NAME : G.NARAYANEE NIMESHIKA

REG NUM : 20BPS1111

DATE : 01.02.2022

LAB WEEK – 4 ASSESSMENT

1) AIM : To find the raw and central moments for the given data using the package and without the package

SYNTAX :

all.moments (data1, order.max = 4, absolute = TRUE)

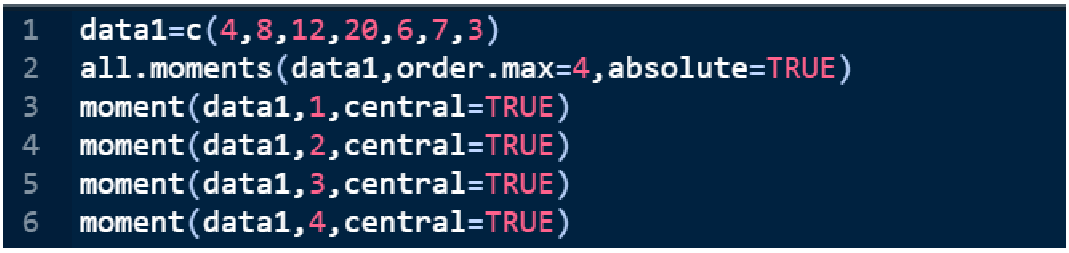
moment (data1, 1, central = TRUE)

moment (data1, 2, central = TRUE)

moment (data1, 3, central = TRUE)

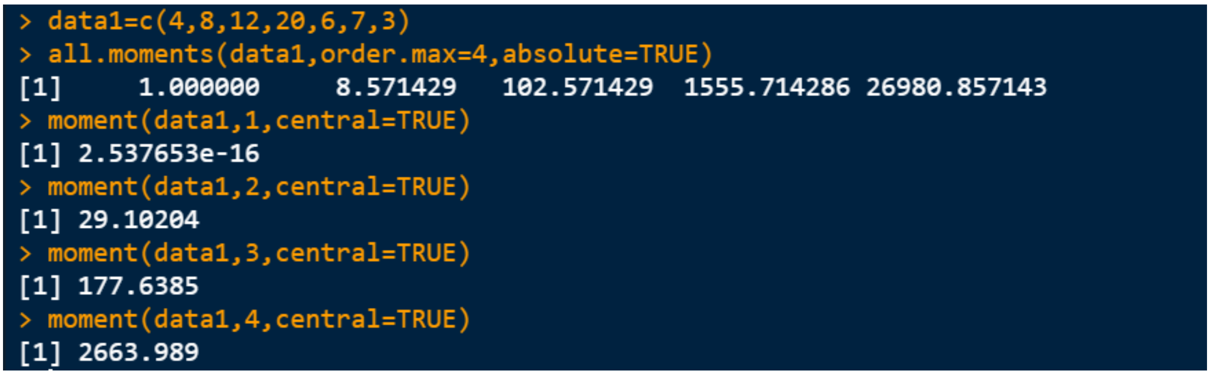
moment (data1, 4, central = TRUE)

CODE (WITH PACKAGE):



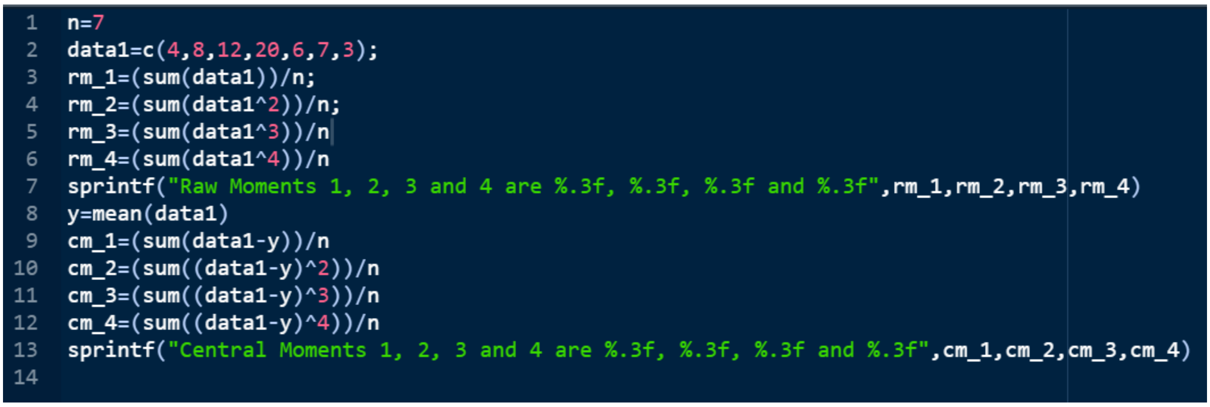
```
data1=c(4,8,12,20,6,7,3)
all.moments(data1,order.max=4,absolute=TRUE)
moment(data1,1,central=TRUE)
moment(data1,2,central=TRUE)
moment(data1,3,central=TRUE)
moment(data1,4,central=TRUE)
```

