STAT_8320_HW1

JAYADITYA NATH

2024-02-09

PROBLEM 1.

Here, the linear regression model under consideration is :

$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$

The design matrix is of full column rank and $\boldsymbol{\epsilon} \sim (0, \sigma^2\mathbf{I})$

Simplifying the right-hand side of the mentioned decomposition,

$\|Y - X\hat{\beta}\|^2 + \|X\hat{\beta}\|^2$

$= (Y - X\hat{\beta})'(Y - X\hat{\beta}) + (X\hat{\beta})'(X\hat{\beta})$

$= Y'Y - \hat{\beta}'X'Y - Y'X\hat{\beta} + \hat{\beta}'X'X\hat{\beta} + \hat{\beta}'X'X\hat{\beta}$

$= Y'Y - \hat{\beta}'X'Y - Y'X\hat{\beta} + 2\hat{\beta}'X'X\hat{\beta}$

Plugging in the value of $\hat{\beta}$,

$= Y'Y - ((X'X)^{-1}X'Y)'X'Y - Y'X(X'X)^{-1}X'Y + 2(X'X)^{-1}X'Y)'X'X(X'X)^{-1}X'Y)$

$= Y'Y - 2Y'X(X'X)^{-1}X'Y + 2Y'X(X'X)^{-1}X'Y$

$= Y'Y$

$= Y'Y - \hat{\beta}'X'Y + \hat{\beta}'X'Y$

Thus, $Y'Y - \hat{\beta}'X'Y$ comprises of SSE and $\hat{\beta}'X'Y$ comprises of SSM.

PROBLEM 2.

Let $\mathbf{X} = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} \sim N(\boldsymbol{\mu} = \begin{pmatrix} -1 \\ 2 \\ 1 \end{pmatrix}, \Sigma = \begin{pmatrix} 3 & 2 & 0 \\ 2 & 4 & 1 \\ 0 & 1 & 3 \end{pmatrix})$

Part (a) Let, $\boldsymbol{X_1}, \boldsymbol{X_2}, ..., \boldsymbol{X_{50}}$ are 50 random vector samples, each of size 3X1 from the multivariate normal distribution mentioned above.

Now, using the property of large sample and using Central Limit Theorem, The sample mean $\bar{\boldsymbol{X}} \sim N(\boldsymbol{\mu}, \Sigma/n) \equiv N(\boldsymbol{\mu} = \begin{pmatrix} -1 \\ 2 \\ 1 \end{pmatrix}, \Sigma = \begin{pmatrix} 3/50 & 1/25 & 0 \\ 1/25 & 2/25 & 1/50 \\ 0 & 1/50 & 3/50 \end{pmatrix})$

Part (b)

Let, Z = $X_1 + 3X_2 + 2X_3$ From part (a), we know that :

$\mathbf{X} = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} \sim N(\boldsymbol{\mu} = \begin{pmatrix} -1 \\ 2 \\ 1 \end{pmatrix}, \Sigma = \begin{pmatrix} 3 & 2 & 0 \\ 2 & 4 & 1 \\ 0 & 1 & 3 \end{pmatrix})$

Now, $\boldsymbol{Z} = \boldsymbol{a}^{'}\boldsymbol{X}$, where $\boldsymbol{a} = \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix}$

So, E(Z) = $\boldsymbol{a}^{'}E(\boldsymbol{X}) = \begin{pmatrix} 1 & 3 & 2 \end{pmatrix} \begin{pmatrix} -1 \\ 2 \\ 1 \end{pmatrix} = 7$

Again, V(Z) = V($\boldsymbol{a}^{'}\boldsymbol{X}$) $= \boldsymbol{a}^{'}\Sigma\boldsymbol{a} = \begin{pmatrix} 1 & 3 & 2 \end{pmatrix} \begin{pmatrix} 3 & 2 & 0 \\ 2 & 4 & 1 \\ 0 & 1 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix} = 75$

So, Z $\sim$ N(7,$(\sqrt{75})^2$)

PROBLEM 3.

Loading the data in R :

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
data_3_ini = read.table("C:\\Users\\Jayaditya Nath\\Downloads\\S24hw1pr3.txt",sep = ",")
data_3 = as.data.frame(data_3_ini[2:21,] %>% rename("X"="V1","Y"="V2"))
data_3
```

```
##       X    Y
## 2  0.71 3.71
## 3  1.85 4.64
## 4   2.1 4.95
## 5  2.14 4.88
## 6  2.73  5.4
## 7  3.78 6.16
## 8  3.86 6.14
## 9   3.9 6.23
## 10 5.03 6.88
## 11 5.77 7.03
## 12 6.33  7.2
## 13 6.64 7.29
## 14  6.9 7.32
## 15  7.2 7.31
## 16 7.72 7.51
## 17  7.8 7.49
## 18 8.99 7.62
## 19 9.09 7.49
## 20 9.45 7.64
## 21 9.92  7.6
```

Part (a)

