BHU

Banaras Hindu University

PRACTICAL FILE

Name : Shubham Kumar

Examination Roll no. : 19419STC036

Enrollment no. : 412131

MSC. Statistics and Computing SEMESTER III

Subject Code : MSMS - 308

SL.NO.	Subject	
1.	Linear Algebra and basics of Linear Regression	
2.	Statistical Machine Learning	
3.	Life Time data Aanalysis	

<u>Linear Algebra and basics of Linear Regression</u>

Wild cats are well drilled to find and produce oil and/or natural gas in an improved area or to find new regressor in a field previously used for the same purpose . Table below gives data on

Y: The number of white cats drilled

X2: Price at the well head in previous period

X3: Domestic output

X4: GNP constant dollars(1972=100)

X5: Trend value 1948-1,1949-2,…………………….1978-31

<u>PROBLEM-1</u>

a) Find estimate of variance V(Y)

b) Test the hypothesis that $H_0 : \beta= \beta_0$ against

$H_1 : \beta \neq \beta_0$, Where β_0 =[2.0,7.0,-2.5,4.8] for the MLR

With X4 and with lnX4

ROW	Y	X2	X3	X4	X5
1	8.01	4.89	5.52	487.67	1
2	9.06	4.83	5.05	490.59	2
3	10:31	4.68	5.41	533.55	3
4	11.76	4.42	6.16	576.57	4
5	12.43	4.36	6.26	598.62	5
6	13.31	4.55	6.34	621.77	6
7	13.1	4.66	6.81	613.67	7
8	14.94	4.54	7.15	654.8	8
9	16.17	4.44	7.17	668.84	9
10	14.71	4.75	6.71	681.02	10
11	13.2	4.56	7.05	679.53	11
12	13.19	4.29	7.04	720.53	12
13	11.7	4.19	7.18	736.86	13
14	10.99	4.17	7.33	755.34	14
15	10.8	4.11	7.54	799.15	15
16	10.66	4.04	7.61	830.7	16
17	10.75	3.96	7.8	874.29	17
18	9.47	3.85	8.3	925.86	18
19	10.31	3.75	8.81	980.98	19
20	8.88	3.69	8.66	1,007.72	20
21	8.88	3.56	8.78	1,051.83	21
22	9.7	3.56	9.18	1,078.76	22
23	7.69	3.48	9.03	1,075.31	23
24	6.92	3.53	9	1,107.48	24
25	7.54	3.39	8.78	1,171.10	25
26	7.47	3.68	8.38	1,234.97	26
27	8.63	5.92	8.01	1,217.81	27
28	9.21	6.03	7.78	1,202.36	28
29	9.23	6.12	7.88	1,271.01	29
30	9.96	6.05	7.88	1,332.67	30
31	10.78	5.89	8.67	1,385.10	31

SOLUTION

CODE-

a)

```
> SS=as.matrix(DATA_REGRESSION)
> Y=SS[,1]
> X1=rep(1,length(Y))
> X2=SS[,2]
> X3=SS[,3]
> X4=SS[,4]
> X5=SS[,5]
> X=matrix(c(X1,X2,X3,X4,X5),nrow =31)
> Y
 [1]  8.0100000  9.0600000  0.4381944 11.7600000 12.4300000 13.3100000 13.
1000000 14.9400000
 [9] 16.1700000 14.7100000 13.2000000 13.1900000 11.7000000 10.9900000 10.
8000000 10.6600000
[17] 10.7500000  9.4700000 10.3100000  8.8800000  8.8800000  9.7000000  7.
6900000  6.9200000
[25]  7.5400000  7.4700000  8.6300000  9.2100000  9.2300000  9.9600000 10.
7800000
> X1
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
> X2
 [1] 4.89 4.83 4.68 4.42 4.36 4.55 4.66 4.54 4.44 4.75 4.56 4.29 4.19 4.17
4.11 4.04 3.96 3.85
[19] 3.75 3.69 3.56 3.56 3.48 3.53 3.39 3.68 5.92 6.03 6.12 6.05 5.89
> X3
 [1] 5.52 5.05 5.41 6.16 6.26 6.34 6.81 7.15 7.17 6.71 7.05 7.04 7.18 7.33
7.54 7.61 7.80 8.30
[19] 8.81 8.66 8.78 9.18 9.03 9.00 8.78 8.38 8.01 7.78 7.88 7.88 8.67
> X4
 [1]  487.67  490.59  533.55  576.57  598.62  621.77  613.67  654.80  668.
84  681.02  679.53
[12]  720.53  736.86  755.34  799.15  830.70  874.29  925.86  980.98 1007.
72 1051.83 1078.76
[23] 1075.31 1107.48 1171.10 1234.97 1217.81 1202.36 1271.01 1332.67 1385.
10
> X5
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
24 25 26 27 28 29 30
[31] 31
> Beta_estimates=solve((t(X)%*%X))%*%t(X)%*%Y
> Beta_estimates
             [,1]
[1,] -20.29092004
[2,]   3.49103943
[3,]   4.55775258
[4,]  -0.02289977
[5,]   0.06220024
> r=qr(X)$rank
> r
[1] 5
> n=length(X1)
> 
> # a) Estimating the variance of Y
> #dim of error space= n-r
> 
> sigma_hat_2= (t(Y)%*%Y- t(Beta_estimates)%*%t(X)%*%Y)/(n-r)
> sigma_hat_2
         [,1]
[1,] 5.977041
```

INTERPETATION: estimate of variance V(Y) = 5.977041 b)

```
> Beta_given=matrix(c(20,2.0,7.0,-2.5,4.8),nrow=5)
> p=length((Beta_given))
>
> F=(t(Beta_estimates-Beta_given)%*%(t(X)%*%X)%*%(Beta_estimates-Beta_give
n))/(p*sigma_hat_2)
> Ftab=qf(0.95,p,n-p)
> F
        [,1]
[1,] 4792525
> Ftab
[1] 2.58679
> if(F>Ftab)
+ {cat("Reject the hypothesis that all explanatory are not significant")}
else{cat("Accept null")}
Reject the hypothesis that all explanatory are not significant>
>
> # Now testing the values of Betas when "lnX4" is used instead of "X4"
>
> ln_X4=log(X4)
>
> X_1=matrix(c(X1,X2,X3,ln_X4,X5),nrow =31)
>
> Beta_estimates_1=solve((t(X_1)%*%X_1))%*%t(X_1)%*%Y
> Beta_estimates
             [,1]
[1,] -20.29092004
[2,]   3.49103943
[3,]   4.55775258
[4,]  -0.02289977
[5,]   0.06220024
> r=qr(X_1)$rank
> r
[1] 5
> n=length(X1)
>
> #Estimating the variance of Y with "lnX4"
> #dim of error space= n-r
>
> sigma_hat_2_1= (t(Y)%*%Y- t(Beta_estimates_1)%*%t(X_1)%*%Y)/(n-r)
> sigma_hat_2_1
         [,1]
[1,] 6.752526
>
> #Testing the Beta values with "lnX4"
>
> Beta_given=matrix(c(20,2.0,7.0,-2.5,4.8),nrow=5)
> p=length((Beta_given))
>
> F=(t(Beta_estimates_1-Beta_given)%*%(t(X_1)%*%X_1)%*%(Beta_estimates_1-B
eta_given))/(p*sigma_hat_2_1)
> Ftab=qf(0.95,p,n-p)
> F
         [,1]
[1,] 18087.22
> Ftab
[1] 2.58679
> if(F>Ftab)
+ {cat("Reject the hypothesis that all explanatory are not significant")}
else{cat("Accept null")}
Reject the hypothesis that all explanatory are not significant>
```

INTERPETATION: In both the casaes with X4 and lnX4 we reject Ho :$\beta = \beta_0$

PROBLEM-2

a) Using coefficients of determination , check if MLR model with X4 is better or that with lnX4.

b) Perform ANOVA to test goodness of fit of both the fitted MLR model in problem 1 and conclude which MLR model should be used.

