UCS409

ASSIGNMENT-1

SUBMITTED TO:

RAAHAT DEVENDER SINGH

SUBMITTED BY:

SHRUTI MAHAJAN

ROLL NO:

101983046

BATCH:

COE-21

ASSIGNMENT1
Name:Shruti Mahajan
Roll Number:101983046

1.

Given the following PDF:

$$f(x,y) = \begin{cases} c & for\ \ 0 < x < 5, y > 0, x-1 < y < x+1 \\ 0 & everywhere\ else \end{cases}$$

i. Find the joint probability density function (i.e., the value of *c*)
ii. Find the associated marginal and conditional distributions.
iii. Find E(*X*), E(*Y*), Var(*X*), Var(*Y*).
iv. Find E(*Y* | *X*) and E(*X* | *Y*).
v. Coefficient of correlation between *X* and *Y*.

ASSIGNMENT1
Name:Shruti Mahajan
Roll Number:101983046

Name - Shruti Mahajan   Roll No-101983046
Batch - COE21

ASSIGNMENT-1

(ii)

① $0 < x < 5$, $x-1 < y < x+1$, $y > 0$

$$f(x,y) = \begin{cases} c & 0<x<5,\ x-1<y<x+1,\ y>0 \\ 0 & \text{everywhere else} \end{cases}$$

(i) $$\int_0^1\int_0^{x+1} c\,dy\,dx + \int_1^5\int_{x-1}^{x+1} c\,dy\,dx = 1$$

$$\int_0^1 c[y]_0^{x+1}dx + \int_1^5 [cy]_{x-1}^{x+1}dx = 1$$

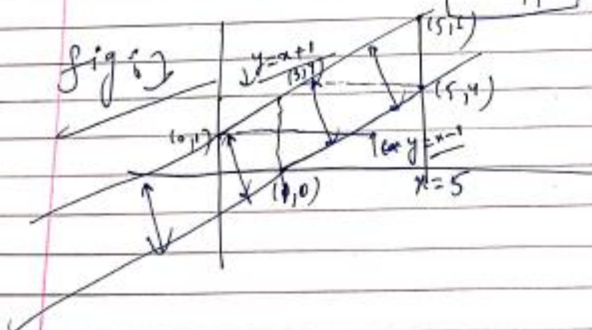
$$\int_0^1 c(x+1)\,dx + \int_1^5 c(x+1-x+1)\,dx = 1$$

$$c\left[\frac{x^2}{2}+x\right]_0^1 + 2c[x]_1^5 = 1$$

$$\frac{19c}{2} = 1$$

$$\boxed{c = \frac{2}{19}}$$

fig (i)



Scanned with CamScanner

ASSIGNMENT1
Name:Shruti Mahajan
Roll Number:101983046

Name - Shruti Mahajan   101983046
COE-21

(ii) $g(x) = \int_{-\infty}^{\infty} f(x,y)\,dy$

$= \int_{0}^{x+1} \frac{2}{19}\,dy$

$= \frac{2}{19}[y]_0^{x+1}$

$= \frac{2}{19}(x+1) \quad 0<x<1$

& $1<x<5$ it is $\Rightarrow \int_{x-1}^{x+1} \frac{2}{19}\,dy$

$= \left[\frac{2}{19}y\right]_{x-1}^{x+1} = \boxed{4/19}$

$$g(x) = \begin{cases} \frac{2}{19}(x+1) & 0<x\le 1 \\ \frac{4}{19} & 1<x<5 \\ 0 & \text{elsewhere} \end{cases}$$

$h(y) = \int_{-\infty}^{\infty} f(x,y)\,dx$

$\therefore\ x<y+1$ & $x>y-1$

$0<y<1 \quad = \int_{0}^{y+1} \frac{2}{19}\,dx$

ASSIGNMENT1
Name:Shruti Mahajan
Roll Number:101983046

Shruti Mahajan COE21 101983046

$$= \frac{2}{19}(y+1)$$

for $1<y<4$

$$= \int_{y-1}^{y+1} \frac{2}{19}\,dx = \boxed{\frac{4}{19}}$$

for $4<y<6$

$$= \int_{y-1}^{5} \frac{2}{19}\,dx = \frac{2}{19}[x]_{y-1}^{5}$$

$$= \frac{2}{19}(6-y)$$

$$h(y) = \begin{cases} \frac{2}{19}(y+1) & 0<y\le 1 \\ \frac{4}{19} & 1<y\le 4 \\ \frac{2}{19}(6-y) & 4<y<6 \\ 0 & \text{elsewhere} \end{cases}$$