```
model_3 = function(theta_vec,x)
{
  theta_0 = theta_vec[1]
  theta_1 = theta_vec[2]
  theta_2 = theta_vec[3]
  y = (theta_0)/(1 + exp(-theta_1*(x-theta_2)))
  return(y)
}

theta_0_candidate = c(7.30, 7.41, 7.52, 7.63, 7.74, 7.86, 7.97, 8.08, 8.19, 8.30)
theta_1_candidate = c(0.30, 0.34, 0.39, 0.43, 0.48, 0.52, 0.57, 0.61, 0.66, 0.70)
theta_2_candidate = c(0.80, 0.84, 0.89, 0.93, 0.98, 1.02, 1.07, 1.11, 1.16, 1.20)

#Grid Search Algorithm

best_est_sse = 1000000
for (theta_0 in theta_0_candidate) {
  for (theta_1 in theta_1_candidate) {
    for (theta_2 in theta_2_candidate) {
      sse_est = sum((as.double(data_3$Y) - model_3(c(theta_0,theta_1,theta_2),as.double(data_3$X)))^2)

      if (sse_est < best_est_sse) {
        best_est_sse <- sse_est
        theta_curl = c(theta_0, theta_1, theta_2)
      }
    }
  }
}

print(paste0("The best estimates of theta are : ",theta_curl[1],",",theta_curl[2]," and ",theta_curl[3]
```

```
## [1] "The best estimates of theta are : 7.74,0.48 and 0.98 respectively."
```

```
print(paste0("The SSE of the best estimates of theta is : ",best_est_sse))
```

```
## [1] "The SSE of the best estimates of theta is : 0.0521038293577142"
```

Part (b)

```
print("The derivates of the mean function with respect to theta_0, theta_1 and theta_2 are : ")
```

```
## [1] "The derivates of the mean function with respect to theta_0, theta_1 and theta_2 are : "
```

```
D(expression((theta_0)/(1 + exp(-theta_1*(x-theta_2)))),'theta_0')
```

```
## 1/(1 + exp(-theta_1 * (x - theta_2)))
```

```
D(expression((theta_0)/(1 + exp(-theta_1*(x-theta_2)))),'theta_1')
```

```
## (theta_0) * (exp(-theta_1 * (x - theta_2)) * (x - theta_2))/(1 +
##     exp(-theta_1 * (x - theta_2)))^2
```

```
D(expression((theta_0)/(1 + exp(-theta_1*(x-theta_2)))),'theta_2')
```

```
## -((theta_0) * (exp(-theta_1 * (x - theta_2)) * theta_1)/(1 +
##     exp(-theta_1 * (x - theta_2)))^2)
```

Part (c)

```
#Gauss-Newton Algorithm
library(pracma)
Y = as.matrix(as.double(data_3$Y),nrow=20)
theta_curl_gauss_new = theta_curl

f_theta_gauss_new = as.matrix(model_3(theta_curl_gauss_new,as.double(data_3$X)),nrow=20)
col_1_gauss_new = eval(D(expression((theta_0)/(1 + exp(-theta_1*(x-theta_2)))),'theta_0'),list(theta_1=
col_2_gauss_new = eval(D(expression((theta_0)/(1 + exp(-theta_1*(x-theta_2)))),'theta_1'),list(theta_1=
col_3_gauss_new = eval(D(expression((theta_0)/(1 + exp(-theta_1*(x-theta_2)))),'theta_2'),list(theta_0=
F_theta_gauss_new = cbind(col_1_gauss_new,col_2_gauss_new,col_3_gauss_new)

delta_gauss_new = pinv(t(F_theta_gauss_new) %*% F_theta_gauss_new) %*% t(F_theta_gauss_new) %*% (Y - f_
theta_curl_gauss_new = as.matrix(theta_curl_gauss_new) + delta_gauss_new

if(max(abs(delta_gauss_new))<0.0001)
  {
    break
  }

print(paste0("The final estimates of theta are : ",theta_curl_gauss_new[1],",",theta_curl_gauss_new[2],
```

```
## [1] "The final estimates of theta are : 7.74744547810942,0.47702871760771 and 0.953838037355924 resp
```

```
print("The Jacobian matrix is : ")
```

```
## [1] "The Jacobian matrix is : "
```

```
F_theta_gauss_new
```

```
##       col_1_gauss_new col_2_gauss_new col_3_gauss_new
##  [1,]       0.4676453      -0.5579041     -0.92491082
##  [2,]       0.6029088       1.7287780     -0.88945520
##  [3,]       0.6312539       2.1638523     -0.86479603
##  [4,]       0.6357118       2.2296745     -0.86037459
##  [5,]       0.6984652       3.0591329     -0.78246406
##  [6,]       0.7931470       3.8128672     -0.60953374
##  [7,]       0.7993762       3.8335779     -0.59582114
##  [8,]       0.8024377       3.8421685     -0.58897608
```

```
##  [9,]       0.8747909       3.6819101     -0.40693240
## [10,]       0.9088108       3.2948121     -0.30789260
## [11,]       0.9287735       2.9375365     -0.24577268
## [12,]       0.9380107       2.7316112     -0.21602627
## [13,]       0.9448828       2.5589661     -0.19348483
## [14,]       0.9519193       2.3628664     -0.17004070
## [15,]       0.9621376       2.0379001     -0.13534029
## [16,]       0.9635119       1.9900787     -0.13061408
## [17,]       0.9790573       1.3631743     -0.07617685
## [18,]       0.9800192       1.3180945     -0.07274946
## [19,]       0.9831366       1.1655249     -0.06159454
## [20,]       0.9864964       0.9884668     -0.04949128
```

