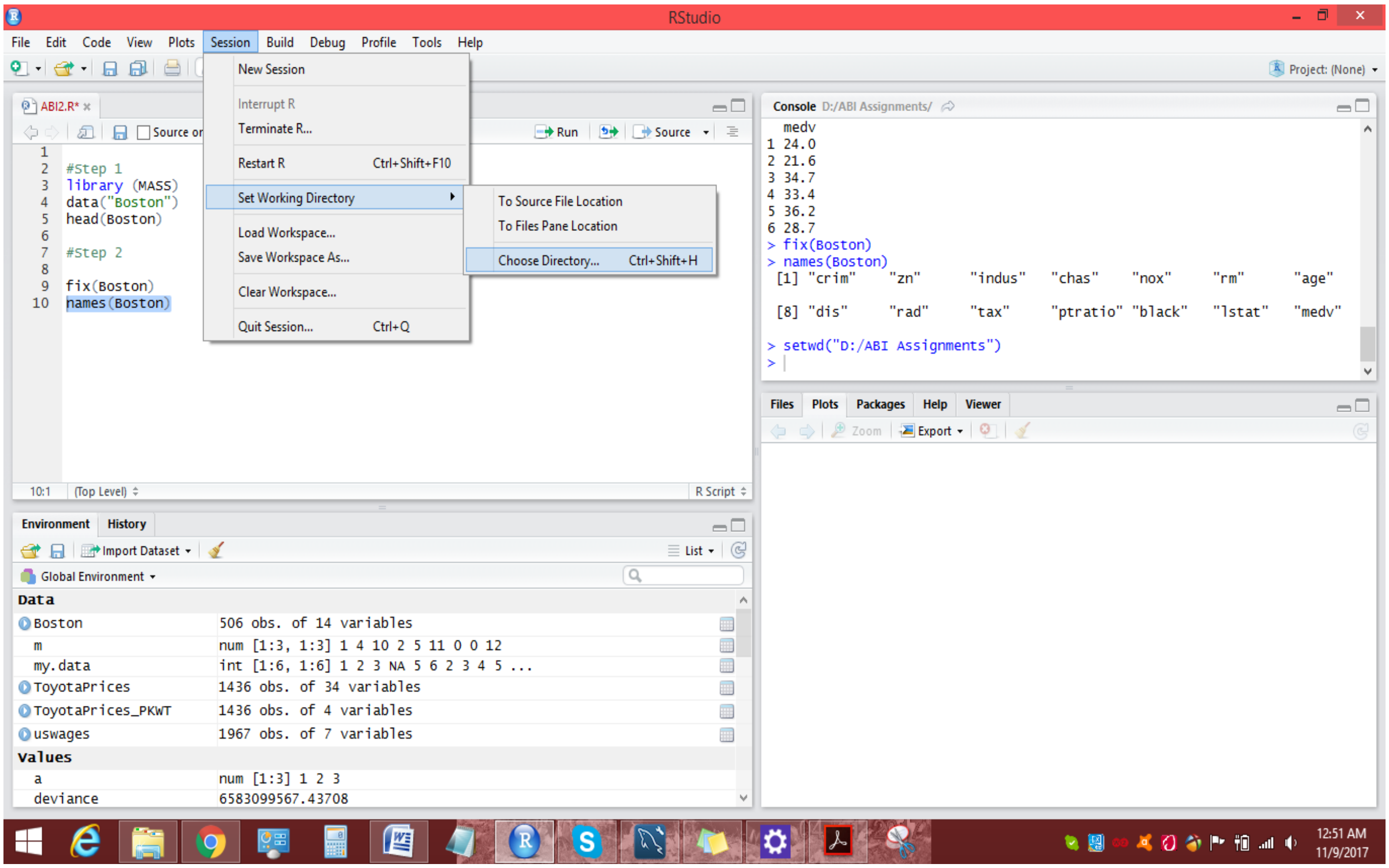
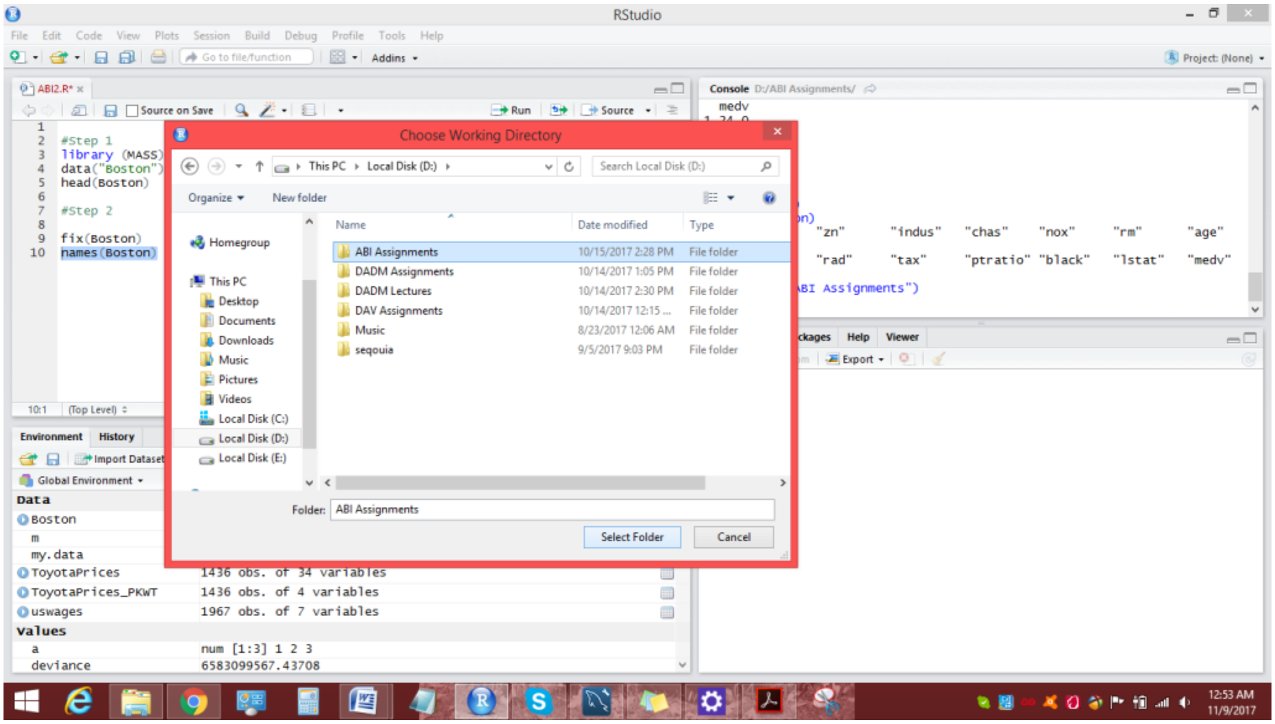
Regression with interaction