c) Test the Hypothesis that

$H_0 : \beta_2 = \beta_3 = \beta_4 = \beta_5$ against

H_1 : at least one of these differ

SOLUTION

CODE-

a)

```
R_2_with_X4=((t(Beta_estimates)%*%t(X)%*%Y)-(n*(mean(Y)^2)))/((t(Y)%*%Y)-n
*(mean(Y)^2))
> R_2_with_X4
          [,1]
[1,] 0.4184495
>
> R_2_with_ln_X4=((t(Beta_estimates_1)%*%t(X_1)%*%Y)-(n*(mean(Y)^2)))/((t(
Y)%*%Y)-n*(mean(Y)^2))
> R_2_with_ln_X4
          [,1]
[1,] 0.3429968
>
> if(R_2_with_X4>R_2_with_ln_X4)
+ {cat("MLR model with X4 is better")} else{cat("MLR model with lnX4 is be
tter")}
MLR model with X4 is better>
```

INTERPETATION: MLR model with X4 is better.

b)

```
Ftab_A=qf(0.95,4,26)
>
> fit_with_ln_X4 <- lm(Y ~ X1+X2+X3+ln_X4+X5, data=DATA_REGRESSION)
> summary(fit_with_ln_X4) # show results
```

```
lm(formula = Y ~ X1 + X2 + X3 + ln_X4 + X5, data = DATA_REGRESSION)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.7642 -1.3667  0.6377  1.4303  3.4360 

Coefficients: (1 not defined because of singularities)
            Estimate Std. Error t value Pr(>|t|)   
(Intercept)  11.1061   119.5699   0.093  0.92671   
X1                NA         NA      NA       NA   
X2            3.3993     1.1073   3.070  0.00496 **
X3            4.9536     1.6063   3.084  0.00480 **
ln_X4        -6.9049    19.7431  -0.350  0.72935   
X5           -0.4173     0.6418  -0.650  0.52129   
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.599 on 26 degrees of freedom
Multiple R-squared:  0.343,	Adjusted R-squared:  0.2419 
F-statistic: 3.393 on 4 and 26 DF,  p-value: 0.0232

> 
> F_calculated_ln_X4=summary(fit_with_ln_X4)$fstatistic[1]
> 
> if(F_calculated_ln_X4<Ftab_A)
+ {cat("MLR model is significant with ln_X4")} else{cat("MLR model is not significant with ln_X4")}
MLR model is not significant with ln_X4>
> 
> fit_with_X4 <- lm(Y ~ X1+X2+X3+X4+X5, data=DATA_REGRESSION)
> summary(fit_with_X4) # show results

Call:
lm(formula = Y ~ X1 + X2 + X3 + X4 + X5, data = DATA_REGRESSION)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.2348 -1.2829  0.1903  1.3951  3.3832 

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)   
(Intercept) -20.29092   13.29068  -1.527  0.13891   
X1                 NA         NA      NA       NA   
X2            3.49104    1.03958   3.358  0.00243 **
X3            4.55775    1.40048   3.254  0.00315 **
X4           -0.02290    0.01222  -1.874  0.07222 . 
X5            0.06220    0.40686   0.153  0.87968   
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.445 on 26 degrees of freedom
Multiple R-squared:  0.4184,	Adjusted R-squared:  0.329 
F-statistic: 4.677 on 4 and 26 DF,  p-value: 0.005604

> 
> F_calculated_X4=summary(fit_with_X4)$fstatistic[1]
> 
> if(F_calculated_X4<Ftab_A)
+ {cat("MLR model is significant with X4")} else{cat("MLR model is not significant with X4")}
MLR model is not significant with X4>
```

INTERPETATION: MLR model is not significant in both the cases with X4 and with lnX4.

c)

```
> fit_with_X4 <- lm(Y ~ X1+X2+X3+X4+X5, data=DATA_REGRESSION)
> summary(fit_with_X4) # show results

Call:
lm(formula = Y ~ X1 + X2 + X3 + X4 + X5, data = DATA_REGRESSION)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.2348 -1.2829  0.1903  1.3951  3.3832 

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)   
(Intercept) -20.29092   13.29068  -1.527  0.13891   
X1                 NA         NA      NA       NA   
X2            3.49104    1.03958   3.358  0.00243 **
X3            4.55775    1.40048   3.254  0.00315 **
X4           -0.02290    0.01222  -1.874  0.07222 . 
X5            0.06220    0.40686   0.153  0.87968   
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.445 on 26 degrees of freedom
Multiple R-squared:  0.4184,	Adjusted R-squared:  0.329 
F-statistic: 4.677 on 4 and 26 DF,  p-value: 0.005604

> 
> F_calculated_X4=summary(fit_with_X4)$fstatistic[1]
> 
> if(F_calculated_X4<Ftab_A)
+ {cat("Accept Ho ")} else{cat("Accept h1 At least one of the Beta differs
")}
Accept h1 At least one of the Beta differs>
```

INTERPETATION: we accept H_1 i.e at least one of the Beta differs.

PROBLEM-3

a) Test if there is significant increase in contribution in goodness of fit by keeping X4 in MLR model,then by deleting it.

b) Perform MWD test on the data given in problem 1 and conclude whether linear regression model is appropriate or log-linear model fits better.

SOLUTION

CODE-

a)

```
> fit_with_X4 <- lm(Y ~ X1+X2+X3+X4+X5, data=DATA_REGRESSION)
>
>
> F_calculated_X4=summary(fit_with_X4)$fstatistic[1]
>
> Adjusted_r_square_with_X4=summary(fit_with_X4)$adj.r.squared
> Adjusted_r_square_with_X4
[1] 0.3289802
>
> fit_without_X4 <- lm(Y ~ X1+X2+X3+X5, data=DATA_REGRESSION)
>
> F_calculated_without_X4=summary(fit_with_X4)$fstatistic[1]
>
> Adjusted_r_square_without_X4=summary(fit_without_X4)$adj.r.squared
> Adjusted_r_square_without_X4
[1] 0.2665621
```

INTERPEТATION: since "Adjusted_r_square"value is more with regressor X4 so keeping X4 in model fit good

b)

```
> lin_reg<-function(y,X)
+ {
+   b=solve((t(X)%*%X))%*%t(X)%*%y
+   n=length(X[,1])
+   r=qr(X)$rank
+   p=nrow(b)
+   sigma_hat2= (t(y)%*%y- t(b)%*%t(X)%*%y)/(n-r)
+   return(list(b,sigma_hat2,p,r))
+ }
>
> slr= lin_reg(Y,X)
> llm= lin_reg(log(Y),log(X[1:n,2:p]))
> y_hat=X%*%slr[[1]]
> lny_hat=log(X[1:n,2:p])%*%llm[[1]]
> Z1= lny_hat-log(y_hat)
> test_lin= lin_reg(Y,cbind(X,Z1))
> Z2=exp(lny_hat)-y_hat
> test_log= lin_reg(log(Y),cbind(log(X[1:n,2:p]),Z2))
>
> ttest<-function(X,b,sb2,i)
+ {
+   s=solve(t(X)%*%X)[i,i]
+   t= b/sqrt(sb2*s)
+   return(t)
+ }
>
> p1=test_lin[[3]]
> sblin=test_lin[[2]][1,1]
> b1=test_lin[[1]][p1]
> p2=test_log[[3]]
> sblog=test_log[[2]][1,1]
> b2=test_log[[1]][p2]
> X1=cbind(X,Z1)
> X2=cbind(log(X[1:n,2:p]),Z2)
> t1=ttest(X1,b1,sblin,p1)
> t2=ttest(X2,b2,sblog,p2)
>
> if(abs(t1)>qt(0.975,n-p1)){cat("Reject Ho \n")} else{ cat("Accept Ho \n"
)}
Accept Ho
>
> if(abs(t2)>qt(0.975,n-p2)){cat("Reject H1 \n")} else{ cat("Accept H1 \n"
)}
Accept H1
```

INTERPETATION: Both the models give same value of MSE and hence, both are appropriate for fitting.