```r
print("The function value vector is : ")
```

```
## [1] "The function value vector is : "
```

```r
f_theta_gauss_new
```

```
##           [,1]
##  [1,] 3.619574
##  [2,] 4.666514
##  [3,] 4.885905
##  [4,] 4.920409
##  [5,] 5.406121
##  [6,] 6.138958
##  [7,] 6.187172
##  [8,] 6.210868
##  [9,] 6.770882
## [10,] 7.034195
## [11,] 7.188707
## [12,] 7.260203
## [13,] 7.313393
## [14,] 7.367856
## [15,] 7.446945
## [16,] 7.457582
## [17,] 7.577903
## [18,] 7.585349
## [19,] 7.609477
## [20,] 7.635482
```

```r
print("The inverse of cross-product of the Jacobian matrix is : ")
```

```
## [1] "The inverse of cross-product of the Jacobian matrix is : "
```

```r
pinv(t(F_theta_gauss_new) %*% F_theta_gauss_new)
```

```
##             [,1]        [,2]       [,3]
## [1,]  0.35825514 -0.10466349 0.01494281
## [2,] -0.10466349  0.04718865 0.05715444
## [3,]  0.01494281  0.05715444 0.41689409
```

```r
print(paste0("The increment vector comprises of : ",delta_gauss_new[1],",",delta_gauss_new[2]," and ",de
```

```
## [1] "The increment vector comprises of : 0.00744547810942505,-0.00297128239228961 and -0.0261619626
```

Part (d)

```r
#Steep Descent Algorithm
gradient = matrix(c(sum(eval(D(expression((y-(theta_0)/(1 + exp(-theta_1*(x-theta_2))))^2),'theta_0'),li
),sum(eval(D(expression((y-(theta_0)/(1 + exp(-theta_1*(x-theta_2))))^2),'theta_1'),list(x=as.double(da
),sum(eval(D(expression((y-(theta_0)/(1 + exp(-theta_1*(x-theta_2))))^2),'theta_2'),list(x=as.double(da
)),nrow=3)
alpha_learn = 1
theta_curl_est = as.matrix(theta_curl,nrow=3) - (alpha_learn*(diag(1,nrow=length(theta_curl),ncol=3) %*%
print(paste0("The final estimates are : ",theta_curl_est[1],", ",theta_curl_est[2]," and ",theta_curl_es
```

```
## [1] "The final estimates are : 8.03412288333194, 1.25291689347563 and 0.730318482817717 respectively
```

Part (e)

```r
library(numDeriv)
```

```
##
## Attaching package: 'numDeriv'

## The following objects are masked from 'package:pracma':
##
##     grad, hessian, jacobian
```

```r
library(matlib)
```

```
##
## Attaching package: 'matlib'

## The following objects are masked from 'package:pracma':
##
##     angle, inv
```

```r
hess_mat = matrix(rep(0,9),nrow = 3)

for(i in 1:dim(data_3)[1])
{
  dat <<- as.double(data_3$X)[i]
  new_mod = function(theta_new)
  {
    return(theta_new[1])/(1 + exp(-theta_new[2]*(dat-theta_new[3])))

  }
new_mod(theta_curl)
```

```r
hess_mat = hess_mat + ((as.double(data_3$Y)[i] - model_3(theta_curl,as.double(data_3$X)[i])) * (hessian
}

gradient_new_rap = matrix(c(sum(eval(D(expression((y-(theta_0)/(1 + exp(-theta_1*(x-theta_2))))^2),'thet
),sum(eval(D(expression((y-(theta_0)/(1 + exp(-theta_1*(x-theta_2))))^2),'theta_1'),list(x=as.double(dat
),sum(eval(D(expression((y-(theta_0)/(1 + exp(-theta_1*(x-theta_2))))^2),'theta_2'),list(x=as.double(dat
)),nrow=3)
alpha_learn_new_rap = 1
col_1_new_rap = eval(D(expression((theta_0)/(1 + exp(-theta_1*(x-theta_2)))),'theta_0'),list(theta_1=the
col_2_new_rap = eval(D(expression((theta_0)/(1 + exp(-theta_1*(x-theta_2)))),'theta_1'),list(theta_1=the
col_3_new_rap = eval(D(expression((theta_0)/(1 + exp(-theta_1*(x-theta_2)))),'theta_2'),list(theta_0=the
F_theta_new_rap = cbind(col_1_new_rap,col_2_new_rap,col_3_new_rap)
hess_mat_final = 2 * ((t(F_theta_new_rap)%*%F_theta_new_rap)-hess_mat)
alpha_learn_new_rap = 1
theta_curl_est_new_rap = as.matrix(theta_curl,nrow=3) - (alpha_learn_new_rap * inv(hess_mat_final) %*% g
print(paste0("The final estimates are : ",theta_curl_est_new_rap[1],", ",theta_curl_est_new_rap[2]," and
```

```
## [1] "The final estimates are : 7.75037195324503, 0.475709279671812 and 0.952239950120925 respectively
```

Part (f)

Sub-part (i)