Step 1 You need create the working directory and connect Boston data file as you did this in Home Work 1.





Step 2 include interaction terms in a linear model using the lm() function. The syntax lstat:black tells R to include an interaction term between lstat and black. The syntax lstat*age simultaneously includes lstat, age, and the interaction term lstat×age as predictors; it is a shorthand for lstat+age+lstat:age.

COMMAND:-

library (MASS)
data("Boston")
head(Boston)

fix(Boston)
names(Boston)

summary(lm(medv~lstat:black, data=Boston))

summary(lm(medv~lstat*age, data = Boston))

Output :-

```
> summary(lm(medv~lstat:black, data=Boston))

Call:
lm(formula = medv ~ lstat:black, data = Boston)

Residuals:
     Min       1Q   Median       3Q      Max 
-21.9377  -3.4993  -0.9214   2.7866  26.0088 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) 30.4706654  0.6669783   45.69   <2e-16 ***
lstat:black -0.0018569  0.0001332  -13.95   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.82 on 504 degrees of freedom
Multiple R-squared:  0.2784,	Adjusted R-squared:  0.277 
F-statistic: 194.5 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
> fix(Boston)
> names(Boston)
 [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"     
 [7] "age"     "dis"     "rad"     "tax"     "ptratio" "black"  
[13] "lstat"   "medv"   
> summary(lm(medv~lstat*age, data = Boston))

Call:
lm(formula = medv ~ lstat * age, data = Boston)

Residuals:
    Min      1Q  Median      3Q     Max 
-15.806  -4.045  -1.333   2.085  27.552 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) 36.0885359  1.4698355  24.553  < 2e-16 ***
lstat       -1.3921168  0.1674555  -8.313 8.78e-16 ***
age         -0.0007209  0.0198792  -0.036   0.9711    
lstat:age    0.0041560  0.0018518   2.244   0.0252 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.149 on 502 degrees of freedom
Multiple R-squared:  0.5557,	Adjusted R-squared:  0.5531 
F-statistic: 209.3 on 3 and 502 DF,  p-value: < 2.2e-16

> 
```

Step 3 (0.5 mark) Use command dim to check how many observations and variables are in the Boston file. Find the names of variables in this file.

COMMAND :-

```
dim(Boston)
names(Boston)
```

OUTPUT:-

```
> dim(Boston)
[1] 506  14
> names(Boston)
 [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"
 [7] "age"     "dis"     "rad"     "tax"     "ptratio" "black"
[13] "lstat"   "medv"
>
```

There are 506 such observations.