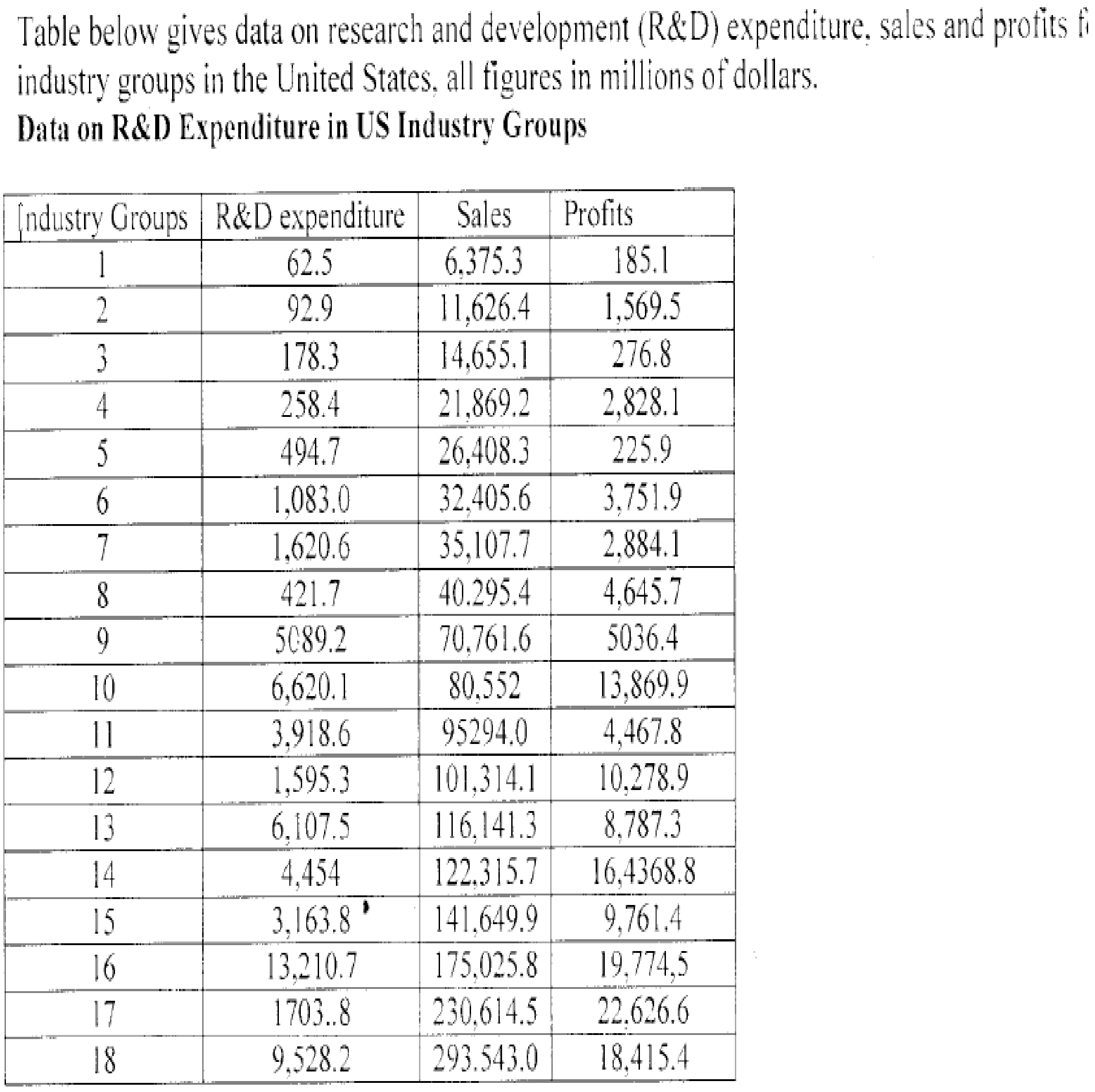


Table below gives data on research and development (R&D) expenditure, sales and profits f industry groups in the United States, all figures in millions of dollars.

Data on R&D Expenditure in US Industry Groups

Industry Groups	R&D expenditure	Sales	Profits
1	62.5	6,375.3	185.1
2	92.9	11,626.4	1,569.5
3	178.3	14,655.1	276.8
4	258.4	21,869.2	2,828.1
5	494.7	26,408.3	225.9
6	1,083.0	32,405.6	3,751.9
7	1,620.6	35,107.7	2,884.1
8	421.7	40.295.4	4,645.7
9	5089.2	70,761.6	5036.4
10	6,620.1	80,552	13,869.9
11	3,918.6	95294.0	4,467.8
12	1,595.3	101,314.1	10,278.9
13	6,107.5	116,141.3	8,787.3
14	4,454	122,315.7	16,4368.8
15	3,163.8	141,649.9	9,761.4
16	13,210.7	175,025.8	19,774,5
17	1703..8	230,614.5	22,626.6
18	9,528.2	293.543.0	18,415.4

<u>**PROBLEM-4**</u>

(i)Regress R&D Expenses on 'Sales' and 'Profits' and determine the residuals
(ii)Apply Durbin Watson test and give your conclusions
(iii)If you suspect that auto regressive error structure is of order p,use the Bruesh-Godfrey(B-G)test to detect autocorrelation .Decide the value of p before doing BG test
(iv)Is there multicollinearity in the data?

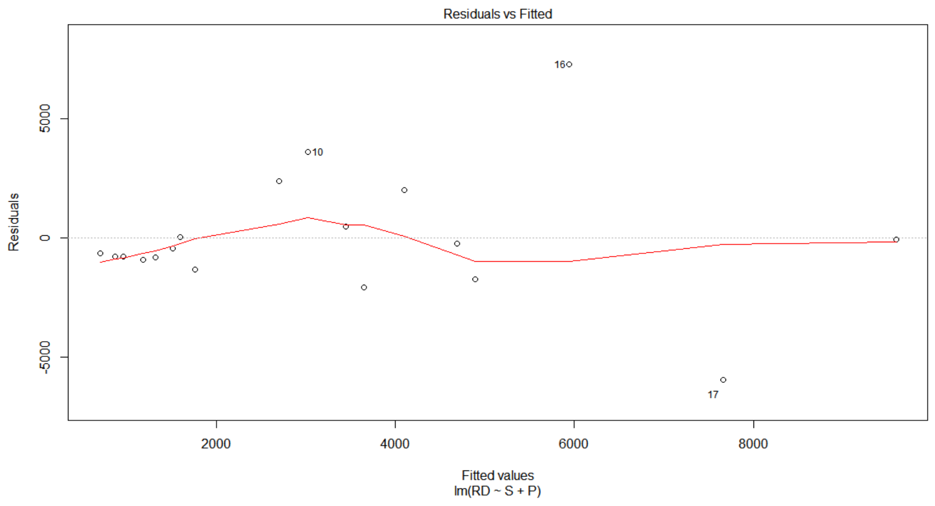
Solution :

(i) For the first problem we use the lm() function to regress the model and obtain the residuals.

```
RD<-
c(62.5,92.9,178.3,258.4,494.7,1083.0,1620.6,421.7,5089.2,6620.1,3918.6,1595.3,6107.5,4454,3163.8,
13210.7,1703,9528.2)
S=c(6375.3,11626.4,14655.1,21869.2,26408.3,32405.6,35107.7,40295.4,70761.6,80552,95294.0,101
314.1,116141.3,122315.7,141649.9,175025.8,230614.5,293543.0)
P<-
c(185.1,1569.5,276.8,2828.1,225.9,3751.9,2884.1,4645.7,5036.4,13869.9,4467.8,10278.9,8787.3,164
368.8,9761.4,19774.5,22626.6,18415.4)
IG<-rep(1:18)
df<-data.frame(IG,RD,S,P)
df[,3:4]=df[,3:4]/100
```

