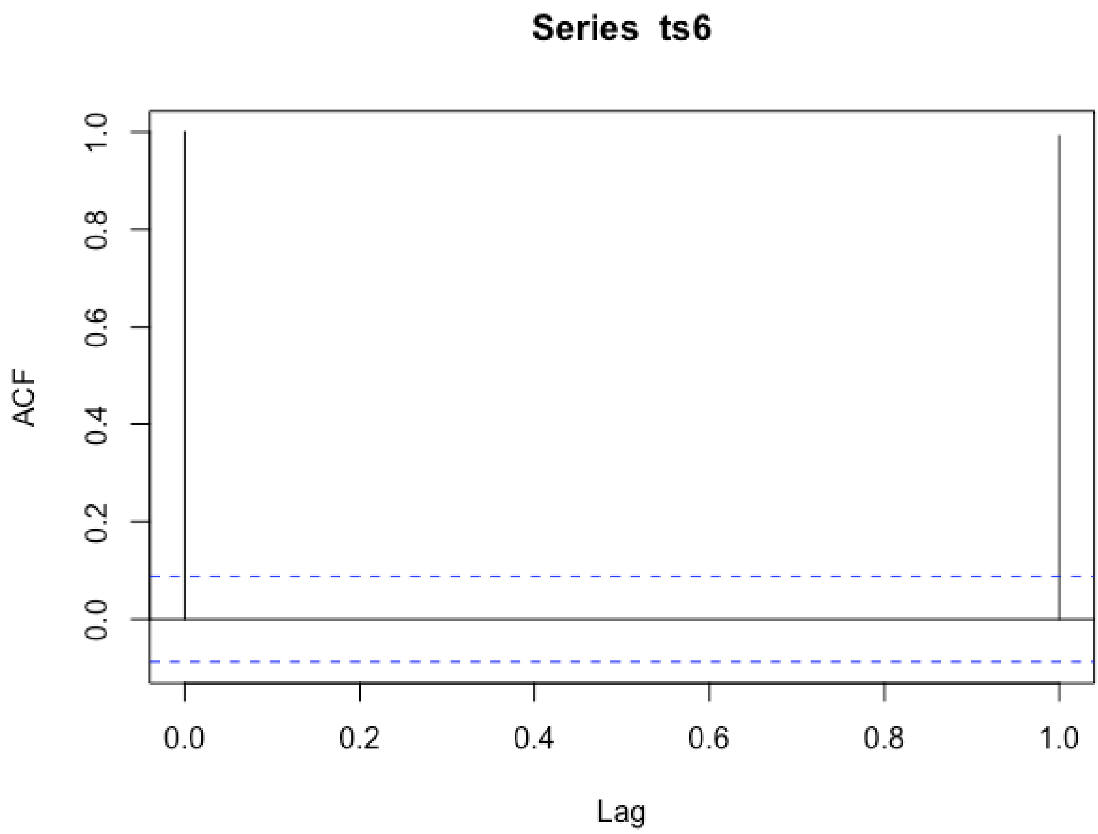
```
ts6_lag = lag(ts6, 1)
plot(as.vector(ts6_lag), df6$FXUSDCAD, main="FX on lag(FX)",
xlab="lag(FX)", ylab="FX", pch=19)
```



6.IX)

We calculate the autocorrelation with lag 1 and observe that there is an almost perfect linear correlation between yt and yt-1. Next we need to take the differences and check if we have a random walk or not.

```
> cor(as.vector(ts6_lag), df6$FXUSDCAD, method = "pearson", use =
"complete.obs")
[1] 0.9944947
acf(ts6, lag = 1)
```



6.X)

Function that calculates the k-lag autocorrelation of a time series.

```
AC <- function(y, k) {
  y0 <- y
  n0 <- NROW(y)
  k <- 2
  y <- y0[(k+1):n0]
  x <- y0[1:(n0-k)]
  n <- NROW(x)
  sx = sum((x-sum(x)/n)^2)
  sy = sum((y-sum(y)/n)^2)
  corr = (sum((x-sum(x)/n)*(y-sum(y)/n)))/(sx*sy)
  return(corr)
}

> AC( as.vector(ts6), 2 )
[1] 0.7936315
```

6.XI)

Model AR(1)

```
> model_ts_AR1 <- arima(ts6,c(1,0,0))
> model_ts_AR1

Call:
arima(x = ts6, order = c(1, 0, 0))

Coefficients:
         ar1  intercept
      0.9946     1.2937
s.e.  0.0040     0.0344

sigma^2 estimated as 2.747e-05:  log likelihood = 1913.85,  aic = -
3821.71
```