Step 4 (0.5 mark) Run the multivariate regression of medv against lstat and age with interaction.

COMMAND:-

```
lm.fit =lm(medv~lstat*age, data = Boston)
attach(Boston)
lm.fit= lm(medv~lstat*age, data = Boston)
summary(lm.fit)
```

OUTPUT:-

```
> lm.fit =lm(medv~lstat*age, data = Boston)
> attach(Boston)
The following objects are masked from Boston (pos = 3):

    age, black, chas, crim, dis, indus, lstat, medv, nox,
    ptratio, rad, rm, tax, zn

> lm.fit= lm(medv~lstat*age, data = Boston)
> summary(lm.fit)

Call:
lm(formula = medv ~ lstat * age, data = Boston)

Residuals:
    Min      1Q  Median      3Q     Max 
-15.806  -4.045  -1.333   2.085  27.552 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) 36.0885359  1.4698355  24.553  < 2e-16 ***
lstat       -1.3921168  0.1674555  -8.313 8.78e-16 ***
age         -0.0007209  0.0198792  -0.036   0.9711    
lstat:age    0.0041560  0.0018518   2.244   0.0252 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.149 on 502 degrees of freedom
Multiple R-squared:  0.5557,	Adjusted R-squared:  0.5531 
F-statistic: 209.3 on 3 and 502 DF,  p-value: < 2.2e-16

> 
```

Step 5 (0.5 mark) Is interaction term significant or not. Is the answer difference for confidence probability 5% and 1%? Formulate the appropriate hypotheses and make a conclusion based on the relevant p-values.

Yes, The interaction term is significant here because the p-value is very small (0.0252) and less than 0.05.
Yes, the answer is different for confidence probability 5% and 1% because P-value is very small so it rejects null hypothesis. If P-value is greater than 5% we accept null hypothesis.

Step 6 (1 Mark) Implement the multiple linear regression of medv against lstat and age without interaction (or just see the results in HW1). Compare these two models with and without interact-ion.

COMMAND:-

lm.fit =lm(medv~lstat+age ,data=Boston)
summary (lm.fit)

OUTPUT:-

```
> lm.fit =lm(medv~lstat+age ,data=Boston )
> summary (lm.fit)

Call:
lm(formula = medv ~ lstat + age, data = Boston)

Residuals:
    Min      1Q  Median      3Q     Max 
-15.981  -3.978  -1.283   1.968  23.158 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 33.22276    0.73085  45.458  < 2e-16 ***
lstat       -1.03207    0.04819 -21.416  < 2e-16 ***
age          0.03454    0.01223   2.826  0.00491 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' '

Residual standard error: 6.173 on 503 degrees of freedom
Multiple R-squared:  0.5513,	Adjusted R-squared:  0.5495 
F-statistic:   309 on 2 and 503 DF,  p-value: < 2.2e-16
```

P-value is drastically changing if we put interaction effect in the model.
Without Interaction: - On Individual Basis, the lstat will still reject null hypothesis, but age will have it changed from being rejected without the interaction variable to being accepted with the interaction variable.

Step 7 (0.5 mark) Compare the residual standard errors for these two models.

The Residual Standard Error for Model 1 with Interaction - 6.149 on 502 degrees of freedom And the Residual standard error for Model 2 without Interaction - 6.173 on 503 degrees of freedom. As we can see that the Residual standard error for both the models are same, although the model without interaction will perform better but there will not be any significant impact on the model.

Nonlinear transform of predictors (3 marks)

Step 1 (1 mark) The lm() function can also accommodate non-linear transformations of the predictors. For instance, given a predictor X, we can create a predictor X^2 using I(X^2). The function I() is needed since the ^ has a special meaning I() in a formula; wrapping as we do allows the standard usage in R, which is to raise X to the power 2. We now perform a regression of medv onto lstat and $lstat^2$.

COMMAND:-

lm.fit1 = lm(medv~lstat + I(lstat^2))
summary(lm.fit1)

OUTPUT:-

```
> lm.fit1 = lm(medv~lstat + I(lstat^2))
> summary(lm.fit1)

Call:
lm(formula = medv ~ lstat + I(lstat^2))

Residuals:
     Min       1Q   Median       3Q      Max 
-15.2834  -3.8313  -0.5295   2.3095  25.4148 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 42.862007   0.872084   49.15   <2e-16 ***
lstat       -2.332821   0.123803  -18.84   <2e-16 ***
I(lstat^2)   0.043547   0.003745   11.63   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.524 on 503 degrees of freedom
Multiple R-squared:  0.6407,	Adjusted R-squared:  0.6393 
F-statistic: 448.5 on 2 and 503 DF,  p-value: < 2.2e-16
```

Step 2 (1 mark) Build the single regression of medv (results can be taken from HW1) against lstat. Compare linear and quadratic model using the adjuster R-squared.