$h(y/x)$ for $0<x\le 1$

$$= \frac{2\times 19}{19\times 2(x+1)} = \frac{1}{x+1}$$

for $1<x<5 \Rightarrow \frac{2\times 19}{19\times 4} = \frac{1}{2}$

Shruti Mahajan COE 21

101983046

$$h(y/x) = \begin{cases} 1/(x+1) & 0 < x \le 1 \\ 1/2 & 1 < x < 3 \\ 0 & \text{elsewhere} \end{cases}$$

$$g(x/y) = \frac{f(x,y)}{h(y)} \quad \text{for } 0 < y \le 1$$

$$= \frac{2/19 \times 19}{2(y+1)} = \boxed{1/(y+1)}$$

for $1 < y \le 4$ $= \frac{2 \times 19}{19 \times 4} = 1/2$

for $4 < y < 6$ $= 1/(6-y)$ $\therefore \boxed{\frac{2 \times 19}{19 \times 2(6-y)}}$

$$g(x/y) = \begin{cases} 1/(y+1) & 0 < y \le 1 \\ 1/2 & 0 < y \le 4 \\ 1/(6-y) & 4 < y < 6 \end{cases}$$

(iii) $E(Y) = \int_{-\infty}^{\infty} y\, h(y)\, dy$

$$= \int_0^1 \frac{2}{19}(y^2+y) + \int_1^4 4/19\, y + \int_4^6 2/19 (6y-y^2)$$

$$= 2/19 \times 5/6 + 4/19 \times 1/2 \times 15 + \frac{2}{19}\left(\frac{36-8}{3}\right)$$

$$= \frac{151}{57}$$

$$E(x) = \int_{-\infty}^{\infty} x\, g(x)\, dx$$

Scanned with CamScanner

Shruti Mahajan 101983046 COE21

$$= \int_0^1 \frac{2}{19}(x^2+x)\,dx + \int_1^5 \frac{4}{19}x\,dx$$

$$= \frac{149}{57}$$

$$E(y^2) = \int_{-\infty}^{\infty} y^2 h(y)\,dx$$

$$= \int_0^1 \frac{2}{19}(y^3+y^2)\,dy + \int_1^4 \frac{4}{19}y^2 + \int_4^6 \frac{2}{19}(6y^2-y^3)$$

$$= \frac{7}{114} + \frac{84}{19} + \frac{2}{19}\times 47$$

$$= \frac{1039}{114}$$

$$E(x^2) = \int_{-\infty}^{\infty} x^2 g(x)\,dx$$

$$= \int_0^1 \frac{2}{19}(x^3+x^2)\,dx + \int_1^5 \frac{4}{19}x^2\,dx$$

$$= \frac{999}{114}$$

$$Var(x) = E(x^2) - (E(x))^2$$

$$= \frac{994}{114} - \left(\frac{149}{57}\right)^2 = 1.9$$

$$Var(y) = E(y^2) - (E(y))^2 = 2.07$$

(iv) $E(x/y) = \int_{-\infty}^{\infty} x\, g(x/y)\,dx$

Shruti Mahajan 101983046 COE-21

$= \int_0^{y+1} \frac{x}{y+1}\, dx$ for $0 < y < 1$

$= \frac{(y+1)}{2}$

for $1 < y < 4 \Rightarrow \int_{y-1}^{y+1} \frac{x}{2}\, dx$

$= y$

for $4 < y < 1$

$= \int_{y-1}^{5} \frac{x}{6-y}\, dx$

$= \frac{1}{6-y}\left(\frac{25}{2} - \frac{(y-1)^2}{2}\right)$

$= \frac{25-(y-1)^2}{2(6-y)}$

$$E(x/y) = \begin{cases} \frac{y+1}{2} & 0 < y < 1 \\ y & 1 < y \leq 4 \\ \frac{25-(y-1)^2}{2(6-y)} & 4 < y < 1 \\ 0 & \text{elsewhere.} \end{cases}$$

$E(y/x) = \int_{-\infty}^{\infty} y\, h(y/x)\, dy$

for $0 < x < 1$
$E(y/x) = \int_x^{x+1} \frac{y}{x+1}\, dy$

Shruti Mahajan 101983046

$$= \frac{x+1}{2}$$

for $1 < x < 5$

$$E(y/x) = \int_{x-1}^{x+1} \frac{1}{2} y \, dy$$

$$= \frac{1}{2}\left(\frac{(x+1)^2 - (x-1)^2}{2}\right)$$

$$= x$$

$$E(y/x) = \begin{cases} \frac{x+1}{2} & 0 < x \leq 1 \\ x & 1 < x < 5 \\ 0 & \text{elsewhere} \end{cases}$$