OUTPUT :



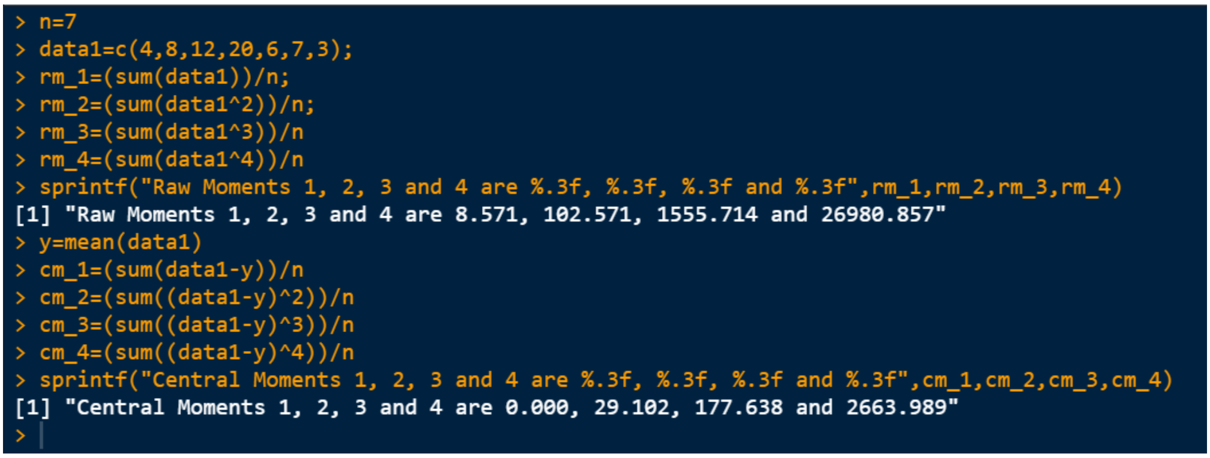
```
> data1=c(4,8,12,20,6,7,3)
> all.moments(data1,order.max=4,absolute=TRUE)
[1]     1.000000     8.571429   102.571429  1555.714286 26980.857143
> moment(data1,1,central=TRUE)
[1] 2.537653e-16
> moment(data1,2,central=TRUE)
[1] 29.10204
> moment(data1,3,central=TRUE)
[1] 177.6385
> moment(data1,4,central=TRUE)
[1] 2663.989
```

CODE (WITHOUT PACKAGE) :



```
n=7
data1=c(4,8,12,20,6,7,3);
rm_1=(sum(data1))/n;
rm_2=(sum(data1^2))/n;
rm_3=(sum(data1^3))/n
rm_4=(sum(data1^4))/n
sprintf("Raw Moments 1, 2, 3 and 4 are %.3f, %.3f, %.3f and %.3f",rm_1,rm_2,rm_3,rm_4)
y=mean(data1)
cm_1=(sum(data1-y))/n
cm_2=(sum((data1-y)^2))/n
cm_3=(sum((data1-y)^3))/n
cm_4=(sum((data1-y)^4))/n
sprintf("Central Moments 1, 2, 3 and 4 are %.3f, %.3f, %.3f and %.3f",cm_1,cm_2,cm_3,cm_4)
```

OUTPUT :



```
> n=7
> data1=c(4,8,12,20,6,7,3);
> rm_1=(sum(data1))/n;
> rm_2=(sum(data1^2))/n;
> rm_3=(sum(data1^3))/n
> rm_4=(sum(data1^4))/n
> sprintf("Raw Moments 1, 2, 3 and 4 are %.3f, %.3f, %.3f and %.3f",rm_1,rm_2,rm_3,rm_4)
[1] "Raw Moments 1, 2, 3 and 4 are 8.571, 102.571, 1555.714 and 26980.857"
> y=mean(data1)
> cm_1=(sum(data1-y))/n
> cm_2=(sum((data1-y)^2))/n
> cm_3=(sum((data1-y)^3))/n
> cm_4=(sum((data1-y)^4))/n
> sprintf("Central Moments 1, 2, 3 and 4 are %.3f, %.3f, %.3f and %.3f",cm_1,cm_2,cm_3,cm_4)
[1] "Central Moments 1, 2, 3 and 4 are 0.000, 29.102, 177.638 and 2663.989"
> 
```

2) AIM :

To find the correct correlation coefficient and the correct terms such as Σx, Σy, Σx^2, Σy^2 and Σxy