COMMAND:-

lm.fit = lm(medv~lstat, data = Boston)
summary(lm.fit)

OUTPUT:-

```
> lm.fit = lm(medv~lstat, data = Boston)
> summary(lm.fit)

Call:
lm(formula = medv ~ lstat, data = Boston)

Residuals:
    Min      1Q  Median      3Q     Max
-15.168  -3.990  -1.318   2.034  24.500

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 34.55384    0.56263   61.41   <2e-16 ***
lstat       -0.95005    0.03873  -24.53   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.216 on 504 degrees of freedom
Multiple R-squared:  0.5441,    Adjusted R-squared:  0.5432
F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16

>
```

#Adjusted R-square for linear model is 0.5432 and for Quadratic model is 0.6393.
In the Quadratic model we see the adjusted R-squared value of 0.6393 is higher than the adjusted R-squared value of 0.5432 in linear model. That means, introduction of a new predictor term lstat^2 improves the model more than would be expected by chance. Also, since the value 0.6393 in quadratic model is closer to 1 and, 0.5432 in linear model is closer to 0, this indicates that the Quadratic model has a better fit.

Step 3 (1 mark) Use the anova() function to further quantify the extent to which the quadratic fit is superior to the linear fit. The anova() function performs a hypothesis test comparing the two models. The null hypothesis is that the two models fit the data equally well, and the alternative hypothesis is that the full model is superior.

COMMAND:-
lm.fit = lm(medv~lstat)
anova(lm.fit, lm.fit1)

OUTPUT:-

```
> lm.fit = lm(medv~lstat)
> anova(lm.fit, lm.fit1)
Analysis of Variance Table

Model 1: medv ~ lstat
Model 2: medv ~ lstat + I(lstat^2)
  Res.Df   RSS Df Sum of Sq     F    Pr(>F)
1    504 19472
2    503 15347  1    4125.1 135.2 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
```

#Quadratic Fit is superior model than linear fit because F-statstic is 135.2 and the associated p-value is highly significant.

Classification: Logistic regression (9 marks)

We will begin by examining some numerical and graphical summaries of the Smarket data, which is part of the ISLR library. This data set consists of percentage returns for the S&P 500 stock index over 1, 250 days, from the beginning of 2001 until the end of 2005. For each date, we have recorded the percentage returns for each of the five previous trading days, Lag1 through Lag5. We have also recorded Volume (the number of shares traded on the previous day, in billions), Today (the percentage return on the date in question) and Direction (whether the market was Up or Down on this date).

Step 1 (0.5 mark) Open library (ISLR) and check the name of variables in Smarket file and provide the summary statistics for al variables.

COMMAND:-

library(ISLR)
names(Smarket)
summary(Smarket)
dim(Smarket)

OUTPUT:-

```
> library(ISLR)
> names(Smarket)
[1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"
[6] "Lag5"      "Volume"    "Today"     "Direction"
> summary(Smarket)
      Year           Lag1                Lag2
 Min.   :2001   Min.   :-4.922000   Min.   :-4.922000
 1st Qu.:2002   1st Qu.:-0.639500   1st Qu.:-0.639500
 Median :2003   Median : 0.039000   Median : 0.039000
 Mean   :2003   Mean   : 0.003834   Mean   : 0.003919
 3rd Qu.:2004   3rd Qu.: 0.596750   3rd Qu.: 0.596750
 Max.   :2005   Max.   : 5.733000   Max.   : 5.733000
      Lag3                Lag4                Lag5
 Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.92200
 1st Qu.:-0.640000   1st Qu.:-0.640000   1st Qu.:-0.64000
 Median : 0.038500   Median : 0.038500   Median : 0.03850
 Mean   : 0.001716   Mean   : 0.001636   Mean   : 0.00561
 3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.59700
 Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.73300
     Volume           Today           Direction
 Min.   :0.3561   Min.   :-4.922000   Down:602
 1st Qu.:1.2574   1st Qu.:-0.639500   Up  :648
 Median :1.4229   Median : 0.038500
 Mean   :1.4783   Mean   : 0.003138
 3rd Qu.:1.6417   3rd Qu.: 0.596750
 Max.   :3.1525   Max.   : 5.733000
> dim(Smarket)
[1] 1250    9
>
```

Step 2 (0.5 mark) Use the cor() function produces a matrix that contains all of the pairwise correlations among the predictors in a data set. Use the parameter [,9] because the Direction (#9) variable is qualitative.