(v) $\rho = \frac{cov(x,y)}{\sqrt{\sigma_x \sigma_y}}$

$$cov(x,y) = E(xy) - E(x)F(y)$$

$$\cancel{C(x,y) = E(x,y) - E(x)}$$

$$E(XY) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} xy \, f(x,y) \, dx \, dy$$

$$= \int_0^1 \int_0^{x+1} xy \frac{2}{19} \, dx \, dy + \int_1^5 \int_{x-1}^{x+1} xy \frac{1}{19} \, dy \, dx$$

$$= \frac{1}{19}\left(\frac{11}{2} + \frac{496}{3}\right)$$

Shruti Mahajan 1019830416 2 of 21

$= \underline{8.7}$

$E(x)\,E(Y) = \underline{6.92}$

$Corr(\rho) = \dfrac{8.7 - 6.9}{\sqrt{1.93 \times 2.096}}$

$= \underline{0.9}$

2) Execution of the following three R commands will give us the data {(x(i), y(i), z(i), i = 1, 2, …,100}. x<-rpois(100, 50) y<-rpois(100, 100) z<-rpois(100, 150) Using this data:

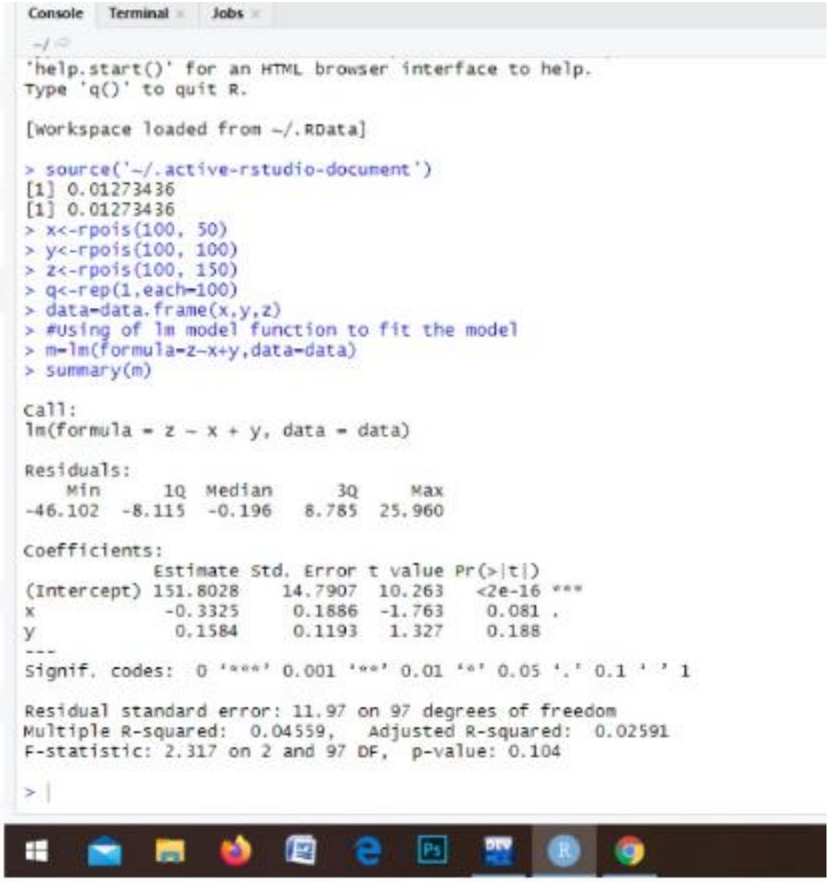
a) a. Fit the linear regression model of the form z = a + b.x + c.y using, i. R ii. By obtaining the three normal equations in a, b, and c. Solve these equations using R and compare the model thus obtained with the model obtained in (i). The compactness of R commands will be important for this part of the problem. For example, a single line R command resulting into the formation and solution of the three equations shall attract the maximum marks. Support your solution with appropriate tables involving the data, R commands, and results obtained from R commands.

```
x<-rpois(100, 50)

y<-rpois(100, 100)

z<-rpois(100, 150)

q<-rep(1,each=100)

data=data.frame(x,y,z)

#Using of lm model function to fit the model

m=lm(formula=z~x+y,data=data)

summary(m)


#R-function for calculating the value of  r-squared for below normal equations
```

```
rsquared=function(a,b,c){

  ybar=mean(z)

  total_var=sum((z-ybar)^2)

  ycap=(a+b*x+c*y)

  residual_var=sum((ycap-ybar)^2)

  rsqr=residual_var/total_var

  return (rsqr)

}

#solving of normal equation

beta = solve(t(as.matrix(cbind(q,x,y))) %% as.matrix(cbind(q,x,y))) %% (t(as.matrix(cbind(q,x,y))) %*% as.matrix(z))

#calling of r-squared function for normal equation method

rsquare_normal=rsquared(beta[1],beta[2],beta[3])


#calling of lm r-squared function

r=summary(m)$r.squared


#arranging the resulted data as the data frame of both result to compare them easily

#d=as.matrix(m$coefficients)

#coefficients_array=data.frame(beta,d)

#c=data.frame(rsquare_normal,r)

#print(coefficients_array)

#print(c)

print(rsquare_normal)

print(r)

library(ggplot2)

xyz<-data.frame(x,y,z)


ggplot(xyz,aes(y=z,x=x,color=y))+
```