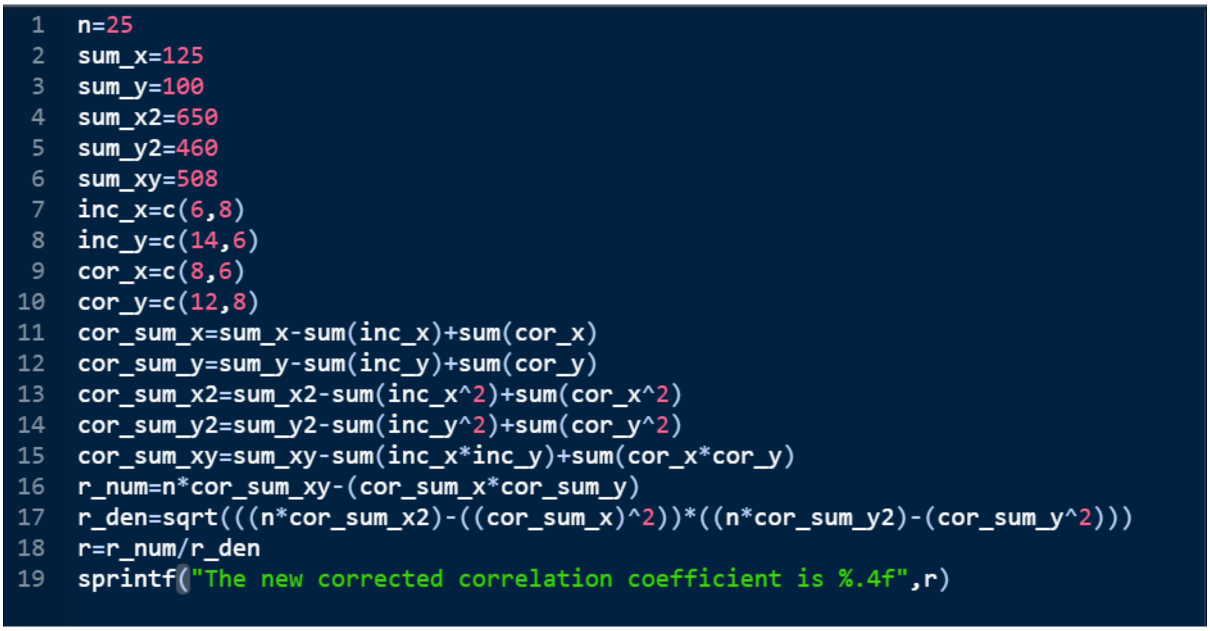
SYNTAX :

r_num = n * cor_sum_xy - (cor_sum_x * cor_sum_y)

r_den = sqrt (((n * cor_sum_x2) - ((cor_sum_x) ^ 2)) * ((n * cor_sum_y2) - (cor_sum_y ^ 2)))