COMMAND:-

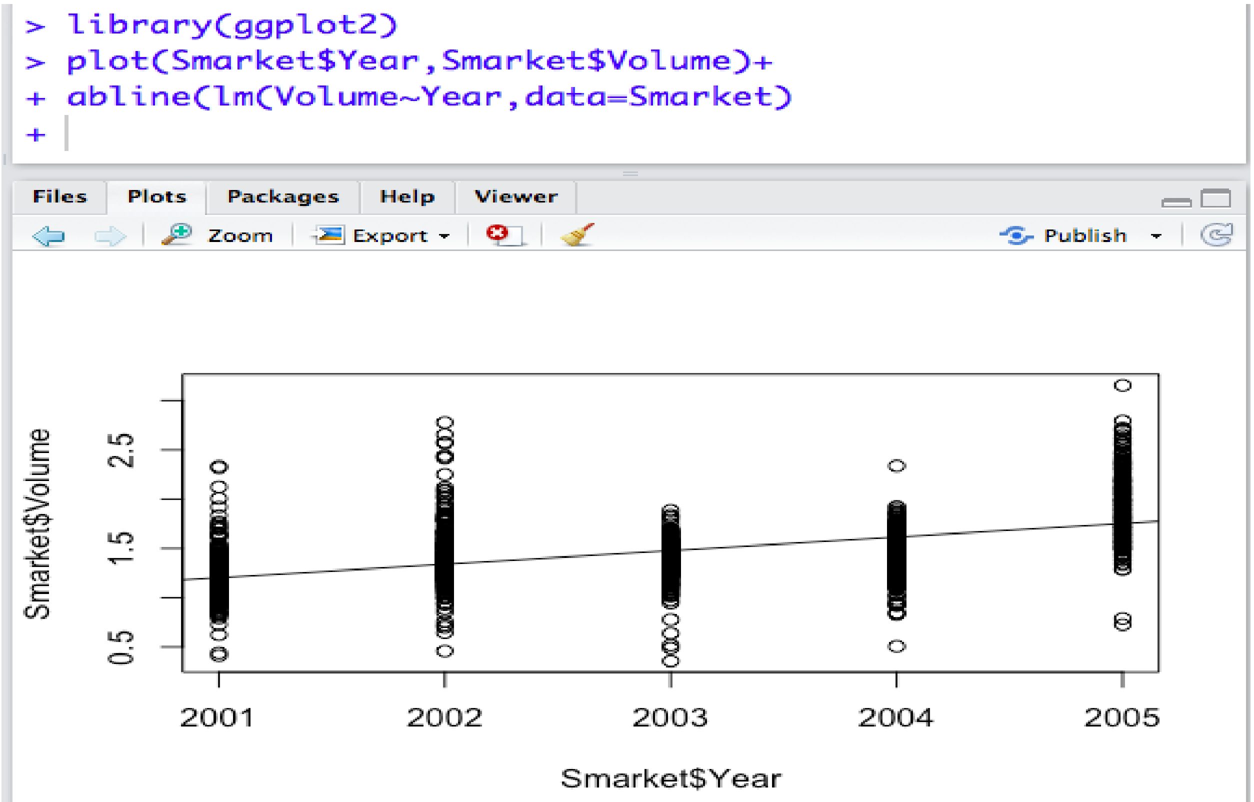
cor(Smarket [-9])

OUTPUT:-

```
> cor(Smarket [-9])
             Year         Lag1         Lag2         Lag3         Lag4
Year   1.00000000  0.029699649  0.030596422  0.033194581  0.035688718
Lag1   0.02969965  1.000000000 -0.026294328 -0.010803402 -0.002985911
Lag2   0.03059642 -0.026294328  1.000000000 -0.025896670 -0.010853533
Lag3   0.03319458 -0.010803402 -0.025896670  1.000000000 -0.024051036
Lag4   0.03568872 -0.002985911 -0.010853533 -0.024051036  1.000000000
Lag5   0.02978799 -0.005674606 -0.003557949 -0.018808338 -0.027083641
Volume 0.53900647  0.040909908 -0.043383215 -0.041823686 -0.048414246
Today  0.03009523 -0.026155045 -0.010250033 -0.002447647 -0.006899527
               Lag5      Volume        Today
Year    0.029787995  0.53900647  0.030095229
Lag1   -0.005674606  0.04090991 -0.026155045
Lag2   -0.003557949 -0.04338321 -0.010250033
Lag3   -0.018808338 -0.04182369 -0.002447647
Lag4   -0.027083641 -0.04841425 -0.006899527
Lag5    1.000000000 -0.02200231 -0.034860083
Volume -0.022002315  1.00000000  0.014591823
Today  -0.034860083  0.01459182  1.000000000
> |
```

Step 3 (1 mark) Explain why volume is correlated with year. Illustrate this graphically.

