NAME : VISHAL KUMAR MAHATHA
REG. NO. : 20BRS1168

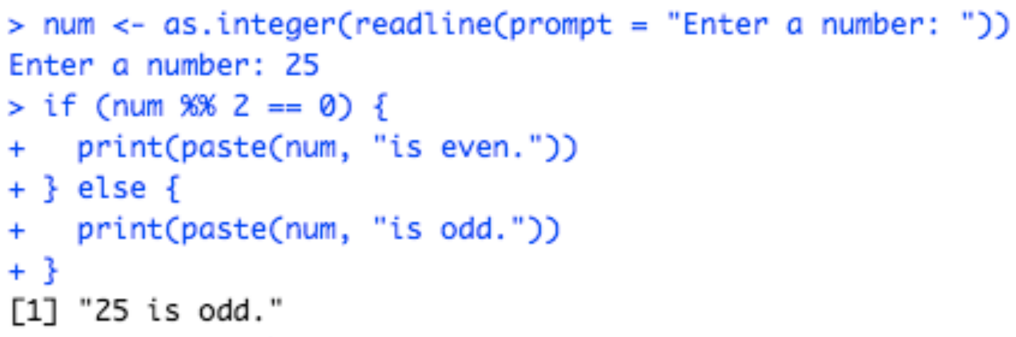
IOT LAB ASSIGNMENT 8

1. Check the Given number odd or even?

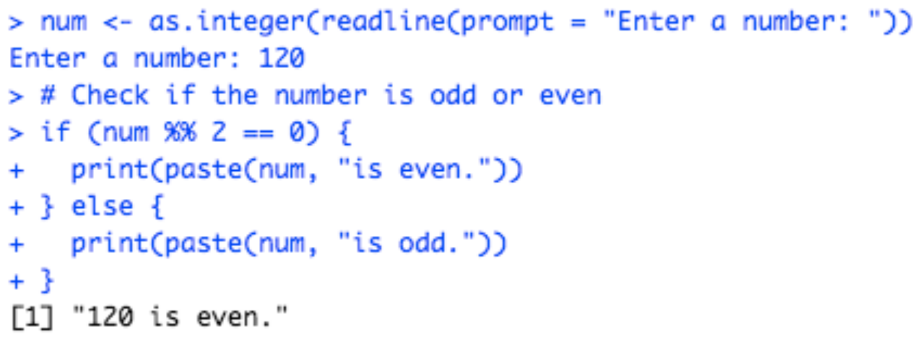
CODE :

```
# Input a number
num <- as.integer(readline(prompt = "Enter a number: "))

# Check if the number is odd or even
if (num %% 2 == 0) {
  print(paste(num, "is even."))
} else {
  print(paste(num, "is odd."))
}
```

OUTPUT :



```
> num <- as.integer(readline(prompt = "Enter a number: "))
Enter a number: 25
> if (num %% 2 == 0) {
+   print(paste(num, "is even."))
+ } else {
+   print(paste(num, "is odd."))
+ }
[1] "25 is odd."
```



```
> num <- as.integer(readline(prompt = "Enter a number: "))
Enter a number: 120
> # Check if the number is odd or even
> if (num %% 2 == 0) {
+   print(paste(num, "is even."))
+ } else {
+   print(paste(num, "is odd."))
+ }
[1] "120 is even."
```

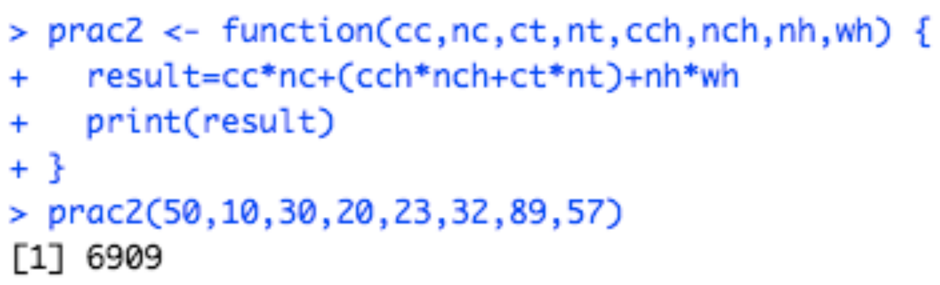
2. An university is setting up a new lab at their premises. Design an algorithm and write R studio code to determine the approximate cost to be spent for setting up the lab. Cost for setting the lab is sum of cost of computers, cost of furniture and labor cost. Use the following formulae for solving the problem:

- Cost of computer = cost of one computer * number of computers
- Cost of furniture = Number of tables * cost of one table + number of chairs * cost of one chair
- Labour cost = number of hours worked * wages per hour

CODE :

```
prac2 <- function(cc,nc,ct,nt,cch,nch,nh,wh) {
  result=cc*nc+(cch*nch+ct*nt)+nh*wh
  print(result)
}
prac2(50,10,30,20,23,32,89,57)
```

OUTPUT :



```
> prac2 <- function(cc,nc,ct,nt,cch,nch,nh,wh) {
+   result=cc*nc+(cch*nch+ct*nt)+nh*wh
+   print(result)
+ }
> prac2(50,10,30,20,23,32,89,57)
[1] 6909
```

3. Given the number of hours and minutes browsed, write a program to calculate bill for Internet Browsing in a browsing center. The conditions are given below.

(a) 1 Hour Rs.50

(b) 1 minute Re. 1

(c) Rs. 200 for every five hours

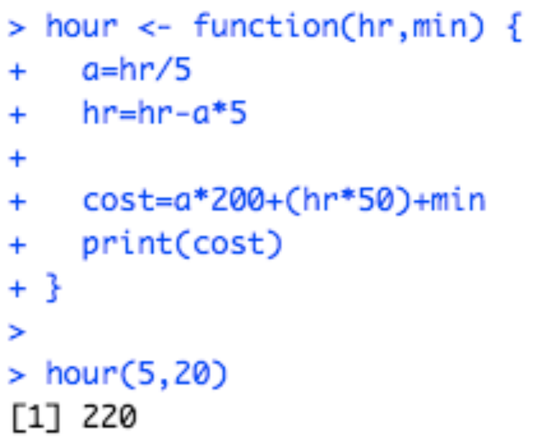
Boundary condition: User can only browse for a maximum of 7 hours

Check boundary conditions

CODE :

```
hour <- function(hr,min) {
  a=hr/5
  hr=hr-a*5


  cost=a*200+(hr*50)+min
  print(cost)
}
hour(5,20)
```

OUTPUT :



```
> hour <- function(hr,min) {
+   a=hr/5
+   hr=hr-a*5
+
+   cost=a*200+(hr*50)+min
+   print(cost)
+ }
>
> hour(5,20)
[1] 220
```

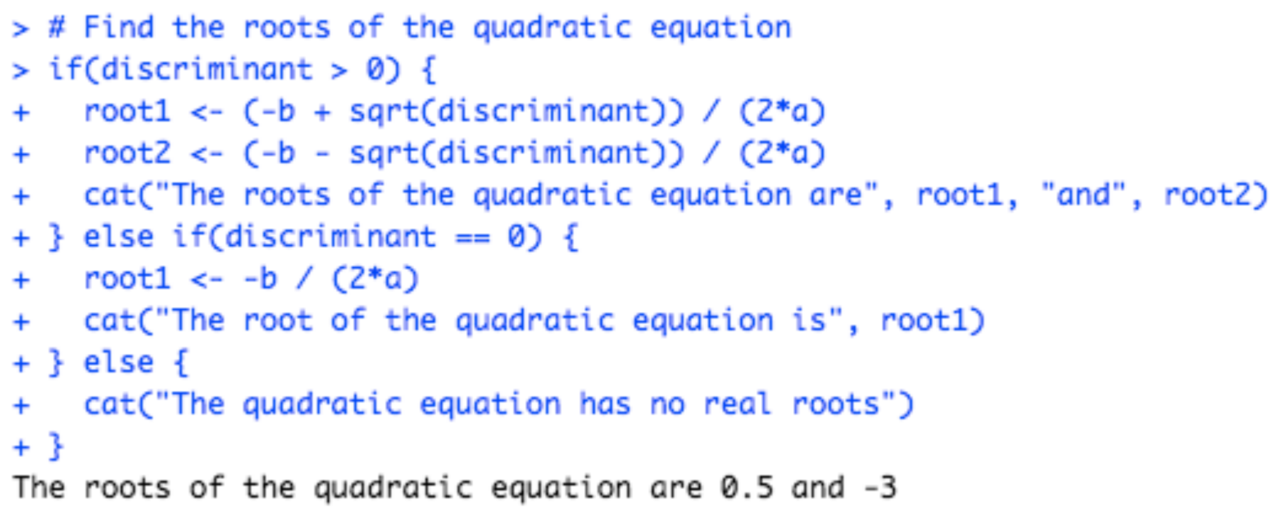
4. Write R-studio code in finding the roots of a quadratic equation?