Now we fit the regression model with the given regressors.

```
fit1<-lm(RD~S+P,data = df)
df$residuals <- residuals(fit1)
# Printing the residuals
df$residuals
[1]  -639.78310  -774.59637  -779.12314  -927.66318  -824.43071  -429.81009
 [7]    26.82807 -1336.30089  2392.25383  3599.08470   467.84925 -2055.69347
[13]  2003.85719  -238.80557 -1727.65814  7266.03552 -5960.35350   -61.69040
#Plotting the residuals
plot(fit1,which=1)
```



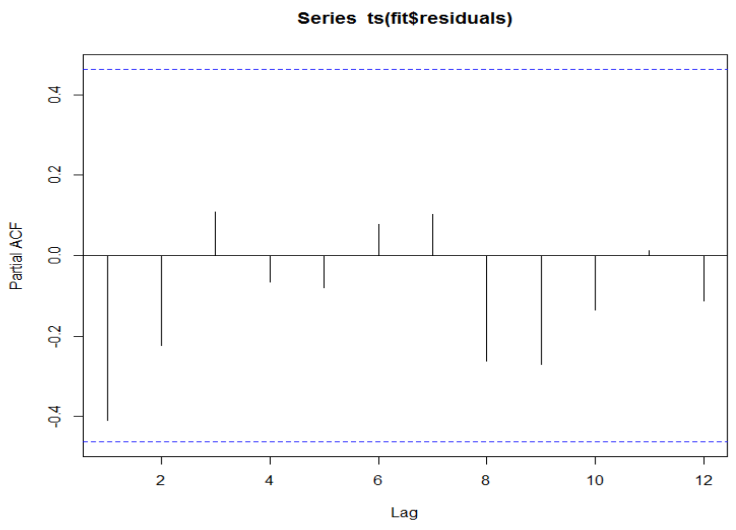
From the plot we observe that the errors are not randomly distributed. Thus we check for autocorrelation and heteroscedasticity

(ii) For checking serial correlation Durbin-Watson test is a very celebrated test , though this test has many assumptions one of which is that the autocorrelation of order 1.
To perform the Durbin Watson test we use the dwtest() function in lmtest() library . Here
H_o = no autocorrelation is present H_A = autocorrelation is present

```
library(lmtest)
dwtest(lm(RD~S+P,data=df))

        Durbin-Watson test
data:  lm(RD ~ S + P, data = df)
DW = 2.816, p-value = 0.947
alternative hypothesis: true autocorrelation is not equal than 0
```

From the p-value we observe that there is no autocorrelation.
Now in the Durbin-Watson test we assume that the errors follow a AR(1) process only and no other higher order processes.
To further investigate we plot the partial autocorrelation function of the errors.

```
acf(ts(fit$residuals), type = "partial")
```



Thus we observe that there might be a possibility of a second order autoregressive process as well . So we use the BG test to check this assumption . Here we take the order as 2.

```
bgtest(lm(RD~S+P),order = 2,data = df)

        Breusch-Godfrey test for serial correlation of order up to 2

data:  lm(RD ~ S + P)
LM test = 4.8596, df = 2, p-value = 0.08805
```

Since the p-value is > 0.05 , we fail to reject the null hypothesis, and conclude that there is no presence of serial correlation in the model.

(iv) To check for multicollinearity we generally check the Variance Inflation Factor , which takes into account the level of linear dependency of the said regressors on the other regressors.

```
 #multicollinearity
 df[,3] = scale(df[,3])
 df[,4] = scale(df[,4])
 model = lm(S~P,data = df)
Multiple R-squared:  0.07078,    Adjusted R-squared:  0.0127
vf1 = 1/(1-(.07078)^2)
```

```
[1] 1.005035
model = lm(P~S,data = df)
summary(model)
Multiple R-squared:  0.07078,    Adjusted R-squared:  0.0127
vf2 = 1/(1-(.07078)^2)
[1] 1.005035
```

Generally VIF value > 5, indicates some sort of multicollinearity. Here we observe that there is no presence of multicollinearity .

PROBLEM-5

Considering Sales as the explanatory Variable . Perform
(a)Park Test (b) GQ Test c=4 (c)Breusch-Pagan-Godfrey Test (d)White's test and (d) KB test

All the above tests are test of homoscedasticity , where the
Null hypothesis H_0 = homoscedasticity is present (all the error variances are equal)
And the alternative hypothesis is H_A = homoscedasticity is not present (the error variances are not equal)

(a)Park test : We first find the residuals and regress log of square of residuals on log of sales(explanatory variable) , if the parameter of log of sales is significant then we reject the null hypothesis.

```
#park test
m3 = lm(RD~S,data = df)
e_i_sq = m3$residuals^2
ln_ei = log(e_i_sq)
d_dash = data.frame(ln_ei,df[3])
m4 = lm(ln_ei~log(S),data = d_dash)
summary(m4)
Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)   9.8024     4.5698   2.145   0.0476 *
log(S)       0.5743     0.7108   0.808   0.4310
        ---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here we observe that the required parameter estimate is not significant, and hence we fail to reject the null hypothesis, thus homoscedasticity is present.

(b) **Goldfield Quandt GQ test for c=4 :** Here we split the data into two equal halves, one with larger values and one with smaller values, then we fit regression model for both and calculate the residual sum of squares for both , then we calculate the F-statistic to calculate whether the variances in two halves are equal or not .

```
df2=data.frame(RD,S)
df2[,2]=(df2[,2])/100
n=18
c=4
p=2
#We first sort in ascending order with respect to values of the explanatory variable
```

```
df2=df2[order(df2$S),]
#We omit c central observations and have two groups each having (n-c)/2 observations
df11=df2[1:7,]
df22=df2[12:18,]
#We then fit two regression to two groups separately and calculate the RSS in each group
lm1=lm(formula=RD~S,data=df11)
df11$residuals <- residuals(lm1)
p1<-matrix(df11$residuals,nrow=1,ncol=7)
RSS1<-p1%*%t(p1)
lm2=lm(formula=RD~S,data=df22)
df22$residuals <- residuals(lm2)
p2<-matrix(df22$residuals,nrow=1,ncol=7)
RSS2<-p2%*%t(p2)
d=(18-4-(2*2))/2
lmda_cal=as.integer(RSS2/RSS1)
lmda_tb=qf(.95, df1=d, df2=d)
if(lmda_cal>=lmda_tb)
{
print("heteroscedasticity is observed in the data.")
}else{
print("no trace for heteroscedasticity was observed in the data.")
}
```

Output-"heteroscedasticity is observed in the data."

Here we observe that the null hypothesis that homoscedasticity is present is rejected

c) Breusch- Pagan- Godfrey test :

```
        #B-P-G test
S_e_i = sum(e_i_sq)/(16)
p_i = m3$residuals/S_e_i
d_ds = data.frame(p_i,df[3])
m5 = lm(p_i~S,data = d_ds)
summary(m5)
Multiple R-squared:  1.358e-33
qchisq(0.95,1)
[1] 3.841459
```

Since the Multiple R_squared value is less than 3.841459, we fail to reject the null hypothesis , and thus homoscedasticity is present.

d) White’s Test: B-P-G test is sensitive to normality assumption, but White’s test does not require normality, here the square of the residuals are fitted with respect to the regressors and higher order terms of the regressors. Then the R-squared value for the model is determined and the test statistic is

nR^2, where n is the number of observations. And this follows a chi-squared distribution with error degree of freedom of the model fitted above.