```r
theta_est_conv = c(7.6, 0.5, 1)

col_1_ci = eval(D(expression((theta_0)/(1 + exp(-theta_1*(x-theta_2)))),'theta_0'),list(theta_1=theta_es
col_2_ci = eval(D(expression((theta_0)/(1 + exp(-theta_1*(x-theta_2)))),'theta_1'),list(theta_1=theta_es
col_3_ci = eval(D(expression((theta_0)/(1 + exp(-theta_1*(x-theta_2)))),'theta_2'),list(theta_0=theta_es
F_theta_ci = cbind(col_1_ci,col_2_ci,col_3_ci)
f_theta_ci = as.matrix(model_3(theta_est_conv,as.double(data_3$X)),nrow=20)
sigma = sqrt((t(Y - f_theta_ci) %*% (Y - f_theta_ci))/(nrow(data_3)-length(theta_est_conv)))
print(paste0("95% confidence interval for theta_2 is (",theta_est_conv[2]-(qt(0.025,17,lower.tail = F)*s
```

```
## [1] "95% confidence interval for theta_2 is (0.444964306003697,0.555035693996303)"
```

Sub-part (ii)

```r
f_theta_ci_2 = model_3(theta_est_conv,2.7)
col_1_ci_2 = eval(D(expression((theta_0)/(1 + exp(-theta_1*(x-theta_2)))),'theta_0'),list(theta_1=theta_
col_2_ci_2 = eval(D(expression((theta_0)/(1 + exp(-theta_1*(x-theta_2)))),'theta_1'),list(theta_1=theta_
col_3_ci_2 = eval(D(expression((theta_0)/(1 + exp(-theta_1*(x-theta_2)))),'theta_2'),list(theta_0=theta_
F_theta_ci_2 = cbind(col_1_ci_2,col_2_ci_2,col_3_ci_2)
sigma_2 = sqrt((t(Y - f_theta_ci_2) %*% (Y - f_theta_ci_2))/(nrow(data_3)-length(theta_est_conv)))
print(paste0("95% confidence interval for Y|X=2.7 is (",f_theta_ci_2-(qt(0.025,17,lower.tail = F)*sigma_
```

```
## [1] "95% confidence interval for Y|X=2.7 is (3.9033851026341,6.74523546297029)"
```

```r
print(paste0("95% prediction interval for Y|X=2.7 is (",f_theta_ci_2-(qt(0.025,17,lower.tail = F)*sigma_
```

```
## [1] "95% prediction interval for Y|X=2.7 is (1.23326276854603,9.41535779705836)"
```

Sub-part (iii)

```
library(pracma)
h_theta_given = theta_est_conv[3]/theta_est_conv[2]
col_1_test = eval(D(expression(theta_2/theta_1),'theta_0'),list(theta_est_conv[1],theta_est_conv[2],the
col_2_test = eval(D(expression(theta_2/theta_1),'theta_1'),list(theta_est_conv[1],theta_est_conv[2],the
col_3_test = eval(D(expression(theta_2/theta_1),'theta_2'),list(theta_est_conv[1],theta_est_conv[2],the
h_vect = c(col_1_test,col_2_test,col_3_test)
sigma = sqrt((t(Y - f_theta_ci) %*% (Y - f_theta_ci))/(nrow(data_3)-length(theta_est_conv)))
col_1_ci_2 = eval(D(expression((theta_0)/(1 + exp(-theta_1*(x-theta_2)))),'theta_0'),list(theta_1=theta_
col_2_ci_2 = eval(D(expression((theta_0)/(1 + exp(-theta_1*(x-theta_2)))),'theta_1'),list(theta_1=theta_
col_3_ci_2 = eval(D(expression((theta_0)/(1 + exp(-theta_1*(x-theta_2)))),'theta_2'),list(theta_0=theta_
F_theta_ci_2 = cbind(col_1_ci_2,col_2_ci_2,col_3_ci_2)

print(paste0("The confidence interval for the given parameter combination is :(",h_theta_given - (qt(0.0
 ,",",h_theta_given + (qt(0.025,17,lower.tail = F)*sigma*sqrt(t(h_vect) %*% pinv(t(F_theta_ci_2) %*% F_t
,")"))
```

```
## [1] "The confidence interval for the given parameter combination is :(1.78839688693593,2.211603113064
```

Conclusion : Since, the value of the parameter combination specified in the null hypothesis is contained within the confidence interval, we fail to reject the null hypothesis.

Sub-part (iv)

```
R_sq_pseudo = 1 - (sum((as.double(data_3$Y) - (as.matrix(model_3(theta_est_conv,as.double(data_3$X)),nro
print(paste0("The value of pseudo R_squared is : ",R_sq_pseudo))
```

```
## [1] "The value of pseudo R_squared is : 0.991327316008666"
```

Sub-part (v)

```
F_theta_hat = cbind(col_1_ci,col_2_ci,col_3_ci)
mat_hat = F_theta_hat %*% inv(t(F_theta_hat) %*% F_theta_hat) %*% t(F_theta_hat)
for(i in seq_along(diag(mat_hat)))
{
  if(mat_hat[i,i]>((2*3)/20))
  {
    print(paste0("The ",i," th observation exhibits substantial leverage."))
  }
  else
  {
    print(paste0("The ",i," th observation does not exhibit substantial leverage."))
  }
}
```