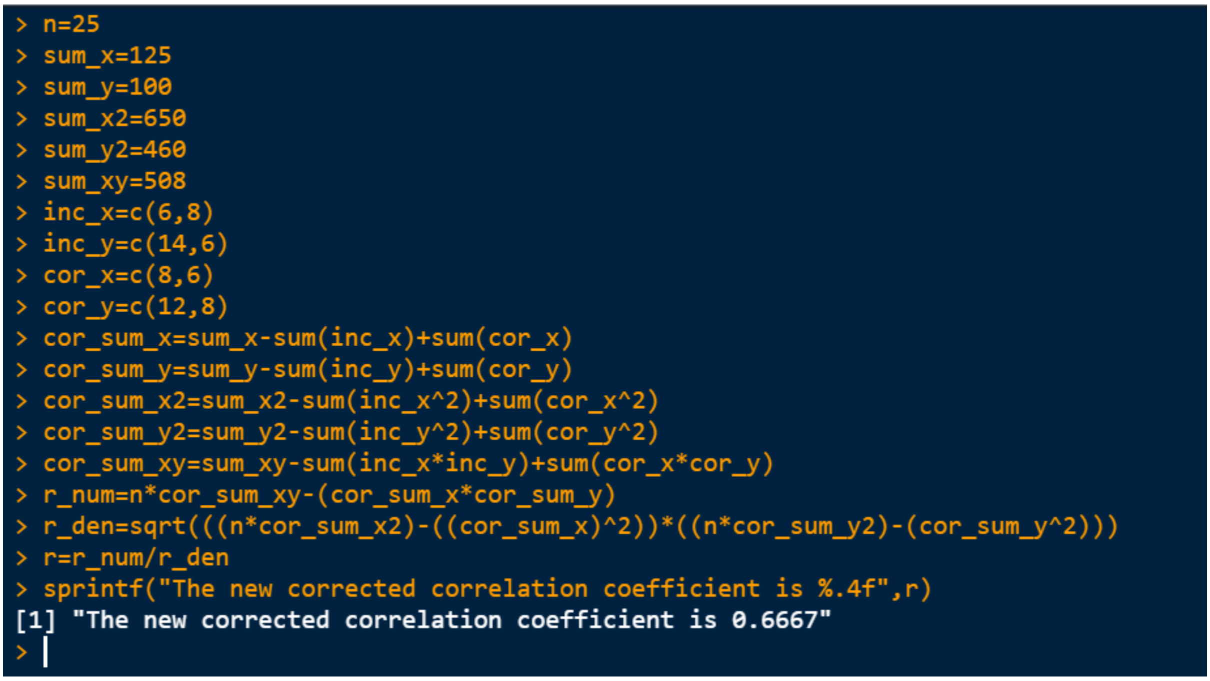
r = r_num / r_den

CODE :



```
n=25
sum_x=125
sum_y=100
sum_x2=650
sum_y2=460
sum_xy=508
inc_x=c(6,8)
inc_y=c(14,6)
cor_x=c(8,6)
cor_y=c(12,8)
cor_sum_x=sum_x-sum(inc_x)+sum(cor_x)
cor_sum_y=sum_y-sum(inc_y)+sum(cor_y)
cor_sum_x2=sum_x2-sum(inc_x^2)+sum(cor_x^2)
cor_sum_y2=sum_y2-sum(inc_y^2)+sum(cor_y^2)
cor_sum_xy=sum_xy-sum(inc_x*inc_y)+sum(cor_x*cor_y)
r_num=n*cor_sum_xy-(cor_sum_x*cor_sum_y)
r_den=sqrt(((n*cor_sum_x2)-((cor_sum_x)^2))*((n*cor_sum_y2)-(cor_sum_y^2)))
r=r_num/r_den
sprintf("The new corrected correlation coefficient is %.4f",r)
```

OUTPUT :



```
> n=25
> sum_x=125
> sum_y=100
> sum_x2=650
> sum_y2=460
> sum_xy=508
> inc_x=c(6,8)
> inc_y=c(14,6)
> cor_x=c(8,6)
> cor_y=c(12,8)
> cor_sum_x=sum_x-sum(inc_x)+sum(cor_x)
> cor_sum_y=sum_y-sum(inc_y)+sum(cor_y)
> cor_sum_x2=sum_x2-sum(inc_x^2)+sum(cor_x^2)
> cor_sum_y2=sum_y2-sum(inc_y^2)+sum(cor_y^2)
> cor_sum_xy=sum_xy-sum(inc_x*inc_y)+sum(cor_x*cor_y)
> r_num=n*cor_sum_xy-(cor_sum_x*cor_sum_y)
> r_den=sqrt(((n*cor_sum_x2)-((cor_sum_x)^2))*((n*cor_sum_y2)-(cor_sum_y^2)))
> r=r_num/r_den
> sprintf("The new corrected correlation coefficient is %.4f",r)
[1] "The new corrected correlation coefficient is 0.6667"
> |
```

INFERENCE:

The corrected correlation co-efficient is a positive correlation co-efficient

3) AIM : To find the rank correlation coefficient for the given data

SYNTAX :

x1=rank(x,ties.method="min")