```
#white test
d_dss = data.frame(e_i_sq, df[3],df[3]^2)
m6 = lm(e_i_sq~S+S.1,data = d_dss)
summary(m6)
Multiple R-squared:  0.2906
Test statistic t= 18*0.2906 = 5.2308
qchisq(.95,2)
```

```
5.991465
```

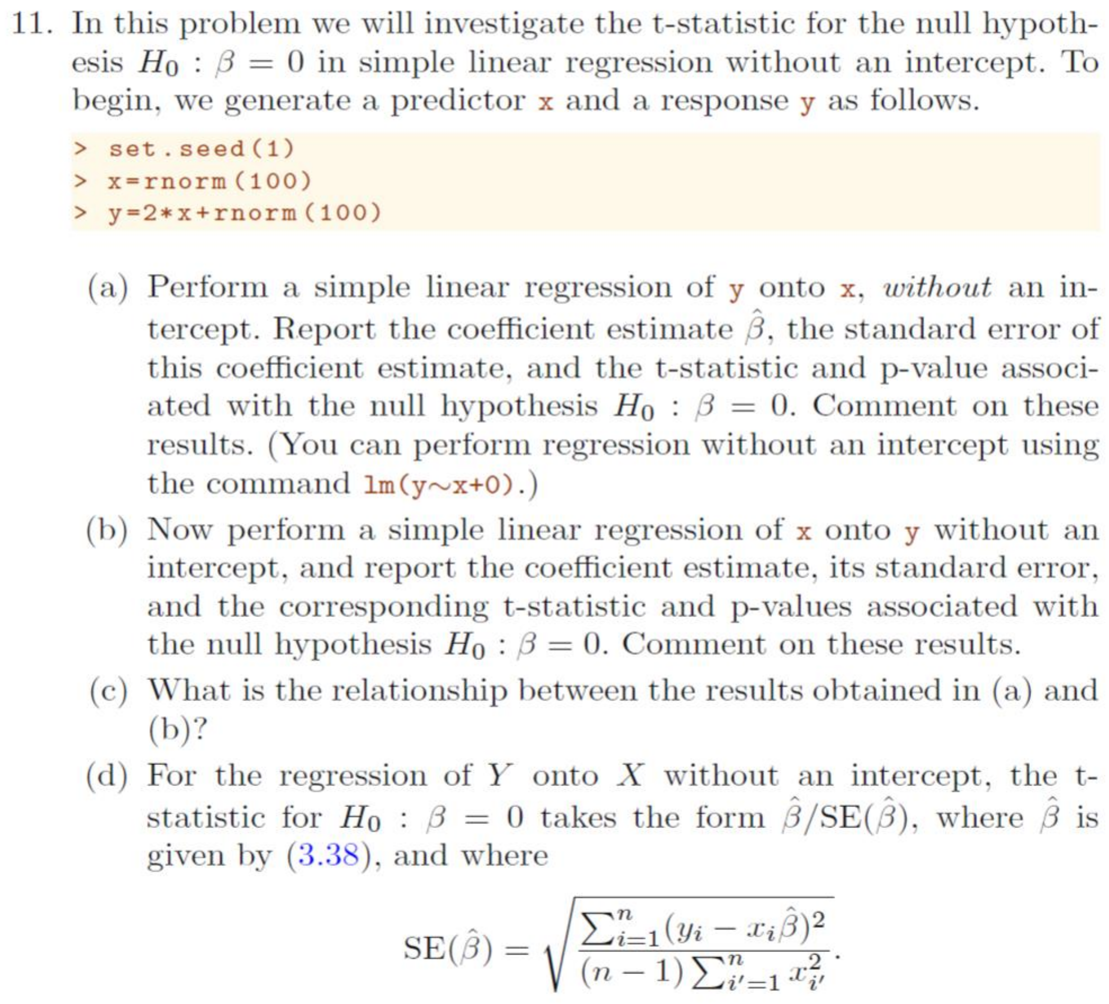
Since the test statistic value is less than 5.991465, we fail to reject the null hypothesis. Thus conclude that homoscedasticity is present.

e) Koenker-Bassett (KB Test) : Here the square of the residuals are fitted against the square of the fitted values. Thus, if the parameter of the fitted values is significant , we conclude to reject the null hypothesis

```
#koenkar-Bassett test
d_ds2 = data.frame(e_i_sq,Y_hat_sq = m3$fitted.values^2)
m7 = lm(e_i_sq~Y_hat_sq,data = d_ds2)
summary(m7)
Coefficients:
              Estimate     Std. Error    t value   Pr(>|t|)
(Intercept) 3.054e+06  3.917e+06    0.78    0.447
Y_hat_sq    2.267e-01  1.350e-01    1.68    0.112
```

Since p-value for Y_hat_sq is greater than 0.05, we fail to reject the null hypothesis and conclude that homoscedasticity is present.

Statistical Machine Learning



11. In this problem we will investigate the t-statistic for the null hypothesis $H_0 : \beta = 0$ in simple linear regression without an intercept. To begin, we generate a predictor `x` and a response `y` as follows.

```
> set.seed(1)
> x=rnorm(100)
> y=2*x+rnorm(100)
```

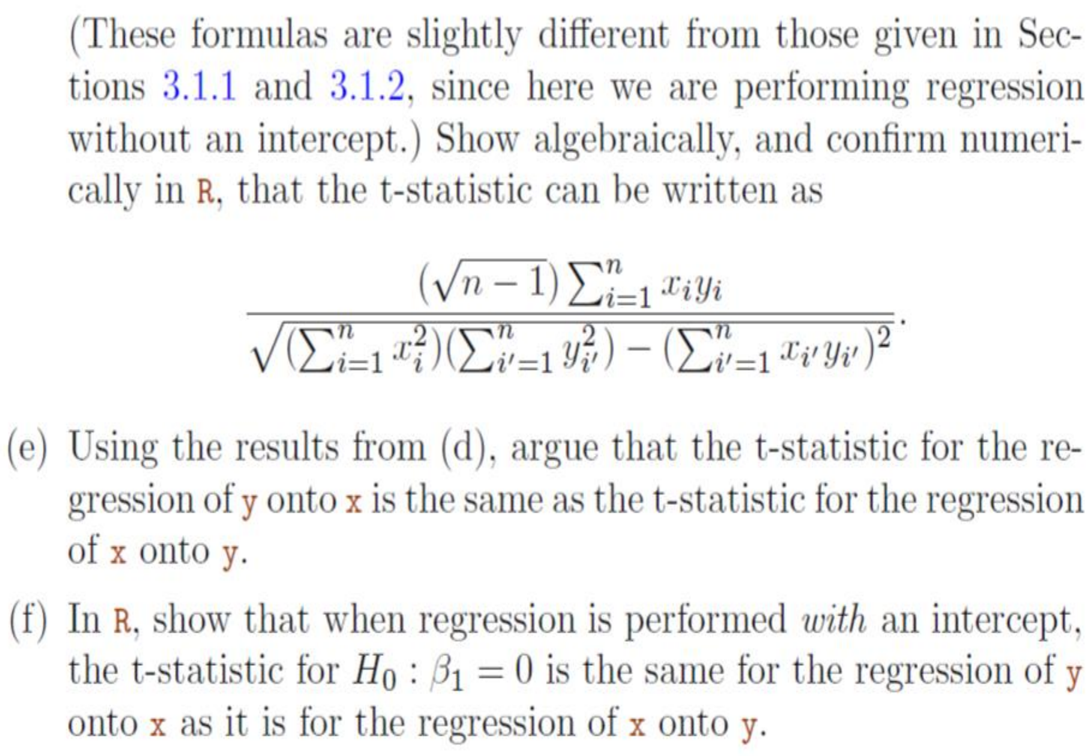
(a) Perform a simple linear regression of `y` onto `x`, *without* an intercept. Report the coefficient estimate $\hat{\beta}$, the standard error of this coefficient estimate, and the t-statistic and p-value associated with the null hypothesis $H_0 : \beta = 0$. Comment on these results. (You can perform regression without an intercept using the command `lm(y∼x+0)`.)

(b) Now perform a simple linear regression of `x` onto `y` without an intercept, and report the coefficient estimate, its standard error, and the corresponding t-statistic and p-values associated with the null hypothesis $H_0 : \beta = 0$. Comment on these results.

(c) What is the relationship between the results obtained in (a) and (b)?

(d) For the regression of Y onto X without an intercept, the t-statistic for $H_0 : \beta = 0$ takes the form $\hat{\beta}/\text{SE}(\hat{\beta})$, where $\hat{\beta}$ is given by (3.38), and where

$$\text{SE}(\hat{\beta}) = \sqrt{\frac{\sum_{i=1}^{n}(y_i - x_i\hat{\beta})^2}{(n-1)\sum_{i'=1}^{n} x_{i'}^2}}.$$


(These formulas are slightly different from those given in Sections 3.1.1 and 3.1.2, since here we are performing regression without an intercept.) Show algebraically, and confirm numerically in R, that the t-statistic can be written as

$$\frac{(\sqrt{n-1})\sum_{i=1}^{n} x_i y_i}{\sqrt{(\sum_{i=1}^{n} x_i^2)(\sum_{i'=1}^{n} y_{i'}^2) - (\sum_{i'=1}^{n} x_{i'} y_{i'})^2}}.$$

(e) Using the results from (d), argue that the t-statistic for the regression of y onto x is the same as the t-statistic for the regression of x onto y.

(f) In R, show that when regression is performed *with* an intercept, the t-statistic for $H_0 : \beta_1 = 0$ is the same for the regression of y onto x as it is for the regression of x onto y.

SOLUTION-