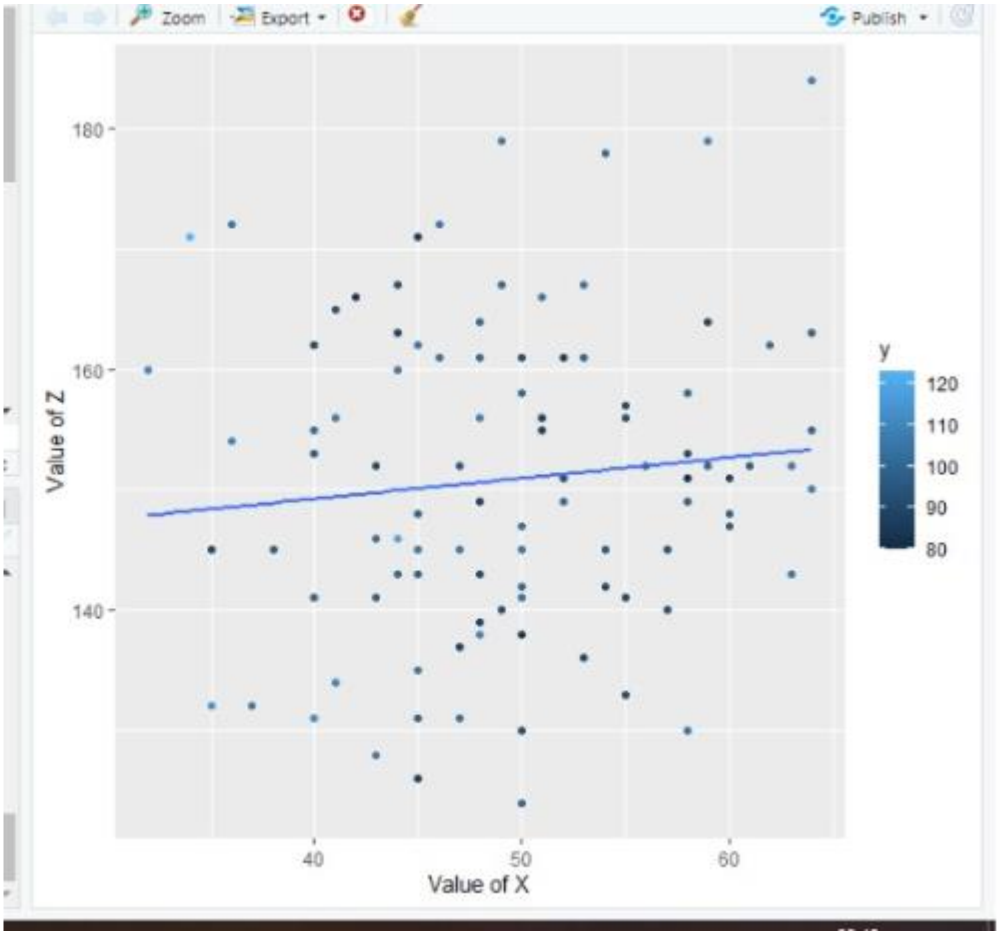
```
  xlab("Value of X")+

  ylab("Value of Z")+

  geom_point()+

  geom_smooth(method="lm",stat = "smooth",se=FALSE)x<-rpois(100, 50)

y<-rpois(100, 100)

z<-rpois(100, 150)

q<-rep(1,each=100)

data=data.frame(x,y,z)

#Using of lm model function to fit the model

m=lm(formula=z~x+y,data=data)

summary(m)


#R-function for calculating the value of  r-squared for below normal equations

rsquared=function(a,b,c){

  ybar=mean(z)

  total_var=sum((z-ybar)^2)

  ycap=(a+b*x+c*y)

  residual_var=sum((ycap-ybar)^2)

  rsqr=residual_var/total_var

  return (rsqr)

}

#solving of normal equation

beta = solve(t(as.matrix(cbind(q,x,y))) %*% as.matrix(cbind(q,x,y))) %*% (t(as.matrix(cbind(q,x,y))) %*%
as.matrix(z))

#calling of r-squared function for normal equation method

rsquare_normal=rsquared(beta[1],beta[2],beta[3])


#calling of lm r-squared function

r=summary(m)$r.squared
```

```
#arranging the resulted data as the data frame of both result to compare them easily

#d=as.matrix(m$coefficients)

#coefficients_array=data.frame(beta,d)

#c=data.frame(rsquare_normal,r)

#print(coefficients_array)

#print(c)

print(rsquare_normal)

print(r)

library(ggplot2)

xyz<-data.frame(x,y,z)


ggplot(xyz,aes(y=z,x=x,color=y))+

  xlab("Value of X")+

  ylab("Value of Z")+

  geom_point()+

  geom_smooth(method="lm",stat = "smooth",se=FALSE)
```



```
Console   Terminal ×   Jobs ×
~/
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> source('~/.active-rstudio-document')
[1] 0.01273436
[1] 0.01273436
> x<-rpois(100, 50)
> y<-rpois(100, 100)
> z<-rpois(100, 150)
> q<-rep(1,each=100)
> data=data.frame(x,y,z)
> #using of lm model function to fit the model