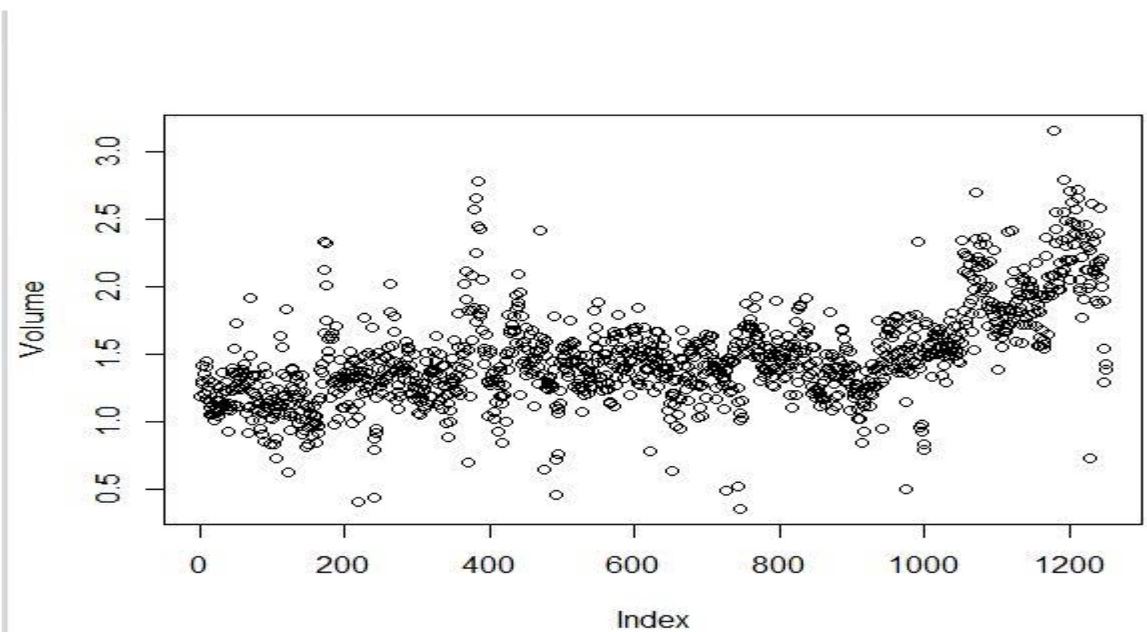
#There is a substantial correlation between Volume and Year.
#By plotting the graph we can see that the volume is increasing over time.



COMMAND:-

attach(Smarket)
plot(Volume)

OUTPUT:-



Step 4 (1 mark) Fit a logistic regression model in order to predict Direction using Lag1 through Lag5 and Volume. The glm() function fits *generalized linear models*, a class of models that includes logistic regression. The syntax of the glm() function is similar to that of lm(), except that we must pass in the argument family=binomial in order to tell R to run a logistic regression rather than some other type of generalized linear model.

COMMAND:-

glm.fit = glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume , data = Smarket , family = binomial)
summary(glm.fit)

OUTPUT:-

```
> glm.fit = glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume , data = Smarke
t , family = binomial)
> summary(glm.fit)

Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + 
    Volume, family = binomial, data = Smarket)

Deviance Residuals: 
   Min      1Q  Median      3Q     Max  
-1.446  -1.203   1.065   1.145   1.326  

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.126000   0.240736  -0.523    0.601
Lag1        -0.073074   0.050167  -1.457    0.145
Lag2        -0.042301   0.050086  -0.845    0.398
Lag3         0.011085   0.049939   0.222    0.824
Lag4         0.009359   0.049974   0.187    0.851
Lag5         0.010313   0.049511   0.208    0.835
Volume       0.135441   0.158360   0.855    0.392

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1731.2  on 1249  degrees of freedom
Residual deviance: 1727.6  on 1243  degrees of freedom
AIC: 1741.6

Number of Fisher Scoring iterations: 3

> |
```

Step 5 (1 mark) Analyze the logistic Table in order our ability to predict trading Volume using the information about lags.

The negative coefficient for this predictor suggests that if the market had a positive return yesterday, then it is less likely to go up today.
The smallest p-value here is associated with Lag1.
At a value of 0.15, the p-value is still relatively large, and so there is no clear evidence of a real association between Lag1and Direction.

Step 6 (0.5 mark) Use the coef() function in order to access just the coefficients for this fitted model. Check the consistency with the previous step.

COMMAND:-
coef(glm.fit)

OUTPUT:-

```
> coef(glm.fit)
 (Intercept)         Lag1         Lag2         Lag3         Lag4
-0.126000257 -0.073073746 -0.042301344  0.011085108  0.009358938
        Lag5       Volume
 0.010313068  0.135440659
```

INFERENCE:

The results for the coefficients for the model are consistent with the previous step as the values of the coefficients are same.