```
> set.seed(1)
> x <- rnorm(100)
> y <- 2 * x + rnorm(100)
```

a)

```
> fit5 <- lm(y ~ x + 0)
> summary(fit5)

Call:
lm(formula = y ~ x + 0)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.9154 -0.6472 -0.1771  0.5056  2.3109 

Coefficients:
  Estimate Std. Error t value Pr(>|t|)    
x   1.9939     0.1065   18.73   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

b)

```
> fit6 <- lm(x ~ y + 0)
> summary(fit6)

Call:
lm(formula = x ~ y + 0)
```

```
Residuals:
    Min      1Q  Median      3Q     Max 
-0.8699 -0.2368  0.1030  0.2858  0.8938 

Coefficients:
  Estimate Std. Error t value Pr(>|t|)    
y  0.39111    0.02089   18.73   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4246 on 99 degrees of freedom
Multiple R-squared:  0.7798,	Adjusted R-squared:  0.7776 
F-statistic: 350.7 on 1 and 99 DF,  p-value: < 2.2e-16
```

c)

We obtain the same value for the t-statistic and consequently the same value for the corresponding p-value. Both results in (a) and (b) reflect the same line created in (a). In other words, $y=2x+\varepsilon$$y=2x+\varepsilon$ could also be written $x=0.5(y-\varepsilon)$$x=0.5(y-\varepsilon)$

d)

```
> n <- length(x)
> t <- sqrt(n - 1)*(x %*% y)/sqrt(sum(x^2) * sum(y^2) - (x %*% y)^2)
> as.numeric(t)
[1] 18.72593
```

We may see that the t above is exactly the t-statistic given in the summary of “fit6”.

e)

It is easy to see that if we replace $x_i$xi by $y_i$yi in the formula for the t-statistic, the result would be the same.

f)

```
> fit7 <- lm(y ~ x)
> summary(fit7)

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.8768 -0.6138 -0.1395  0.5394  2.3462 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -0.03769    0.09699  -0.389    0.698    
x            1.99894    0.10773  18.556   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.9628 on 98 degrees of freedom
Multiple R-squared:  0.7784,	Adjusted R-squared:  0.7762 
F-statistic: 344.3 on 1 and 98 DF,  p-value: < 2.2e-16

> 
> fit8 <- lm(x ~ y)
> summary(fit8)

Call:
lm(formula = x ~ y)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.90848 -0.28101  0.06274  0.24570  0.85736 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.03880    0.04266    0.91    0.365    
y            0.38942    0.02099   18.56   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4249 on 98 degrees of freedom
Multiple R-squared:  0.7784,  Adjusted R-squared:  0.7762 
F-statistic: 344.3 on 1 and 98 DF,  p-value: < 2.2e-16
```

It is again easy to see that the t-statistic for “fit7” and “fit8” are both equal to 18.5555993

```
Residual standard error: 0.9586 on 99 degrees of freedom
Multiple R-squared:  0.7798,  Adjusted R-squared:  0.7776 
F-statistic: 350.7 on 1 and 99 DF,  p-value: < 2.2e-16
```

Consider the Smarket data from IRLS library in R. Dataset consists of percentage returns for the S&P 500 stock return over the period of 1250 days from the beginning of 2001 until the end of 2005. For each day, percentage return for each of the five previous days has been recorded. These variables are lag1, lag2, lag3, lag4 and lag5. Volume (i.e. the number of shares traded on the previous day, in billions), Today (the percentage return on the date) and direction (whether the market was up or down at that date) were also recorded. Based on the above dataset.

1. 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. Use glm() function to check your results.
2. Perform LDA on the Smarket data. Use Lag1 and Lag2 to predict Direction. In R, we fit a LDA model using the lda() function, which is part of the MASS library. Use this function to verify your result.

Solution :

1. **Logistic regression** : In logistic regression the response variable is binary or categorical, hence we use logistic link function and our model is ,

   p(xi) = e0+1x1i+....+pxpi1+ e0+1x1i+.....+pxpi,

   where $p(x_i)$ is the probability of occurrence of i th event, and x_i’s are the regressors and 's are the estimates of the parameters.

   Likelihood function of the Logistic Regression

   Because logistic regression predicts probabilities, rather than just classes, we can fit it using likelihood. For each training data-point, we have a vector of features, x_i , and an observed class, y_i . The probability of that class was either p, if y_i= 1, or 1 − p, if y_i= 0. The likelihood is then

   L(0,1,.....,p) = i=nnp(xi)yi(1-p(xi))1-yi (1)

   Taking logarithm on both sides the above equation finally reduces to, l(0,1,.....,p) = i = 1n-log(1+ e0+....+pxpi)+i = 1nyi(0+....+pxpi)   (2)

In our given problem , we are to fit a logistic regression model. The dataset contains two categories in the response variable “Direction”. First we start by loading the dataset and indexing the required columns

[R code]

```
> library(ISLR)
> data = Smarket
> data = data[c(2,3,4,5,6,7,9)]
> summary(data)


#    Lag1                      Lag2                    Lag3                    Lag4
Lag5
# Min.   :-4.922000     Min.   :-4.922000     Min.   :-4.922000       Min.   :-
4.922000     Min.   :-4.92200
#1st Qu.:-0.639500    1st Qu.:-0.639500   1st Qu.:-0.640000     1st Qu.:-
0.640000   1st Qu.:-0.64000
# Median : 0.039000   Median : 0.039000  Median : 0.038500   Median :
0.038500   Median : 0.03850
# Mean   : 0.003834    Mean   : 0.003919  Mean   : 0.001716    Mean   :
0.001636    Mean   : 0.00561
# 3rd Qu.: 0.596750    3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.:
0.596750    3rd Qu.: 0.59700
# Max.   : 5.733000     Max.   : 5.733000   Max.   : 5.733000     Max.   :
5.733000     Max.   : 5.73300
#    Volume              Direction
 #Min.   :0.3561       Down:602
# 1st Qu.:1.2574     Up  :648
# Median :1.4229
# Mean   :1.4783
# 3rd Qu.:1.6417
# Max.   :3.1525
```