> m=lm(formula=z~x+y,data=data)
> summary(m)

Call:
lm(formula = z ~ x + y, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-46.102  -8.115  -0.196   8.785  25.960

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 151.8028    14.7907  10.263   <2e-16 ***
x            -0.3325     0.1886  -1.763    0.081 .
y             0.1584     0.1193   1.327    0.188
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 11.97 on 97 degrees of freedom
Multiple R-squared:  0.04559,	Adjusted R-squared:  0.02591
F-statistic: 2.317 on 2 and 97 DF,  p-value: 0.104

>
```

```
Console   Terminal ×   Jobs ×
~/
> rsquared=function(a,b,c){
+   ybar=mean(z)
+   total_var=sum((z-ybar)^2)
+   ycap=(a+b*x+c*y)
+   residual_var=sum((ycap-ybar)^2)
+   rsqr=residual_var/total_var
+   return (rsqr)
+ }
> #solving of normal equation
> beta = solve(t(as.matrix(cbind(q,x,y))) %*% as.matrix(cbind(q,x,y))) %*% (t(as.matrix
(cbind(q,x,y))) %*% as.matrix(z))
> #calling of r-squared function for normal equation method
> rsquare_normal=rsquared(beta[1],beta[2],beta[3])
>
> #calling of lm r-squared function
> r=summary(m)$r.squared
>
> #arranging the resulted data as the data frame of both result to compare them easily
> #d=as.matrix(m$coefficients)
> #coefficients_array=data.frame(beta,d)
> #c=data.frame(rsquare_normal,r)
> #print(coefficients_array)
> #print(c)
> print(rsquare_normal)
[1] 0.04559069
> print(r)
[1] 0.04559069
> library(ggplot2)
> xyz<-data.frame(x,y,z)
>
> ggplot(xyz,aes(y=z,x=x,color=y))+
+   xlab("value of x")+
+   ylab("value of z")+
+   geom_point()+
+   geom_smooth(method="lm",stat = "smooth",se=FALSE)
`geom_smooth()` using formula 'y ~ x'
> |
```