Step 7 (0.5 mark) Use the summary() function to access particular aspects of the fitted model, such as the p-values for the coefficients.

COMMAND:-

summary(glm.fit)$coef
summary (glm.fit)$coef[,4]

OUTPUT:-

```
> summary (glm.fit )$coef
               Estimate Std. Error    z value  Pr(>|z|)
(Intercept) -0.126000257 0.24073574 -0.5233966 0.6006983
Lag1        -0.073073746 0.05016739 -1.4565986 0.1452272
Lag2        -0.042301344 0.05008605 -0.8445733 0.3983491
Lag3         0.011085108 0.04993854  0.2219750 0.8243333
Lag4         0.009358938 0.04997413  0.1872757 0.8514445
Lag5         0.010313068 0.04951146  0.2082966 0.8349974
Volume       0.135440659 0.15835970  0.8552723 0.3924004
> summary (glm.fit )$coef[,4]
(Intercept)        Lag1        Lag2        Lag3        Lag4
  0.6006983   0.1452272   0.3983491   0.8243333   0.8514445
       Lag5      Volume
  0.8349974   0.3924004
> 
```

Step 8 (0.5 mark) Use the predict() function, which can be used to predict the probability that the market will go up, given values of the predictors. The type="response" option tells R to output probabilities of the form $P(Y = 1|X)$, as opposed to other information such as the logit. If no data set is supplied to the predict() function, then the probabilities are computed for the training data that was used to fit the logistic regression model. Print the first ten probabilities.

COMMAND:-
glm.probs = predict(glm.fit , type = "response")
glm.probs[1:10]

OUTPUT:-

```
> glm.probs = predict(glm.fit , type = "response")
> glm.probs[1:10]
        1         2         3         4         5         6         7
0.5070841 0.4814679 0.4811388 0.5152224 0.5107812 0.5069565 0.4926509
        8         9        10
0.5092292 0.5176135 0.4888378
> 
```

Step 9 (0.5 mark) Use contrasts() function, which indicates that R has created a dummy variable with a 1 for Up, to check if these values correspond to the probability of the market going up, rather than down.

COMMAND:-

contrasts(Direction)

OUTPUT:-

```
> contrasts(Direction)
     Up
Down  0
Up    1
> 
```

Step 10 (0.5 mark) Convert these predicted probabilities into class labels, Up or Down, in order to make our prediction more transparent. Check the result with command fix.

COMMAND:-

glm.pred=rep ("Down " ,1250)
glm.pred[glm.probs >.5]=" Up"
fix(Smarket)

OUTPUT:-

Data Editor

File Edit Help

	Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today	Direction
1	2001	0.381	-0.192	-2.624	-1.055	5.01	1.1913	0.959	Up
2	2001	0.959	0.381	-0.192	-2.624	-1.055	1.2965	1.032	Up
3	2001	1.032	0.959	0.381	-0.192	-2.624	1.4112	-0.623	Down
4	2001	-0.623	1.032	0.959	0.381	-0.192	1.276	0.614	Up
5	2001	0.614	-0.623	1.032	0.959	0.381	1.2057	0.213	Up
6	2001	0.213	0.614	-0.623	1.032	0.959	1.3491	1.392	Up
7	2001	1.392	0.213	0.614	-0.623	1.032	1.445	-0.403	Down
8	2001	-0.403	1.392	0.213	0.614	-0.623	1.4078	0.027	Up
9	2001	0.027	-0.403	1.392	0.213	0.614	1.164	1.303	Up
10	2001	1.303	0.027	-0.403	1.392	0.213	1.2326	0.287	Up
11	2001	0.287	1.303	0.027	-0.403	1.392	1.309	-0.498	Down
12	2001	-0.498	0.287	1.303	0.027	-0.403	1.258	-0.189	Down
13	2001	-0.189	-0.498	0.287	1.303	0.027	1.098	0.68	Up
14	2001	0.68	-0.189	-0.498	0.287	1.303	1.0531	0.701	Up
15	2001	0.701	0.68	-0.189	-0.498	0.287	1.1498	-0.562	Down
16	2001	-0.562	0.701	0.68	-0.189	-0.498	1.2953	0.546	Up
17	2001	0.546	-0.562	0.701	0.68	-0.189	1.1188	-1.747	Down
18	2001	-1.747	0.546	-0.562	0.701	0.68	1.0484	0.359	Up
19	2001	0.359	-1.747	0.546	-0.562	0.701	1.013	-0.151	Down

Step 11 (0.5 mark) Given these predictions, use the table() function to produce a confusion matrix in order to determine how many observations were correctly or incorrectly classified.