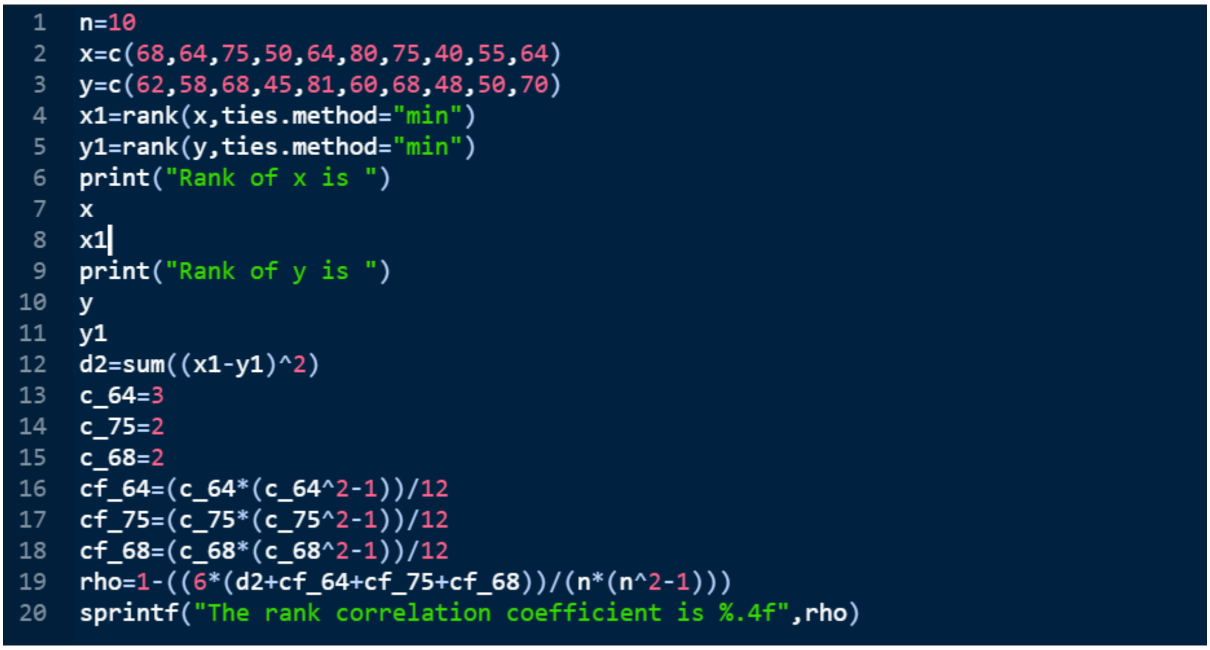
cf_64=(c_64*(c_64^2-1))/12

cf_75=(c_75*(c_75^2-1))/12

cf_68=(c_68*(c_68^2-1))/12

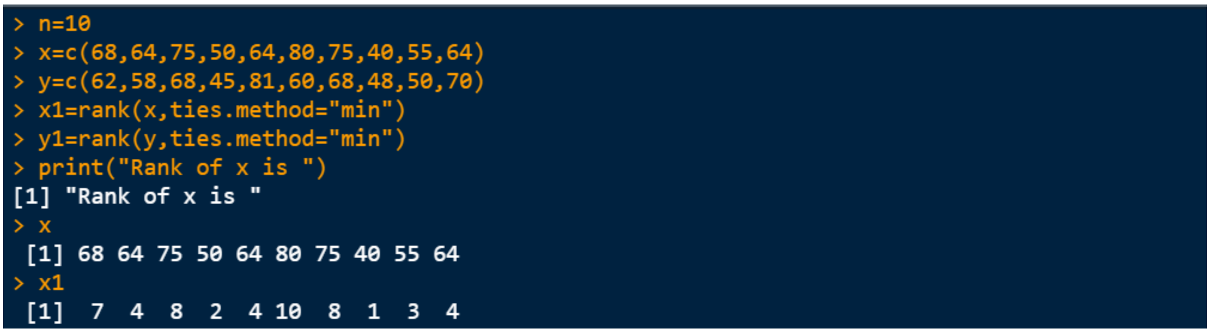
rho=1-((6*(d2+cf_64+cf_75+cf_68))/(n*(n^2-1)))

CODE :

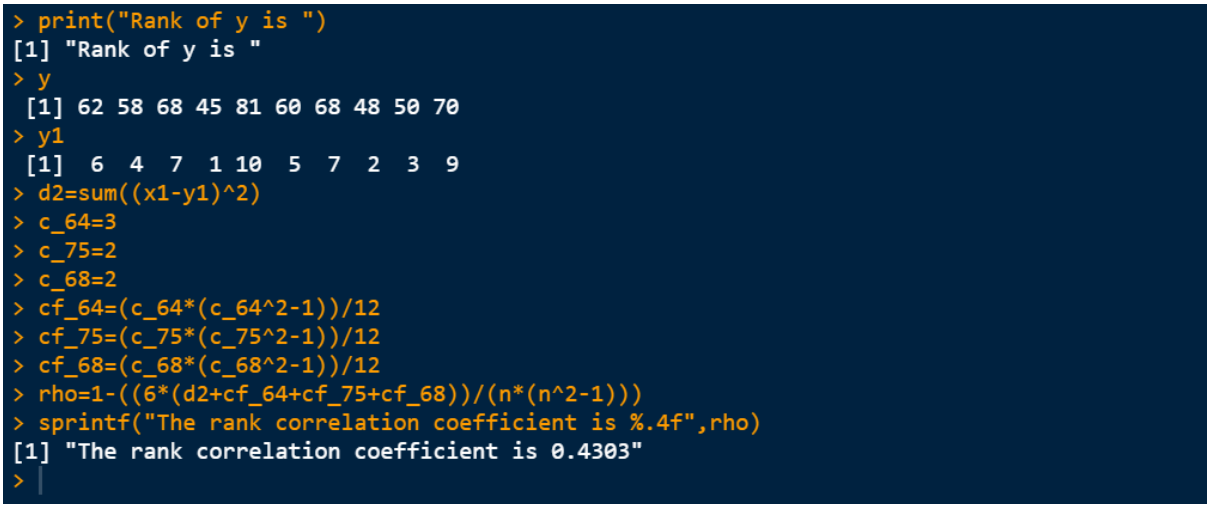


```
n=10
x=c(68,64,75,50,64,80,75,40,55,64)
y=c(62,58,68,45,81,60,68,48,50,70)
x1=rank(x,ties.method="min")
y1=rank(y,ties.method="min")
print("Rank of x is ")
x
x1
print("Rank of y is ")
y
y1
d2=sum((x1-y1)^2)
c_64=3
c_75=2
c_68=2
cf_64=(c_64*(c_64^2-1))/12
cf_75=(c_75*(c_75^2-1))/12
cf_68=(c_68*(c_68^2-1))/12
rho=1-((6*(d2+cf_64+cf_75+cf_68))/(n*(n^2-1)))
sprintf("The rank correlation coefficient is %.4f",rho)
```