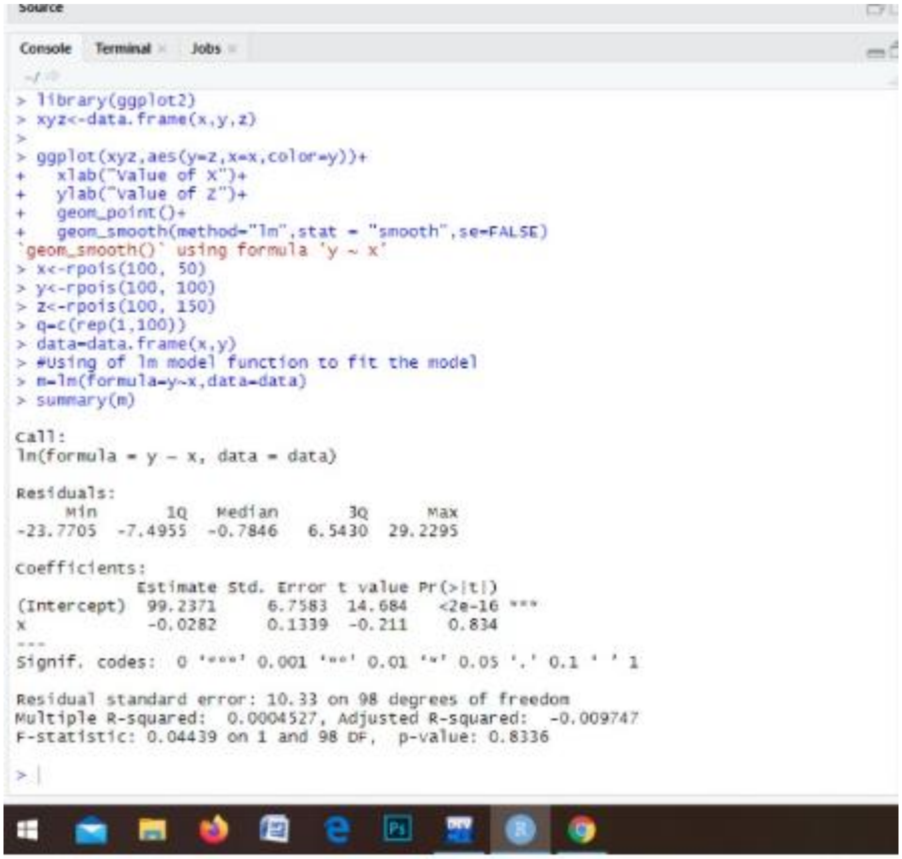


b.) Fit the three models of the form y = a + b.x, y = a + b.x + c.x2 , and y = a.bx to this data using, i. R ii. By obtaining two normal equations in a and b (or three normal equations in a, b, and c). Solve these equations using R and compare the models thus obtained with the models obtained in (i). The compactness of R commands will again be important for this part of the problem. For example, a single line R command resulting into the formation and solution of the 2 equations shall attract the maximum marks. iii. Also, find the coefficient of determination, with the help of formula, for the three models and decide for the best model.

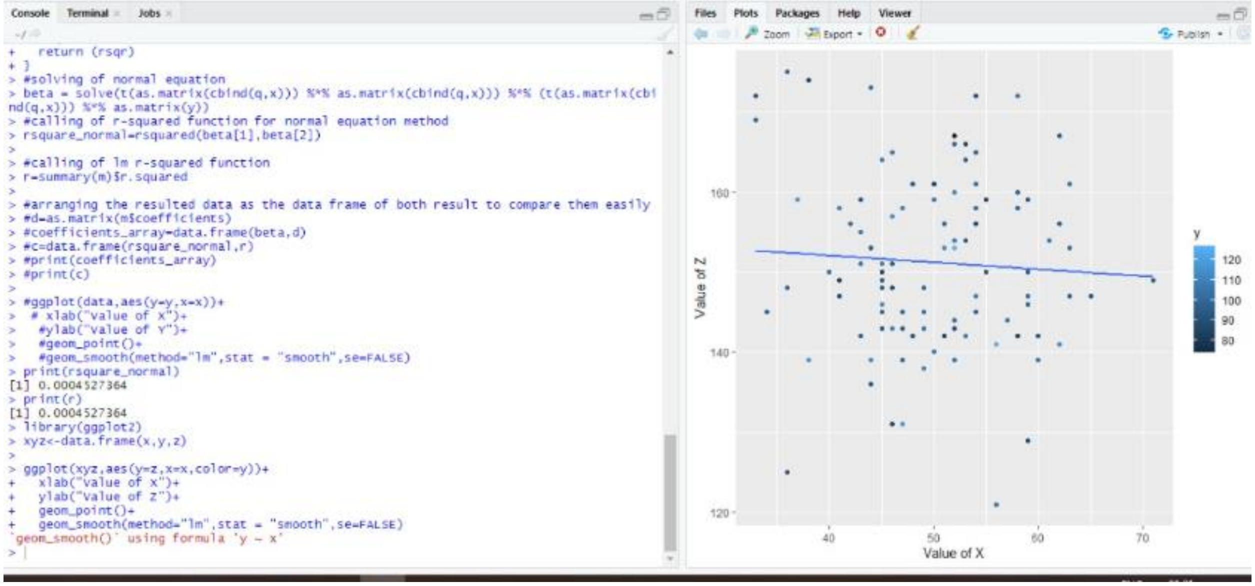
FOR-LINEAR(y = a + b.x)