CODE :

```
# Define the coefficients of the quadratic equation
a <- 2
```

```
b <- 5

c <- -3

# Calculate the discriminant

discriminant <- b^2 - 4*a*c

# Find the roots of the quadratic equation

if(discriminant > 0) {

  root1 <- (-b + sqrt(discriminant)) / (2*a)

  root2 <- (-b - sqrt(discriminant)) / (2*a)

  cat("The roots of the quadratic equation are", root1, "and", root2)

} else if(discriminant == 0) {

  root1 <- -b / (2*a)

  cat("The root of the quadratic equation is", root1)

} else {

  cat("The quadratic equation has no real roots")

}
```

OUTPUT :



```
> # Find the roots of the quadratic equation
> if(discriminant > 0) {
+   root1 <- (-b + sqrt(discriminant)) / (2*a)
+   root2 <- (-b - sqrt(discriminant)) / (2*a)
+   cat("The roots of the quadratic equation are", root1, "and", root2)
+ } else if(discriminant == 0) {
+   root1 <- -b / (2*a)
+   cat("The root of the quadratic equation is", root1)
+ } else {
+   cat("The quadratic equation has no real roots")
+ }
The roots of the quadratic equation are 0.5 and -3
```

5. Write a program in R studio to segregate student based on their CGPA. The details are as follows:

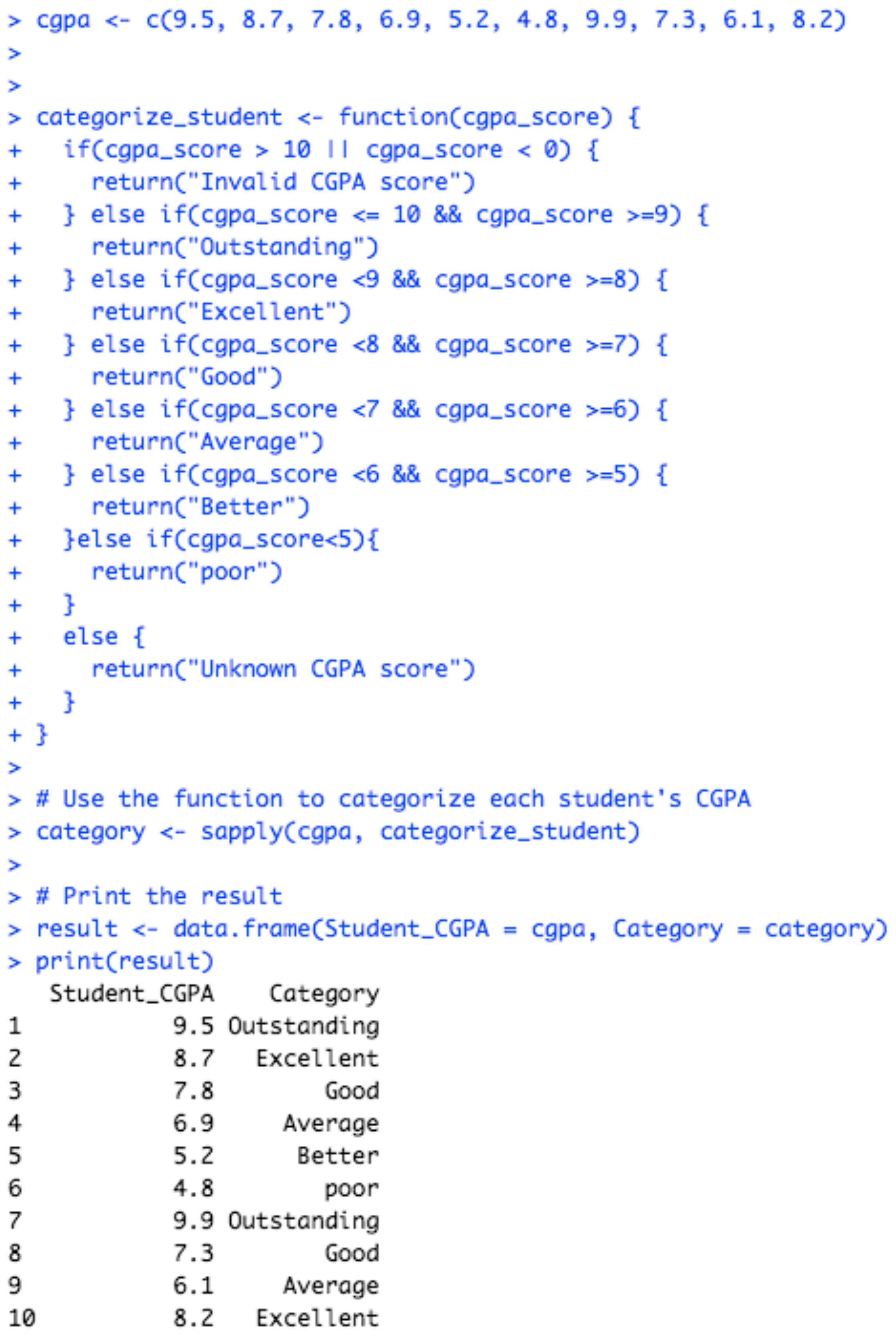
<=9 CGPA <=10	- outstanding
<=8 CGPA <9	- excellent
<=7 CGPA <8	- good
<=6 CGPA <7	- average
<=5 CGPA <6	- better
CGPA<5	- poor