COMMAND:-

table(glm.pred ,Direction)
summary(Direction)

(507+145)/1250

OUTPUT:-

```
> table(glm.pred ,Direction )
        Direction
glm.pred Down  Up
    Up    457 507
   Down   145 141
> summary(Direction)
Down   Up
 602  648
>
> (507+145)/1250
[1] 0.5216
>
```

Step 12 (0.5 mark) Calculate the proportion of correct predictions using command mean. Check the results manually using the information from the confusion table.

COMMAND:-

mean(glm.pred==Direction)

OUTPUT:-

```
>
> mean(glm.pred==Direction)
[1] 0.5216
>
```

Step 13 (0.5 mark) The statistics in Step 13 corresponds to the training set equal to the total set of observation. Create the new training set for records from 2001 to 2004. We will then use this vector to create a held out data set of observations from 2005. How many records remains in the test set?

COMMAND:-

train =(Year<2005)
Smarket.2005= Smarket [!train ,]
dim(Smarket.2005)

OUTPUT:-

```
> train =(Year <2005)
> Smarket.2005= Smarket [!train ,]
> dim(Smarket.2005)
[1] 252   9
```

```
# There are 252 observations and 9 variables left.
```

Step 14 (1 mark) Implement command

```
>Direction.2005= Direction [! train]
```

The object train is a vector of 1*,* 250 elements, corresponding to the observations in our data set. The elements of the vector that correspond to observations that occurred before 2005 are set to TRUE, whereas those that correspond to observations in 2005 are set to FALSE. The object train is a *Boolean* vector, since its elements are TRUE and FALSE.

Fit a logistic regression model using only the subset of the observations that correspond to dates before 2005, using the subset argument. Compute the predictions for 2005 and compare them to the actual movements of the market over that time period. Interpret the results.

COMMAND:-

```
Direction.2005= Direction[!train]
glm.fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume ,
          data=Smarket, family =binomial,subset =train)

glm.probs=predict(glm.fit, Smarket.2005,type="response")
glm.pred=rep ("Down",252)

glm.pred[glm.probs >0.5]="Up"

table(glm.pred, Direction.2005)

mean(glm.pred==Direction.2005)
mean(glm.pred!= Direction.2005)
```

OUTPUT:-

```
> Direction.2005= Direction[!train]
> glm.fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume ,
+              data=Smarket, family =binomial,subset =train)
>
> glm.probs=predict(glm.fit, Smarket.2005,type="response")
> glm.pred=rep ("Down",252)
>
> glm.pred[glm.probs >0.5]="Up"
>
> table(glm.pred, Direction.2005)
        Direction.2005
glm.pred Down Up
    Down   77 97
    Up     34 44
>
> mean(glm.pred==Direction.2005)
[1] 0.4801587
> mean(glm.pred!= Direction.2005)
[1] 0.5198413
>
```

COMMAND:-

```
glm.fits=glm(Direction~Lag1+Lag2,data=Smarket ,family =binomial ,
          subset =train )
glm.probs = predict(glm.fits,Smarket.2005,type = "response")
glm.pred = rep("Down", 252)
glm.pred[glm.probs>.5]="Up"
table(glm.pred,Direction.2005)

mean(glm.pred == Direction.2005)
106/(106+76)

predict(glm.fits,newdata=data.frame(Lag1=c(1.2,1.5),Lag2=c(1.1,-0.8)),
      type="response")
```

OUTPUT:-

```
> glm.fits=glm(Direction~Lag1+Lag2,data=Smarket ,family =binomial ,
+             subset =train )
> glm.probs = predict(glm.fits,Smarket.2005,type = "response")
> glm.pred = rep("Down", 252)
> glm.pred[glm.probs>.5]="Up"
> table(glm.pred,Direction.2005)
        Direction.2005
glm.pred Down  Up
    Down   35  35
    Up     76 106
>
> mean(glm.pred == Direction.2005)
[1] 0.5595238
> 106/(106+76)
[1] 0.5824176
>
> predict(glm.fits,newdata=data.frame(Lag1=c(1.2,1.5),Lag2=c(1.1,-0.8)),
+         type="response")
        1         2
0.4791462 0.4960939
```

Now the results appear to be a little better: 56% of the daily movements have been correctly predicted. The results suggest that there 52% test error rate which shows that it is worse than random guessing. Also the p-value of all the predictors is highly insignificant which can be seen using summary command. So, we also tried by removing the variables that are not helpful in predicting Direction, thereby making a more effective model. As these variables might cause Deterioration in the test error rate so, we remove such variables to yield an improvement. After the removal of certain variables it can be seen that there has been better prediction of the Direction which 56%. To predict the values of Lag1 and Lag2 on a particular day for a particular value of Lag1 and Lag2 we use predict() Function.