rsquared=function(a,b){

```
  ybar=mean(y)

  total_var=sum((y-ybar)^2)

  ycap=(a+b*x)

  residual_var=sum((ycap-ybar)^2)

  rsqr=residual_var/total_var

  return (rsqr)

}

#solving of normal equation

beta = solve(t(as.matrix(cbind(q,x))) %% as.matrix(cbind(q,x))) %% (t(as.matrix(cbind(q,x))) %*%
as.matrix(y))

#calling of r-squared function for normal equation method

rsquare_normal=rsquared(beta[1],beta[2])


#calling of lm r-squared function

r=summary(m)$r.squared

print(rsquare_normal)

print(r)

library(ggplot2)

xyz<-data.frame(x,y,z)


ggplot(xyz,aes(y=z,x=x,color=y))+

  xlab("Value of X")+

  ylab("Value of Z")+

  geom_point()+

  geom_smooth(method="lm",stat = "smooth",se=FALSE)
```



```
> library(ggplot2)
> xyz<-data.frame(x,y,z)
>
> ggplot(xyz,aes(y=z,x=x,color=y))+
+   xlab("value of x")+
+   ylab("value of z")+
+   geom_point()+
+   geom_smooth(method="lm",stat = "smooth",se=FALSE)
`geom_smooth()` using formula 'y ~ x'
> x<-rpois(100, 50)
> y<-rpois(100, 100)
> z<-rpois(100, 150)
> q=c(rep(1,100))
> data=data.frame(x,y)
> #Using of lm model function to fit the model
> m=lm(formula=y~x,data=data)
> summary(m)

Call:
lm(formula = y ~ x, data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-23.7705  -7.4955  -0.7846   6.5430  29.2295

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  99.2371     6.7583  14.684   <2e-16 ***
x            -0.0282     0.1339  -0.211    0.834
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.33 on 98 degrees of freedom
Multiple R-squared:  0.0004527, Adjusted R-squared:  -0.009747
F-statistic: 0.04439 on 1 and 98 DF,  p-value: 0.8336

>
```



```
+   return (rsqr)
+ }
> #solving of normal equation
> beta = solve(t(as.matrix(cbind(q,x))) %*% as.matrix(cbind(q,x))) %*% (t(as.matrix(cbi
nd(q,x))) %*% as.matrix(y))
> #calling of r-squared function for normal equation method
> rsquare_normal=rsquared(beta[1],beta[2])
>
> #calling of lm r-squared function
> r=summary(m)$r.squared
>
> #arranging the resulted data as the data frame of both result to compare them easily
> #d=as.matrix(m$coefficients)
> #coefficients_array=data.frame(beta,d)
> #c=data.frame(rsquare_normal,r)
> #print(coefficients_array)
> #print(c)
>
> #ggplot(data,aes(y=y,x=x))+
>  # xlab("value of x")+
>   #ylab("value of Y")+
>   #geom_point()+
>   #geom_smooth(method="lm",stat = "smooth",se=FALSE)
> print(rsquare_normal)
[1] 0.0004527364
> print(r)
[1] 0.0004527364
> library(ggplot2)
> xyz<-data.frame(x,y,z)
>
> ggplot(xyz,aes(y=z,x=x,color=y))+
+   xlab("value of x")+
+   ylab("value of z")+
+   geom_point()+
+   geom_smooth(method="lm",stat = "smooth",se=FALSE)
`geom_smooth()` using formula 'y ~ x'
>
```

FOR-QUADRATIC(y = a + b.x + c.x2)

```
x<-rpois(100, 50)

y<-rpois(100, 100)

z<-rpois(100, 150)

q=c(rep(1,100))

data=data.frame(x,y)

data$x2=data$x^2;

x2=as.matrix(data$x2)


#Using of lm model function to fit the model

m=lm(formula=y~.,data=data)

summary(m)


#R-function for calculating the value of  r-squared for below normal equations

rsquared=function(a,b,c){

  ybar=mean(y)

  total_var=sum((y-ybar)^2)

  ycap=(a+b*x+c*x2)

  residual_var=sum((ycap-ybar)^2)

  rsqr=residual_var/total_var

  return (rsqr)

}

#solving of normal equation

beta = solve(t(as.matrix(cbind(q,x,x2))) %% as.matrix(cbind(q,x,x2))) %% (t(as.matrix(cbind(q,x,x2))) %*%
as.matrix(y))

#calling of r-squared function for normal equation method

rsquare_normal=rsquared(beta[1],beta[2],beta[3])
```

#calling of lm r-squared function

r=summary(m)$r.squared

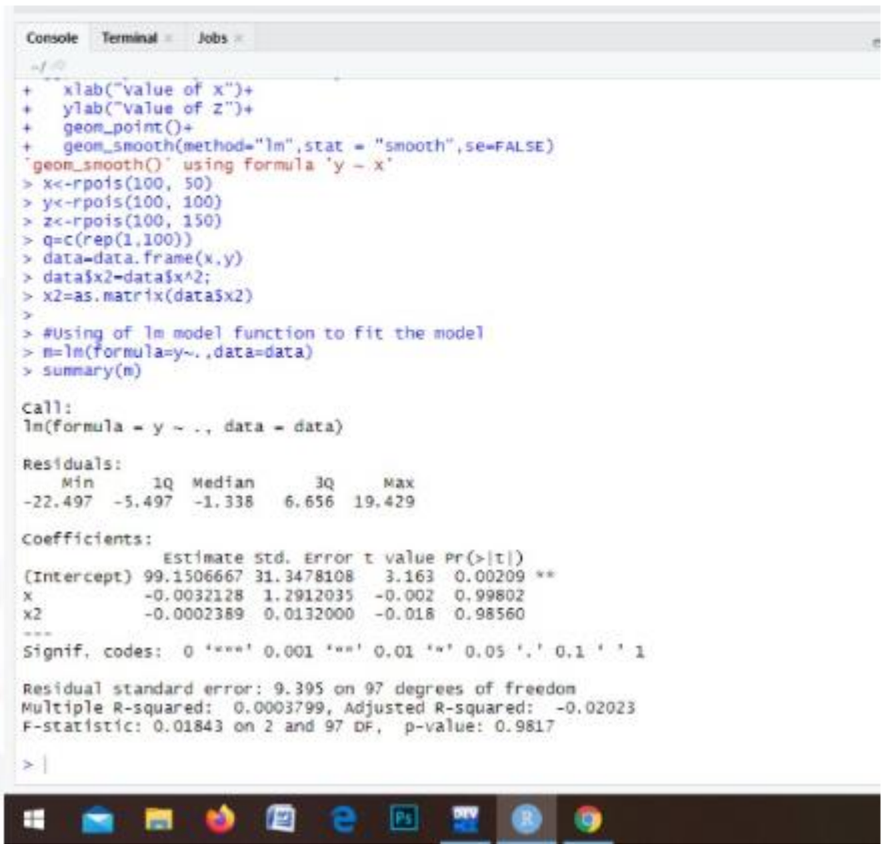
#arranging the resulted data as the data frame of both result to compare them easily