OUTPUT :



```
> n=10
> x=c(68,64,75,50,64,80,75,40,55,64)
> y=c(62,58,68,45,81,60,68,48,50,70)
> x1=rank(x,ties.method="min")
> y1=rank(y,ties.method="min")
> print("Rank of x is ")
[1] "Rank of x is "
> x
 [1] 68 64 75 50 64 80 75 40 55 64
> x1
 [1]  7  4  8  2  4 10  8  1  3  4
```



```
> print("Rank of y is ")
[1] "Rank of y is "
> y
 [1] 62 58 68 45 81 60 68 48 50 70
> y1
 [1]  6  4  7  1 10  5  7  2  3  9
> d2=sum((x1-y1)^2)
> c_64=3
> c_75=2
> c_68=2
> cf_64=(c_64*(c_64^2-1))/12
> cf_75=(c_75*(c_75^2-1))/12
> cf_68=(c_68*(c_68^2-1))/12
> rho=1-((6*(d2+cf_64+cf_75+cf_68))/(n*(n^2-1)))
> sprintf("The rank correlation coefficient is %.4f",rho)
[1] "The rank correlation coefficient is 0.4303"
> 
```

4) AIM : To find and interpret the correlation coefficient r and also include a plot of the given data.

SYNTAX :

r_num = n * sum_xy - (sum_x * sum_y)

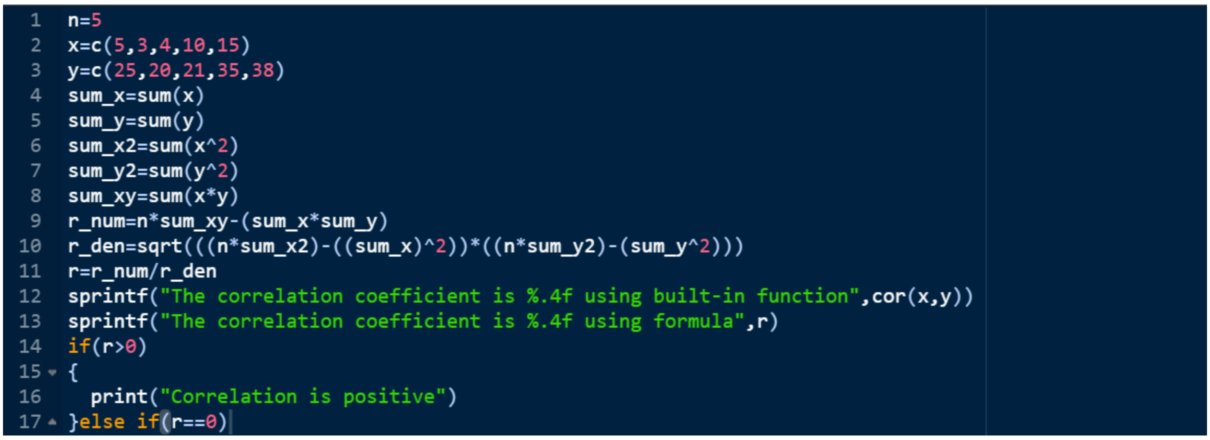
r_den = sqrt (((n * sum_x2) - ((sum_x) ^ 2)) * ((n * sum_y2) - (sum_y^2)))

r = r_num / r_den

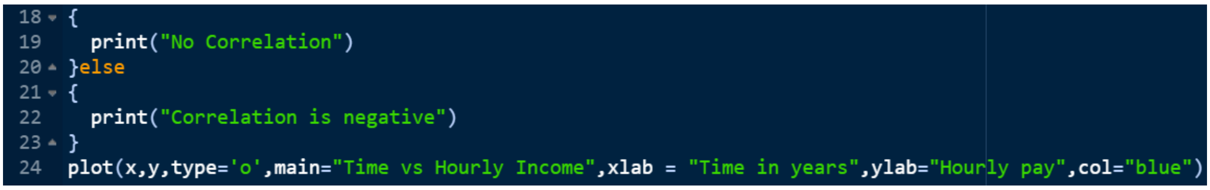
cor(x, y) [BUILT-IN FUNCTION]

plot (x , y , type = 'o' , main = " Time vs Hourly Income " , xlab = " Time in years " , ylab = " Hourly pay " , col = " blue ")

CODE :