Building Logistic Classifier using nls function in R.

```
> #Converting the factors into binary values Up = 1 , Down = 0
> p = ifelse(Direction == "Up",1,0)
> d = as.numeric(p)
> m = d
> #Obtaining the estimates of the parameters solving the non-linear least squares
> estimate = nls(d~(-(log(1+
    exp(beta0*rep(1,1250)+Lag1*beta1+Lag2*beta2+Lag3*beta3+Lag4*beta4+Lag5*beta5+
    Volume*beta6))))+sum(d*(beta0+Lag1*beta1+Lag2*beta2+Lag3*beta3+Lag4*beta4+Lag
    5*beta5+Volume*beta6)),start = list(beta0 = 1.46,beta1 =
    0,beta2=0,beta3=0,beta4=0,beta5=0,beta6 = 0))
> summary(estimate)

      # Formula: d ~ (-(log(1 + exp(beta0 * rep(1, 1250) + Lag1 * beta1 + Lag2 *
  # beta2 + Lag3 * beta3 + Lag4 * beta4 + Lag5 * beta5 + Volume *
```

```
   # beta6)))) + sum(d * (beta0 + Lag1 * beta1 + Lag2 * beta2 +
   # Lag3 * beta3 + Lag4 * beta4 + Lag5 * beta5 + Volume * beta6))

   #      Parameters:
   #           Estimate Std. Error t value Pr(>|t|)
         beta0  0.107797   0.118196   0.912    0.362
         beta1  0.036744   0.024847   1.479    0.139
         beta2  0.021777   0.024841   0.877    0.381
         beta3 -0.006251   0.024793  -0.252    0.801
              beta4 -0.004062   0.024746  -0.164    0.870
         beta5 -0.003539   0.024595  -0.144    0.886
         beta6 -0.069878   0.079489  -0.879    0.380

  #    Residual standard error: 0.5003 on 1243 degrees of freedom

# Number of iterations to convergence: 4
# Achieved convergence tolerance: 2.342e-06
```

Note : the initial values for the estimation of the parameters are given based on the linear model fit of the above data using lm()

```
> #predicting the results using built-in logistic classifier
> p = predict(estimate, data[-7])
> p = ifelse(p>0.5,1,0)
> #fitting model using glm function and predicting the results
> glm.fits=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume ,
+             data=Smarket ,family =binomial )
> p_dash = predict(glm.fits, type = "response")
> g = ifelse(p_dash>0.5,1,0)
> #checking the results from both the classifiers.
> table(p)

# p
#   0   1
# 286 964

> table(g)

# g
#    0   1
#  286 964
```

From the above results we conclude that our builtin classifier and glm() function in R classifies the responses with the same accuracy.

2. **Linear Discriminant Analysis** : In case of p(>1) features , in linear discriminant analysis it is assumed that the observations of the kth class are drawn from the multivariate normal distribution. Suppose X be the matrix of features, then X follows N(,), where is the covariance matrix common to the k classes.

Thus is Linear Discriminant Analysis, the observation X = x is assigned to the class for which, k(x) = xT-1k-12Tk-1k + log k, is the largest .

where, k is the men vector for the k-th class, and is the common covariance matrix for all the classes, kis the proportion of the k-th class.

In our given problem , we are to fit a LDA model. The dataset contains two categories in the response variable “Direction”. Here as predictor we take only two variables ‘Lag1’ and ‘Lag2’ [R Code]

```
> #Loading dataset
> library(ISLR)
> data = Smarket
> delta_up = seq(0)
> delta_down = seq(0)
> my_lda = seq(0)

> #creating mean vectors and sigma matrix for all classes
> library(tidyverse)
> data_up = data%>%
+   filter(Direction == "Up")
> data_down = data%>%
+   filter(Direction == "Down")
> mew_up = colMeans(data_up[2:3])
> mew_down = colMeans(data_down[2:3])
> up.group = data_up
> down.group = data_down
> cov.up = cov(up.group[,c(2,3)])
> cov.down = cov(down.group[,c(2,3)])
> nUp = nrow(up.group)
> nDown = nrow(down.group)
> n = nUp + nDown; K = 2

> #Creating sigma matrix common to all classes
> Sigma <- 1/(n - K) * (cov.up * (nUp - 1) + cov.down * (nDown - 1))
> SigmaInv <- solve(Sigma)
> for(i in 1:n){
+   delta_up[i] = as.matrix(data[i,2:3])%*%SigmaInv%*%as.matrix(mew_up) -
as.matrix(t(mew_up))%*%SigmaInv%*%as.matrix(mew_up)/2 + log(nUp/n)}
> for(j in 1:n){
+   delta_down[j] = as.matrix(data[j,2:3])%*%SigmaInv%*%as.matrix(mew_down) -
as.matrix(t(mew_down))%*%SigmaInv%*%as.matrix(mew_down)/2 + log(nDown/n)
+ }

> #Constructing the lda classifier
> for(i in 1:n){
+   if(delta_up[i]>delta_down[i]){
+     my_lda[i] = "Up"
+   }else{
+     my_lda[i] = "Down"
+   }
+ }

> #creating the table of observe the performance
> my_lda = as.factor(my_lda)
> table(my_lda)
```

```
my_lda
Down   Up
 216 1034
```

```
> #fitting lda model from MASS package to verify our results
> library(MASS)
> model = lda(Direction~Lag1+Lag2, data = data[c(2,3,9)])
> p = predict(model, data,interval = "prediction")
> table(p$class)
```

```
Down   Up
 216 1034
```

Result : from both the results we observe that our LDA classifier performs with the same accuracy as that of the lda() function in MASS package.

Practical 6

Perform PCA on the USArrests data set, which is part of the base R package. Obtain the PCs using correlation as well as covariance matrix. You can use the prcomp() function to verify your results.

Solution :
Principal Component Analysis is a dimension reduction technique. Here we create new components or features which are completely independent among themselves. The new features try to explain the complete variance of the old features , by orthogonal projections.
Suppose we have n features in our dataset, and through PCA we create k new features such that variance of those k features is almost equal to the variance of the n features.
The principal components can be formed from the covariance matrix of the old features as well as the correlation matrix , which is nothing but the covariance matrix of the scaled feature(standardized). We will show both the methods, and the values of the features and the components are completely different.

So in this problem we are given the dataset USArrests. And we have to apply PCA on this.

[R
Code]

```
> #Importing dataset
> data = USArrests

> #PCA using covariance matrix
> S = cov(data)
> p = eigen(S)
> m = colMeans(data)
> d = data - m
> f = p$vectors
> y = as.matrix(d )%*% as.matrix(f)
> ss = sqrt(p$values)
> #Verifying the results using procomp function
> p1 = prcomp(data, scale = FALSE)
> ss1 = p1$sdev
> ss
```
[1] 83.732400 14.212402 6.489426 2.482790
```
> ss1
```
[1] 83.732400 14.212402 6.489426 2.482790

Hence we observe that the standard deviation our PCAs and from prcomp function are the same.

```
> f
          [,1]        [,2]        [,3]        [,4]
[1,] -0.04170432  0.04482166  0.07989066  0.99492173
[2,] -0.99522128  0.05876003 -0.06756974 -0.03893830
[3,] -0.04633575 -0.97685748 -0.20054629  0.05816914
[4,] -0.07515550 -0.20071807  0.97408059 -0.07232502
> p1
Standard deviations (1, .., p=4):
[1] 83.732400 14.212402  6.489426  2.482790

Rotation (n x k) = (4 x 4):
                PC1         PC2         PC3         PC4
Murder   0.04170432 -0.04482166  0.07989066 -0.99492173
Assault  0.99522128 -0.05876003 -0.06756974  0.03893830
UrbanPop 0.04633575  0.97685748 -0.20054629 -0.05816914
Rape     0.07515550  0.20071807  0.97408059  0.07232502
```

And from the above results we can observe that the PCAs are same is magnitude , only they differ in sign

```
> #Importing dataset
> data = USArrests
> #PCA using correlation matrix
> data = scale(data)
> S = cov(data)
> p = eigen(S)
> m = colMeans(data)
> d = data - m
> f = p$vectors
> y = as.matrix(d )%*% as.matrix(f)
> ss = sqrt(p$values)
> ss
```
[1] 1.5748783 0.9948694 0.5971291 0.4164494
```
> #Verifying the results using procomp function (scaled dataset)
> p1 = prcomp(data, scale = TRUE)
> ss1 = p1$sdev
> ss1
```
[1] 1.5748783 0.9948694 0.5971291 0.4164494