print(rsquare_normal)

print(r)

library(ggplot2)

xyz<-data.frame(x,y,z)



```
Console   Terminal   Jobs
+    xlab("Value of x")+
+    ylab("Value of Z")+
+    geom_point()+
+    geom_smooth(method="lm",stat = "smooth",se=FALSE)
`geom_smooth()` using formula 'y ~ x'
> x<-rpois(100, 50)
> y<-rpois(100, 100)
> z<-rpois(100, 150)
> q=c(rep(1,100))
> data=data.frame(x,y)
> data$x2=data$x^2;
> x2=as.matrix(data$x2)
>
> #Using of lm model function to fit the model
> m=lm(formula=y~.,data=data)
> summary(m)

Call:
lm(formula = y ~ ., data = data)

Residuals:
    Min      1Q  Median      3Q     Max 
-22.497  -5.497  -1.338   6.656  19.429 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)   
(Intercept) 99.1506667 31.3478108   3.163  0.00209 **
x           -0.0032128  1.2912035  -0.002  0.99802   
x2          -0.0002389  0.0132000  -0.018  0.98560   
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.395 on 97 degrees of freedom
Multiple R-squared:  0.0003799,	Adjusted R-squared:  -0.02023 
F-statistic: 0.01843 on 2 and 97 DF,  p-value: 0.9817

> |
```

ASSIGNMENT1
Name:Shruti Mahajan
Roll Number:101983046

```
> #calling of r-squared function for normal ec
> rsquare_normal=rsquared(beta[1],beta[2],beta
>
> #calling of lm r-squared function
> r=summary(m)$r.squared
>
> #arranging the resulted data as the data fra
> print(rsquare_normal)
[1] NA
> print(r)
[1] 0.0003799483
> library(ggplot2)
> xyz<-data.frame(x,y,z)
>
> ggplot(xyz,aes(y=z,x=x,color=y))+
+   xlab("value of x")+
```

FOR-exponential(y = a.bx)

```
x<-rpois(100, 50)

y<-rpois(100, 100)

z<-rpois(100, 150)

q=c(rep(1,100))

data=data.frame(x,y)


#Using of lm model function to fit the model

m=lm(formula=log(y,base = exp(1))~x,data=data)

summary(m)


#R-function for calculating the value of  r-squared for below normal equations

rsquared=function(a,b){

  a=exp(a)

  b=exp(b)

  ybar=mean(y)

  total_var=sum((y-ybar)^2)

  ycap=(a*b^x)

  residual_var=sum((ycap-ybar)^2)

  rsqr=residual_var/total_var
```

```
  return (rsqr)

}

#solving of normal equation

beta = solve(t(as.matrix(cbind(q,x))) %*% as.matrix(cbind(q,x))) %*% (t(as.matrix(cbind(q,x))) %*%
as.matrix(log(y),base = exp(1)));  0beta

#calling of r-squared function for normal equation method

rsquare_normal=rsquared(beta[1],beta[2])


#calling of lm r-squared function

r=summary(m)$r.squared


#arranging the resulted data as the data frame of both result to compare them easily

d=as.matrix(m$coefficients)

coefficients_array=data.frame(beta,d)

c=data.frame(rsquare_normal,r)

print(coefficients_array)

print(c)
```

```
$Rscript main.r

Call:
lm(formula = log(y, base = exp(1)) ~ x, data = data)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.34341 -0.06114  0.02309  0.06354  0.26734 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 4.519536   0.066933  67.524   <2e-16 ***
x           0.001683   0.001334   1.261     0.21    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Multiple R-squared:  0.01597,    Adjusted R-squared:  0.005929
F-statistic:  1.59 on 1 and 98 DF,  p-value: 0.2102

         [,1]
q 4.519536428
x 0.001682945
         beta           d
q 4.519536428 4.519536428
x 0.001682945 0.001682945
  rsquare_normal          r
1     0.01885869 0.01597032
```