```
## [1] "The 1 th observation exhibits substantial leverage."
## [1] "The 2 th observation does not exhibit substantial leverage."
## [1] "The 3 th observation does not exhibit substantial leverage."
## [1] "The 4 th observation does not exhibit substantial leverage."
## [1] "The 5 th observation does not exhibit substantial leverage."
## [1] "The 6 th observation does not exhibit substantial leverage."
```

```
## [1] "The 7 th observation does not exhibit substantial leverage."
## [1] "The 8 th observation does not exhibit substantial leverage."
## [1] "The 9 th observation does not exhibit substantial leverage."
## [1] "The 10 th observation does not exhibit substantial leverage."
## [1] "The 11 th observation does not exhibit substantial leverage."
## [1] "The 12 th observation does not exhibit substantial leverage."
## [1] "The 13 th observation does not exhibit substantial leverage."
## [1] "The 14 th observation does not exhibit substantial leverage."
## [1] "The 15 th observation does not exhibit substantial leverage."
## [1] "The 16 th observation does not exhibit substantial leverage."
## [1] "The 17 th observation does not exhibit substantial leverage."
## [1] "The 18 th observation does not exhibit substantial leverage."
## [1] "The 19 th observation does not exhibit substantial leverage."
## [1] "The 20 th observation does not exhibit substantial leverage."
```

PROBLEM 4.

Loading the data in R :

```r
library(dplyr)

data_4 = read.table("C:\\Users\\Jayaditya Nath\\Downloads\\S24hw1pr4.txt")
data_4 = as.data.frame(data_4 %>% rename("X"="V1","Y"="V2"))
data_4
```

```
##          X      Y
## 1   0.0094 2.6722
## 2   0.2391 2.0101
## 3   0.3057 2.0008
## 4   0.3524 1.8691
## 5   0.4541 2.0877
## 6   0.5560 1.5966
## 7   0.5758 1.4953
## 8   0.7677 1.7202
## 9   0.8002 1.4170
## 10  0.8916 1.5328
## 11  0.9374 1.2885
## 12  0.9676 1.4269
## 13  0.9763 0.6416
## 14  0.9895 1.4577
## 15  1.0207 1.8053
## 16  1.1413 1.4597
## 17  1.1446 1.4212
## 18  1.2862 0.9679
## 19  1.4773 0.9059
## 20  1.5027 0.9005
## 21  1.6191 0.7584
## 22  1.6807 0.4525
## 23  1.6888 0.5767
## 24  1.8040 0.5312
## 25  1.8722 0.3936
## 26  1.9028 0.7308
## 27  1.9044 1.1955
## 28  1.9746 0.0847
```

```
## 29   2.0107  0.4554
## 30   2.1268 -0.0583
## 31   2.1459  0.6801
## 32   2.1755  0.6890
## 33   2.2155 -0.1797
## 34   2.2308  0.7976
## 35   2.4108 -0.1479
## 36   2.4142  0.2091
## 37   2.4533  0.4240
## 38   2.4997  0.1355
## 39   2.5567  0.6905
## 40   2.6373  0.7700
## 41   2.6569  0.0432
## 42   2.7000 -0.5403
## 43   2.7855 -0.1630
## 44   2.9963 -0.3977
## 45   3.0106 -0.3814
## 46   3.1620 -0.5466
## 47   3.1879 -0.2270
## 48   3.1968  0.3249
## 49   3.2061 -0.4518
## 50   3.3155 -0.3754
## 51   3.4033 -0.0113
## 52   3.4148 -0.0027
## 53   3.4565 -0.2164
## 54   3.4652 -0.0714
## 55   3.5398 -0.1388
## 56   3.5470 -0.4841
## 57   3.5975 -0.2781
## 58   3.6989 -0.5079
## 59   3.7076 -0.6749
## 60   3.7523 -0.2673
## 61   3.7610 -0.4251
## 62   3.7693 -0.4971
## 63   3.8268 -0.3814
## 64   3.8883 -0.2065
## 65   3.9362 -0.3811
## 66   4.1012 -0.4722
## 67   4.3648 -0.4972
## 68   4.3701 -0.5607
## 69   4.4412 -0.7977
## 70   4.7186 -0.4049
## 71   4.7660 -0.2393
## 72   4.8649 -0.7803
## 73   4.8653 -0.1062
## 74   5.0736 -0.6969
## 75   5.1813 -0.7208
## 76   5.3196 -0.9049
## 77   5.3989 -0.8780
## 78   5.4235 -0.9224
## 79   5.5326 -0.5659
## 80   5.7253 -0.2251
## 81   5.7334 -0.5179
## 82   5.8962 -0.4256
```