Model MA(1)

```
> model_ts_MA1 <- arima(ts6,c(0,0,1))
> model_ts_MA1

Call:
arima(x = ts6, order = c(0, 0, 1))

Coefficients:
         ma1  intercept
      0.9394     1.3126
s.e.  0.0097     0.0023

sigma^2 estimated as 0.0007018:  log likelihood = 1104.91,  aic = -
2203.82
```

6.XII)

Model ARMA(2,1,2)

```
> model_ts_ARMA212 <- arima(ts6,c(2,1,2))
> model_ts_ARMA212

Call:
arima(x = ts6, order = c(2, 1, 2))

Coefficients:
          ar1      ar2     ma1     ma2
      -0.7106  -0.5212  0.8835  0.5638
s.e.   0.6194   0.1486  0.6279  0.2579

sigma^2 estimated as 2.64e-05:  log likelihood = 1922.14,  aic = -
3834.29
```

Evaluation Measures

```
> accuracy(model_ts_AR1)
                        ME         RMSE          MAE          MPE
MAPE      MASE       ACF1
Training set -3.866334e-05 0.005241367 0.003677667 -0.004807655
0.2777307 1.002927 0.1505532
> accuracy(model_ts_MA1)
                       ME        RMSE        MAE         MPE      MAPE
MASE       ACF1
Training set 3.180891e-06 0.02649244 0.0197092 -0.07494507 1.507115
5.374844 0.8957084
> accuracy(model_ts_ARMA212)
                        ME         RMSE          MAE         MPE
MAPE      MASE          ACF1
Training set -0.0001299159 0.005133601 0.003672129 -0.01085626
0.27751 1.001417 -0.003874124
```

AR(1) and ARMA(2,1,2) Models have almost the same MAE, MSE & MAPE, and smaller than those given by MA(1).
We choose the AR(1) model because it is simpler than the ARMA(2,1,2) model.