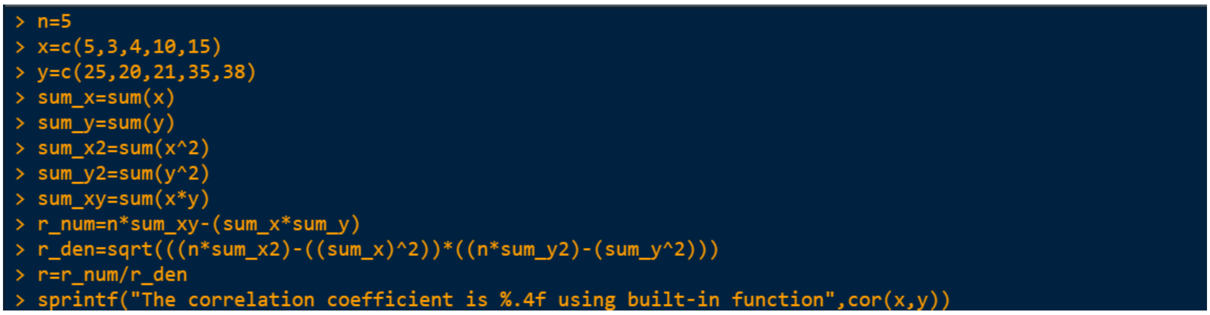


```
n=5
x=c(5,3,4,10,15)
y=c(25,20,21,35,38)
sum_x=sum(x)
sum_y=sum(y)
sum_x2=sum(x^2)
sum_y2=sum(y^2)
sum_xy=sum(x*y)
r_num=n*sum_xy-(sum_x*sum_y)
r_den=sqrt(((n*sum_x2)-((sum_x)^2))*((n*sum_y2)-(sum_y^2)))
r=r_num/r_den
sprintf("The correlation coefficient is %.4f using built-in function",cor(x,y))
sprintf("The correlation coefficient is %.4f using formula",r)
if(r>0)
{
  print("Correlation is positive")
}else if(r==0)
```

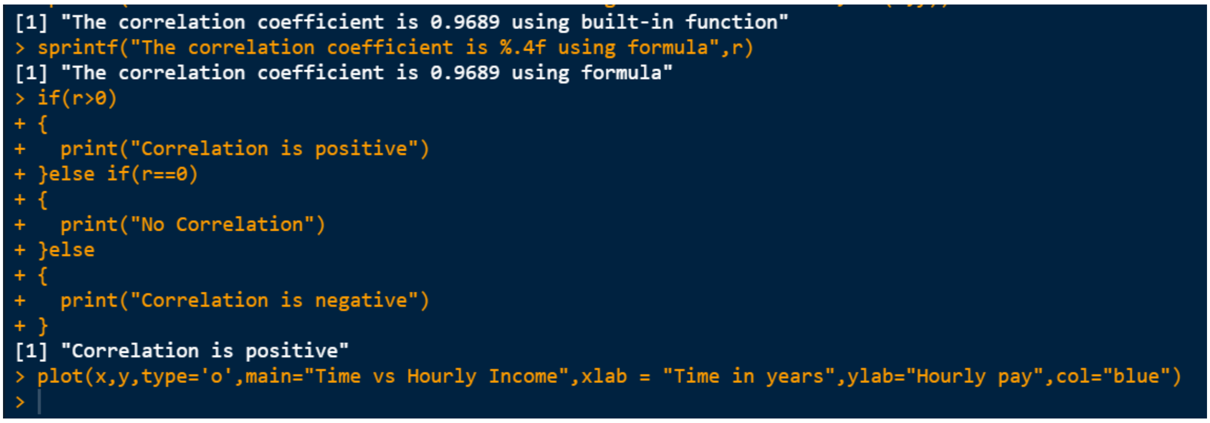


```
{
  print("No Correlation")
}else
{
  print("Correlation is negative")
}
plot(x,y,type='o',main="Time vs Hourly Income",xlab = "Time in years",ylab="Hourly pay",col="blue")
```

OUTPUT :

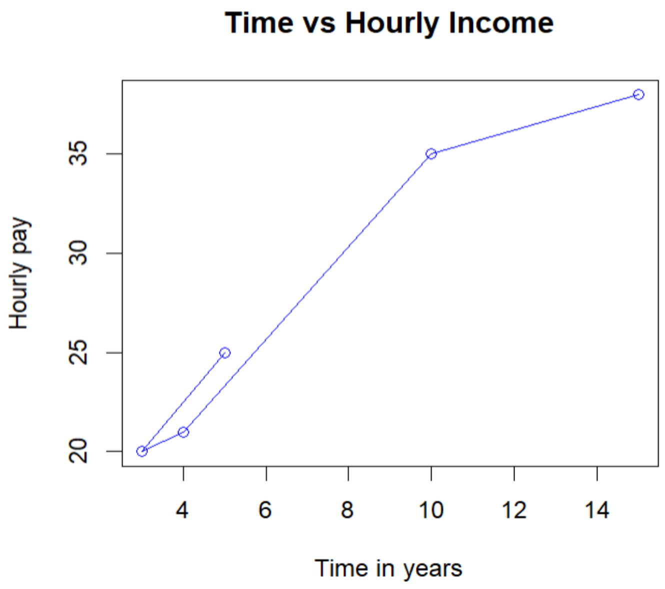


```
> n=5
> x=c(5,3,4,10,15)
> y=c(25,20,21,35,38)
> sum_x=sum(x)
> sum_y=sum(y)
> sum_x2=sum(x^2)
> sum_y2=sum(y^2)
> sum_xy=sum(x*y)
> r_num=n*sum_xy-(sum_x*sum_y)
> r_den=sqrt(((n*sum_x2)-((sum_x)^2))*((n*sum_y2)-(sum_y^2)))
> r=r_num/r_den
> sprintf("The correlation coefficient is %.4f using built-in function",cor(x,y))
```



```
[1] "The correlation coefficient is 0.9689 using built-in function"
> sprintf("The correlation coefficient is %.4f using formula",r)
[1] "The correlation coefficient is 0.9689 using formula"
> if(r>0)
+ {
+   print("Correlation is positive")
+ }else if(r==0)
+ {
+   print("No Correlation")
+ }else
+ {
+   print("Correlation is negative")
+ }
[1] "Correlation is positive"
> plot(x,y,type='o',main="Time vs Hourly Income",xlab = "Time in years",ylab="Hourly pay",col="blue")
> 
```