Hence we observe that the PCA from the correlation matrix is same the obtained from the prcomp function with the scaled dataset.

```
> f
           [,1]      [,2]       [,3]        [,4]
[1,] -0.5358995  0.4181809 -0.3412327  0.64922780
[2,] -0.5831836  0.1879856 -0.2681484 -0.74340748
```

```
[3,] -0.2781909 -0.8728062 -0.3780158  0.13387773
[4,] -0.5434321 -0.1673186  0.8177779  0.08902432
> p1
Standard deviations (1, .., p=4):
[1] 1.5748783 0.9948694 0.5971291 0.4164494

Rotation (n x k) = (4 x 4):
                PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

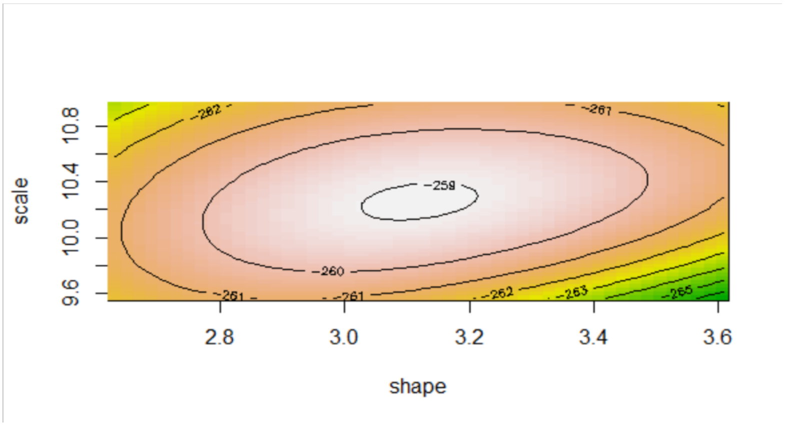
Hence we observe that the PCs are same .

Life Time data Aanalysis

1) Generate 100 observations from Weibull distribution with shape parameter 3 and scale parameter 10. Hence obtain the ML estimation of its parameters. Also draw the two-dimensional likelihood plot of Weibull model for the given dataset. Finally obtain the ML estimate of Mean failure time and compare it with sample mean.

SOLUTION-

```
  #Generating 100 observations from weibull with shape parameter 3 and scale parameter 10

> sample<-rweibull(n=100,shape=3,scale=10)
> sample
  [1] 12.081386  8.912287 12.172307  5.401799 10.045019  8.606428 11.786153
  [8] 11.663773  6.150583  7.637962 15.855920 10.440038  4.891701  9.824133
 [15] 10.483931  6.677550  8.133340 10.572723  7.068911 10.063666 14.270472
 [22]  7.341141 12.004034  9.480990  7.340955 11.234654  3.670719 11.121310
 [29]  2.538356  6.075332  6.020747  9.589156  8.023092  6.532884 10.604761
 [36]  6.896062  2.708143 10.787908  5.353617 16.763594 10.568132  5.899232
 [43]  7.970929  9.281217  3.446171  2.190044 11.174496  8.430260 11.673488
 [50] 10.097250 12.348516  8.120097  7.617115  7.384174  8.883584 14.866521
 [57] 11.286997 12.610949  8.578275 11.363273 13.329543  8.765707  7.559577
 [64]  6.839604 10.880193  9.763110  4.981761  7.092753  8.908237 10.920958
 [71]  3.036891 10.062169  6.315620 11.964483  9.554476  5.555793  9.026584
 [78]  8.361361  5.858065 12.220806  5.013457 17.750392  8.732087  3.744102
 [85]  9.819653  8.504815 12.579613 13.345451 10.257108 12.127353  2.414382
 [92] 15.580475 14.489353 10.816854 12.665872 10.533775 13.295094  8.006125
 [99]  8.251133  8.779933
> #Maximum Likelihood estimation of parameters
> b_est<-function(x_i,n,Beta)
+ {
+   u=sum(log(x_i))
+   v=sum(x_i^Beta)
+   w=sum((x_i^Beta )*log(x_i))
+   s=sum((x_i^Beta)*(log(x_i))^2)
+   f1=(1/Beta)+(u/n)-(w/v)
+   f2=(-1/Beta^2)+(w^2/v^2)-(s/v)
+   f=Beta-(f1/f2)
+   return(f)
+ }
> r1=b_est(sample,100,3)
> r2=b_est(sample,100,r1)
> while(abs(r1-r2)>0.001)
+ {
+   r1=r2
+   r2=b_est(sample,100,r1)
+ }
> b=r1
> a=(sum(sample^b)/100)^(1/b)
>
> a
[1] 10.25882
> b
[1] 3.119245

> library(fitdistrplus)
> fite <- fitdist(sample, "weibull")
> llplot(fite)
```



```
> Sample_mean=mean(sample)
>
> MTTF=a*gamma(1+1/b)
> Sample_mean
[1] 9.18323
> MTTF
[1] 9.177161

Result- Parametr Estimates  a = 10.25882, b =3.119245
        sample mean = 9.18323
        MTTF = 9.177161 i.e sample mean and MTTF are approximately equal
```

2) The recorded death times of 15 patients were 7.35, 8.69, 8.80, 9.63, 9.63, 9.89, 9.98, 10.24, 10.36, 10.37, 10.48, 11.33, 11.39, 12.02 and 13.12 days, 10 patients whose are alive were removed from the test at 20 days. Suppose recorded time follows Weibull distribution, then

a) Find maximum likelihood estimates of parameter.

b) Using estimates of part 1 draw survival and hazard rate curve.

c) Comment on behaviour of hazard rate.

SOLUTION-

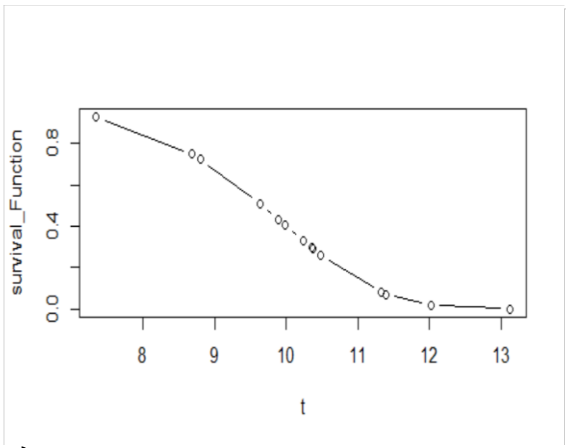
a)

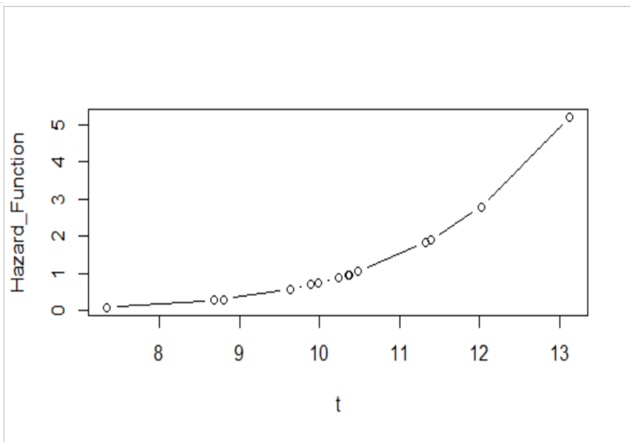
```
> x=c(7.35, 8.69, 8.80, 9.63, 9.63, 9.89, 9.98, 10.24, 10.36, 10.37, 10.48
, 11.33, 11.39, 12.02, 13.12)

> fn<-function(theta)
+ {
+   n=25
+   c=20
+   s=15
+   L=s*(log(theta[2])-log(theta[1]))+((theta[2]-1)*sum(log(x)))-(sum(x^th
eta[2])/theta[1]^theta[2])-(10*(c/theta[1])^theta[2])
+   return(L)
+ }
>
> nlm(fn,theta<-c(10.8,8))
```

```
$estimate
[1] 10.11 8.15
```

b)

```
> a = 10.11
> b = 8.15
>
> survival_Function = exp(-(x/a)^b)
> Hazard_Function = (b/a^b)*x^(b-1)
>
>
> plot(t,survival_Function,type = 'b')
> plot(t,Hazard_Function,type = 'b')
```





c)

In this hazard plot, the hazard rate for variable decreases in the early period, then levels up , and starts increasing i.e grater chance of dying as time increases