```
## 83   5.9549 -0.2905
## 84   5.9554 -1.3425
## 85   5.9929 -0.7028
## 86   6.0453 -0.5585
## 87   6.2588 -0.4500
## 88   6.2618 -1.0036
## 89   6.4661 -0.9109
## 90   6.4734 -0.4184
## 91   6.5234 -0.6639
## 92   6.5847 -0.4766
## 93   6.7931 -0.8769
## 94   7.0413 -1.0423
## 95   7.0822 -0.3831
## 96   7.1047 -0.2629
## 97   7.1422 -0.1396
## 98   7.1746 -1.0035
## 99   7.2824 -0.9408
## 100  7.3252 -0.9746
## 101  7.4614 -0.5704
## 102  7.5065 -0.7235
## 103  7.6376 -0.1831
## 104  7.7386 -0.9321
## 105  7.8956 -0.9960
## 106  7.9510 -0.4574
## 107  7.9615 -0.1859
## 108  7.9632 -0.7323
## 109  8.0133 -0.8543
## 110  8.0285 -0.5471
## 111  8.0707 -0.9503
## 112  8.3671 -0.4690
## 113  8.4072 -0.2431
## 114  8.4324 -0.4079
## 115  8.6303 -0.9715
## 116  8.6314 -0.5955
## 117  8.7173 -0.5231
## 118  8.7274  0.0573
## 119  8.8123 -0.1808
## 120  8.8264 -0.4457
## 121  8.9101 -0.0636
## 122  8.9132 -0.7080
## 123  8.9657 -0.5124
## 124  8.9841 -0.0284
## 125  9.2408  0.0198
## 126  9.2596 -0.1899
## 127  9.3609 -0.2464
## 128  9.3939 -0.5384
## 129  9.4049 -0.1729
## 130  9.4573 -0.5990
## 131  9.4982 -0.2112
## 132  9.5462 -0.2872
## 133  9.6132 -0.0465
## 134  9.6444 -0.4724
## 135  9.6947 -0.1773
## 136  9.7197 -0.1321
```

```
## 137  9.7333 -0.5942
## 138  9.7899 -0.3006
## 139  9.8049  0.0953
## 140  9.9412 -0.2999
## 141 10.0626 -0.4053
## 142 10.0901 -0.1934
## 143 10.1135 -0.3920
## 144 10.1696 -0.1908
## 145 10.3268 -0.3367
## 146 10.3439 -0.3034
## 147 10.4416 -0.1647
## 148 10.4660  0.0840
## 149 10.4805 -0.1642
## 150 10.4902  0.1642
## 151 10.5478 -0.8384
## 152 10.5497 -0.4058
## 153 10.5845 -0.2156
## 154 10.7498 -0.4969
## 155 11.0262 -0.5983
## 156 11.0648 -0.0773
## 157 11.1773 -0.0710
## 158 11.2673 -0.1537
## 159 11.3021 -0.1224
## 160 11.3379 -0.4668
## 161 11.4336 -0.3587
## 162 11.5077 -0.2471
## 163 11.5256 -0.4051
## 164 11.6199 -0.1202
## 165 11.6516  0.4385
## 166 11.6676 -0.3403
## 167 11.8970 -0.0178
## 168 11.9305 -0.4765
## 169 11.9955  0.1772
## 170 12.0828 -0.4504
## 171 12.0858 -0.1276
## 172 12.1570 -0.3504
## 173 12.2157 -0.0378
## 174 12.3015 -0.1705
## 175 12.3803 -0.1817
## 176 12.5129  0.6445
## 177 12.5328 -0.3348
## 178 12.5400 -0.1747
## 179 12.6729  0.5791
## 180 12.7245 -0.4673
## 181 12.9788  0.0093
## 182 13.1159 -0.4312
## 183 13.1355  0.1467
## 184 13.2528 -0.0412
## 185 13.2541  0.0478
## 186 13.2812 -0.2061
## 187 13.2955 -0.0620
## 188 13.3426  0.2759
## 189 13.4198 -0.7475
## 190 13.4767 -0.5696
```