R CODE APPENDIX

```
setwd('C:/Users/Giorgos/Desktop/ergasies_metaptxiakwn/ergasia
filippaki_(predictive
analytics)/ergasia_examinou/ergasia_mfilip_2021/ergasia_mfilip_2021/
dataset_merous_1_kai_2')


#1
library(MASS) #pima.te - diabetes dataset is in MASS library
data <- Pima.te
set.seed(789)
training = sample(nrow(data),265,replace = FALSE)
train = data[training, ]
test = data[-training, ]
model.all <- glm(type ~.,data = train,family = "binomial")
summary(model.all)
model2 <- glm(type ~ npreg+glu+skin+bmi+ped+age, data = train,family
= "binomial")
summary(model2)
model3 <- glm(type ~ npreg+glu+skin+bmi+ped, data = train,family =
"binomial")
summary(model3)
model4 <- glm(type ~ npreg+glu+bmi+ped, data = train,family =
"binomial")
summary(model4)
newdata = read.csv('meros1_exc1_new_data.csv')
pred_new <- predict(model4, newdata = newdata, type = "response")
pred_new


# 2
library(cluster)
df2 = read.csv('decathlon.csv')
df2_std <- as.data.frame( scale( df2[,2:11] ) )
summary( df2_std )
dist_mat <- dist( df2_std, method = 'euclidean')
hclust_avg <- hclust(dist_mat, method = 'average')
plot(hclust_avg)
# prune(hclust_avg)
# cutree(hclust_avg, k = 1:5)
# draw.clust (prune.clust (agnes (df2_std), k=6))
# kmeans clustering
cluster_kmeans7 <- kmeans(df2_std, 7)
cluster_kmeans7


# 3
# 3.1
library(astsa)
library(TTR)
ts = cmort
model_ts <- arima(ts,c(2,0,0))
ts_forecast <- predict(model_ts, n.ahead = 4)
msft_forecast_values <- ts_forecast$pred
```

```
msft_forecast_se <- ts_forecast$se
lower_bound = msft_forecast_values - 1.96*msft_forecast_se
upper_bound = msft_forecast_values + 1.96*msft_forecast_se
confidence_int = cbind(lower_bound, upper_bound)
msft_forecast_values
confidence_int

# 3.2
ts = arima.sim( n = 500, list(ar = 0.9, ma = -0.9, sd = 1) )
acf(ts, main = 'ARMA(1,1) ACF')
pacf(ts, main = 'ARMA(1,1) PACF')
model_ts <- arima(ts,c(1,0,1))
model_ts$coef

# 3.3
ts = oil
fit = auto.arima(ts)
fit
library(aTSA)
ts.diag(estimate(ts,p=1,d=1,q=3))

# 3.4
ts = globtemp
fit = auto.arima(ts)
fit
model_ts <- arima(ts,c(1,1,3))
library(aTSA)
ts.diag(estimate(ts,p=1,d=1,q=3))
ts_forecast <- predict(model_ts, n.ahead = 10)
ts_forecast$pred

# 3.5
ts = chicken
fit = auto.arima(ts)
fit
model_ts <- arima(ts,c(2,1,1))
library(aTSA)
ts.diag(estimate(ts,p=2,d=1,q=1))
ts_forecast <- predict(model_ts, n.ahead = 12)
ts_forecast$pred


# 4
dat = c(1,3,6,4,
        2,2,7,5,
        2,4,8,5,
        1,3,8,6)
ts <- ts(dat, start=c(2016, 1), end=c(2019, 4), frequency=4)
plot.ts(ts)
# 4.I
library(fpp)
ts_comp <- decompose(ts)
ts_SA <- ts - ts_comp$seasonal
plot.ts(ts_SA)
ts_comp$figure # Seasonality Indexes
# 4.II
```

```
x <- (1:length(ts))
ts.plot(ts)
plot(x,ts)
lines(predict(lm(ts~x)),col='green')
# 4.III
decomp <- stl(ts, s.window="periodic")
plot(decomp)


# 5
dat5 = c(37.44, 44.14, 46.25, 43.99, 51.84, 49.10, 58.56, 58.02,
70.28)
ts5 <- ts(dat5, start=c(2011), end=c(2019), frequency=1)
plot.ts(ts5)
x5 <- (1:length(ts5))
ts.plot(ts5)
plot(x5,ts5)
lines(predict(lm(ts5~x5)),col='green')


# 6
df6 = read.csv('FXUSDCAD.csv')
library(xts)
ts6 <- xts(df6[,-1], order.by=as.Date(df6[,1], "%d/%m/%Y"))
# 6.I
plot.ts(ts6)
# 6.II
x6 <- (1:length(ts6))
plot.ts(ts6)
plot(x6,ts6)
lines(predict(lm(ts6~x6)),col='green')
# 6.III
# Quadratic trend
plot.ts(ts6)
plot(x6,ts6)
lines( predict( lm( ts6 ~ x6 + I(x6^2) ) ), col='green' )

# 6.IV)
dts6 = diff(ts6, differences = 1)
plot.ts(dts6)
d2ts6 = diff(ts6, differences = 2)
plot.ts(d2ts6)

# 6.V)
library(xts)
library("forecast")
model1 <- tslm( ts(ts6) ~ trend )
model1_f22 = forecast(model1, h=22)
model1_f22
plot(model1_f22)

model2 <- tslm( ts(ts6) ~ trend + I(trend^2) )
model2_f22 = forecast(model2, h=22)
model2_f22
plot(model2_f22)
```

```
model3 <- tslm( ts(dts6) ~ 1 )
model3_f22 = forecast(model3, h=22)
model3_f22
plot(model3_f22)

model4 <- tslm( ts(d2ts6) ~ 1 )
model4_f22 = forecast(model4, h=22)
model4_f22
plot(model4_f22)

# 6.VI)
df6$dates = as.Date(df6$date)
df6$months = months(df6$dates)
df6$Jan = ifelse(df6$months == "January", 1, 0)
model6VI <- tslm( ts(df6$FXUSDCAD) ~  df6$Jan)
model6VI

# 6.VII)
model6VII <- tslm( ts(ts6) ~ trend + I(trend^2) + df6$Jan)
summary(model6VII)

# 6.VIII)
ts6_lag = lag(ts6, 1)
plot(as.vector(ts6_lag), df6$FXUSDCAD, main="FX on lag(FX)",
xlab="lag(FX)", ylab="FX", pch=19)

# 6.IX)
cor(as.vector(ts6_lag), df6$FXUSDCAD, method = "pearson", use =
"complete.obs")
acf(ts6, lag = 1)

# 6.X)
AC <- function(y, k) {
  y0 <- y
  n0 <- NROW(y)
  k <- 2
  y <- y0[(k+1):n0]
  x <- y0[1:(n0-k)]
  n <- NROW(x)
  sx = sum((x-sum(x)/n)^2)
  sy = sum((y-sum(y)/n)^2)
  corr = (sum((x-sum(x)/n)*(y-sum(y)/n)))/(sx*sy)
  return(corr)
}

AC( as.vector(ts6), 2 )


# 6.XI
model_ts_AR1 <- arima(ts6,c(1,0,0))
model_ts_AR1

model_ts_MA1 <- arima(ts6,c(0,0,1))
model_ts_MA1
```

```
# 6.XII
model_ts_ARMA212 <- arima(ts6,c(2,1,2))
model_ts_ARMA212

accuracy(model_ts_AR1)
accuracy(model_ts_MA1)
accuracy(model_ts_ARMA212)
```