GRAPH:



INFERENCE:

The correlation co-efficient is a positive correlation co-efficient

5) AIM : To find the correlation co-effiecient by using the Karl Pearson’s formula

SYNTAX :

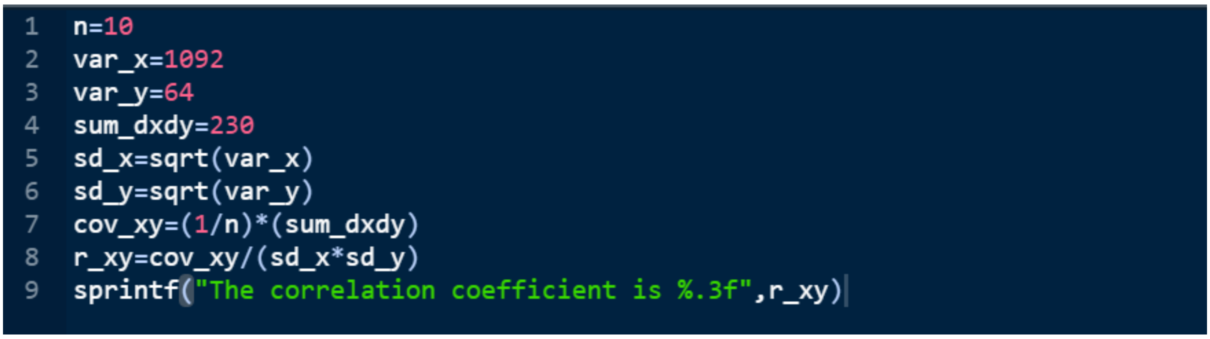
sd_x=sqrt(var_x)

sd_y=sqrt(var_y)

cov_xy=(1/n)*(sum_dxdy)

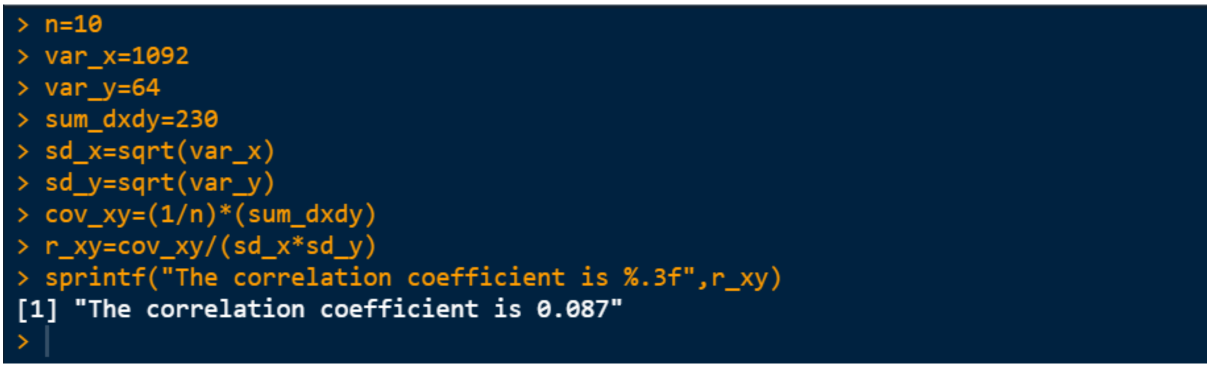
r_xy=cov_xy/(sd_x*sd_y)

CODE :



```
n=10
var_x=1092
var_y=64
sum_dxdy=230
sd_x=sqrt(var_x)
sd_y=sqrt(var_y)
cov_xy=(1/n)*(sum_dxdy)
r_xy=cov_xy/(sd_x*sd_y)
sprintf("The correlation coefficient is %.3f",r_xy)
```

OUTPUT :



```
> n=10
> var_x=1092
> var_y=64
> sum_dxdy=230
> sd_x=sqrt(var_x)
> sd_y=sqrt(var_y)
> cov_xy=(1/n)*(sum_dxdy)
> r_xy=cov_xy/(sd_x*sd_y)
> sprintf("The correlation coefficient is %.3f",r_xy)
[1] "The correlation coefficient is 0.087"
> 
```

INFERENCE:

The correlation co-efficient is a positive correlation co-efficient