```
## 191 13.5888 -0.0044
## 192 13.5914 -0.2974
## 193 13.7431 -0.1939
## 194 13.7512 -0.5224
## 195 13.8394 -0.3572
## 196 13.8425 -0.2200
## 197 13.8441 -0.3080
## 198 13.8956 -0.3448
## 199 14.1401 -0.2152
## 200 14.2193 -0.3554
## 201 14.2273  0.2705
## 202 14.4243 -0.3619
## 203 14.5533  0.4102
## 204 14.6771 -0.1868
## 205 14.7277 -0.4805
## 206 14.7910 -0.2508
## 207 14.8605  0.1764
## 208 15.0030  0.2131
## 209 15.1597 -0.0285
## 210 15.2074  0.0075
## 211 15.2132  0.1022
## 212 15.3128 -0.4685
## 213 15.3540 -0.2273
## 214 15.3738 -0.5688
## 215 15.4609 -0.1887
## 216 15.4879  0.1948
## 217 15.5005  0.1316
## 218 15.5156  0.5620
## 219 15.5288 -0.0186
## 220 15.5779 -0.6539
## 221 15.6589  0.4010
## 222 15.7307  0.2909
## 223 15.8223 -0.2181
## 224 15.9171  0.0042
## 225 15.9799  0.1579
## 226 16.0335  0.0507
## 227 16.1090  0.1374
## 228 16.1494 -0.0835
## 229 16.1728 -0.0629
## 230 16.2153 -0.0274
## 231 16.2928 -0.2665
## 232 16.4184  0.3415
## 233 16.7096 -0.0421
## 234 16.7330  0.1445
## 235 16.7616 -0.2772
## 236 16.7914 -0.3371
## 237 16.8193 -0.0672
## 238 16.9046 -0.1103
## 239 16.9169  0.2863
## 240 17.0628  0.1270
## 241 17.0875  0.1093
## 242 17.5093 -0.4294
## 243 17.6815 -0.2968
## 244 17.7192  0.0239
```

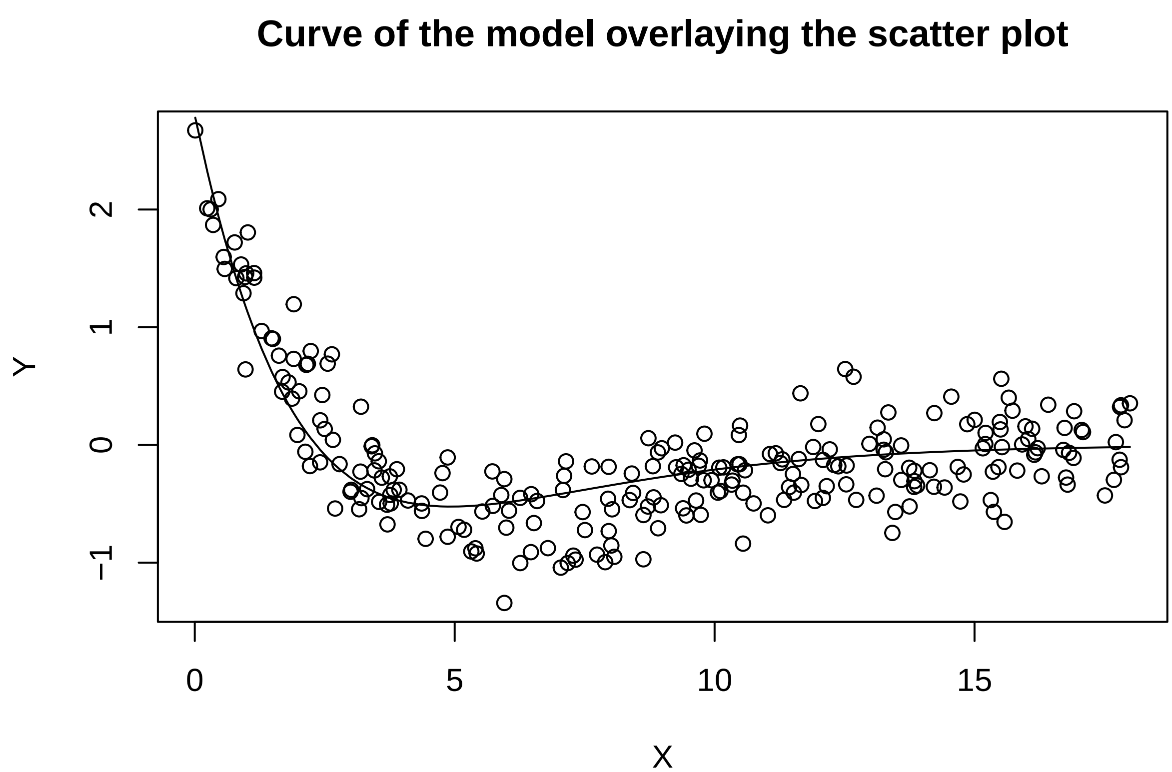
```
## 245 17.7915 -0.1259
## 246 17.8004  0.3228
## 247 17.8177  0.3369
## 248 17.8191 -0.1894
## 249 17.8867  0.2100
## 250 17.9908  0.3535
```

Part (a)

I set the initial values of the parameters by graphing out the function with different parameter values(by trial and error method) overlaying the scatterplot :

```
model = function(theta_vec,x)
{
  theta_0 = theta_vec[1]
  theta_1 = theta_vec[2]
  theta_2 = theta_vec[3]
  y = (theta_0 + theta_1*x)/(1 + theta_2*exp(0.4*x))
  return(y)
}

initial_theta_curl = c(14,-6,4)
plot(data_4$X,data_4$Y,xlab="X",ylab="Y",main = "Curve of the model overlaying the scatter plot")
lines(data_4$X,model(initial_theta_curl,data_4$X))
```



Part (b)

```r
#Gauss_Newton Algorithm

library(pracma)
Y = as.matrix(data_4$Y,nrow=250)
theta_curl = initial_theta_curl

for(iter in 1:11)
{
  f_theta = as.matrix(model(theta_curl,data_4$X),nrow=250)
  col_1 = eval(D(expression((theta_0 + theta_1*x)/(1 + theta_2*exp(0.4*x))),'theta_0'),list(theta_2=the
  col_2 = eval(D(expression((theta_0 + theta_1*x)/(1 + theta_2*exp(0.4*x))),'theta_1'),list(theta_2=the
  col_3 = eval(D(expression((theta_0 + theta_1*x)/(1 + theta_2*exp(0.4*x))),'theta_2'),list(theta_0=the
  F_theta = cbind(col_1,col_2,col_3)

  delta = pinv(t(F_theta) %*% F_theta) %*% t(F_theta) %*% (Y - f_theta)
  theta_curl = as.matrix(theta_curl) + delta

  if(max(abs(delta))<0.0001)
  {
    break
  }
}


print(paste0("The final estimates of theta are : ",theta_curl[1],",",theta_curl[2]," and ",theta_curl[3
```

```
## [1] "The final estimates of theta are : -199810.043853842,83234.820631118 and -62599.4118620489 resp
```

```r
print(paste0("The value of sigma_squared is : ",(t(Y - f_theta) %*% (Y - f_theta))/(nrow(data_4)-dim(th
```

```
## [1] "The value of sigma_squared is : 0.103611874745618"
```

```r
print("The variance-covariance matrix is of the form : ")
```

```
## [1] "The variance-covariance matrix is of the form : "
```

```r
as.numeric((t(Y - f_theta) %*% (Y - f_theta))/(nrow(data_4)-dim(theta_curl)[1]))*as.matrix(solve(crossp
```

```
##               col_1         col_2         col_3
## col_1  3.661025e+18 -1.525069e+18  1.146967e+18
## col_2 -1.525069e+18  6.352962e+17 -4.777906e+17
## col_3  1.146967e+18 -4.777906e+17  3.593344e+17
```

```r
print(paste0("The algorithm converges at the ",iter," th iteration."))
```

```
## [1] "The algorithm converges at the 7 th iteration."
```

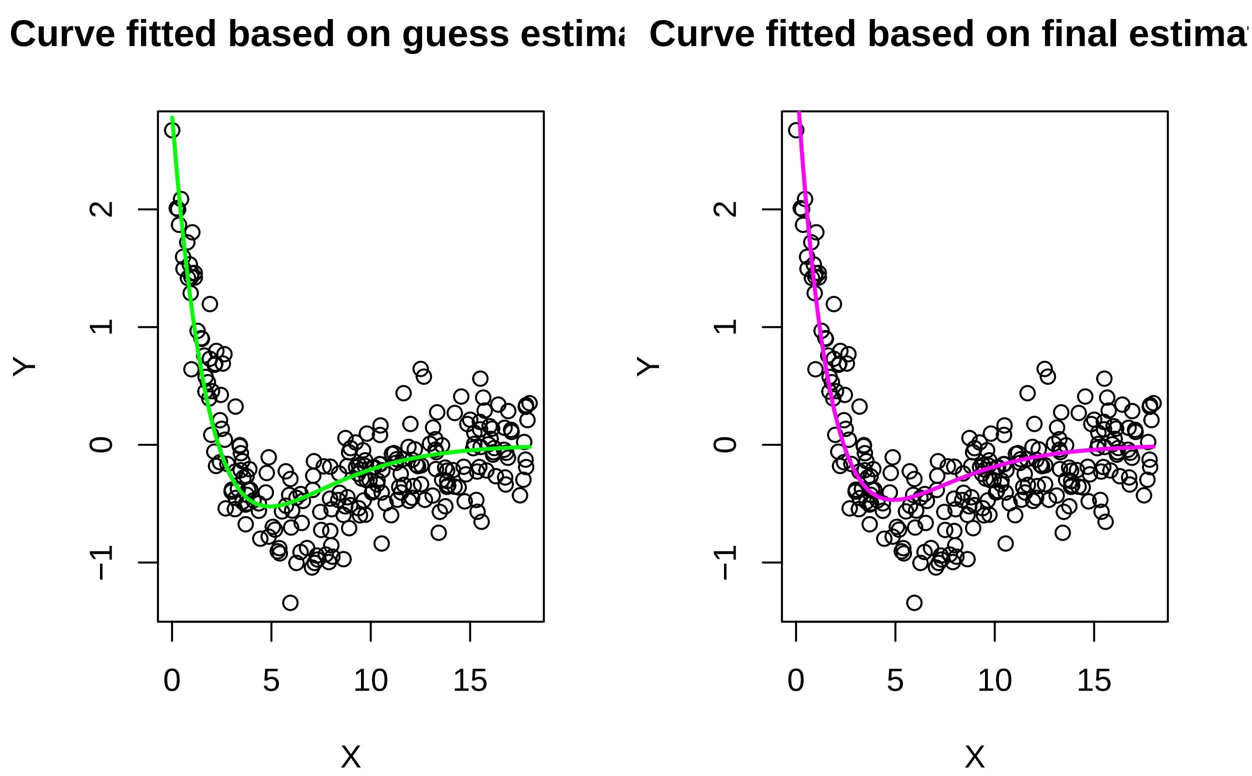
```r
print(paste0("The value of the objective function at convergence is : ",sum((Y-f_theta)^2)))
```

```
## [1] "The value of the objective function at convergence is : 25.5921330621676"
```

```r
print("The convergence criterion used above relates to the minor change in parameter estimates.")
```

```
## [1] "The convergence criterion used above relates to the minor change in parameter estimates."
```

```r
par(mfrow=c(1,2))
plot(data_4$X,data_4$Y,xlab="X",ylab="Y",main = "Curve fitted based on guess estimates")
lines(data_4$X,model(initial_theta_curl,data_4$X),col="green",lwd=2)
plot(data_4$X,data_4$Y,xlab="X",ylab="Y",main="Curve fitted based on final estimates")
lines(data_4$X,model(theta_curl,data_4$X),col="magenta",lwd=2)
```



Comment : The only issue I recognized while solving the given problem is that the Jacobian matrix was singular and thus not invertible. So, as a remedy, I used the Moore-Penrose inverse.