CODE :

```
cgpa <- c(9.5, 8.7, 7.8, 6.9, 5.2, 4.8, 9.9, 7.3, 6.1, 8.2)


categorize_student <- function(cgpa_score) {
  if(cgpa_score > 10 || cgpa_score < 0) {
    return("Invalid CGPA score")
  } else if(cgpa_score <= 10 && cgpa_score >=9) {
    return("Outstanding")
  } else if(cgpa_score <9 && cgpa_score >=8) {
    return("Excellent")
  } else if(cgpa_score <8 && cgpa_score >=7) {
    return("Good")
  } else if(cgpa_score <7 && cgpa_score >=6) {
    return("Average")
  } else if(cgpa_score <6 && cgpa_score >=5) {
    return("Better")
  }else if(cgpa_score<5){
    return("poor")
  }
  else {
    return("Unknown CGPA score")
  }
}

# Use the function to categorize each student's CGPA
category <- sapply(cgpa, categorize_student)

# Print the result
result <- data.frame(Student_CGPA = cgpa, Category = category)
```

print(result)

OUTPUT :



```
> cgpa <- c(9.5, 8.7, 7.8, 6.9, 5.2, 4.8, 9.9, 7.3, 6.1, 8.2)
>
>
> categorize_student <- function(cgpa_score) {
+   if(cgpa_score > 10 || cgpa_score < 0) {
+     return("Invalid CGPA score")
+   } else if(cgpa_score <= 10 && cgpa_score >=9) {
+     return("Outstanding")
+   } else if(cgpa_score <9 && cgpa_score >=8) {
+     return("Excellent")
+   } else if(cgpa_score <8 && cgpa_score >=7) {
+     return("Good")
+   } else if(cgpa_score <7 && cgpa_score >=6) {
+     return("Average")
+   } else if(cgpa_score <6 && cgpa_score >=5) {
+     return("Better")
+   }else if(cgpa_score<5){
+     return("poor")
+   }
+   else {
+     return("Unknown CGPA score")
+   }
+ }
>
> # Use the function to categorize each student's CGPA
> category <- sapply(cgpa, categorize_student)
>
> # Print the result
> result <- data.frame(Student_CGPA = cgpa, Category = category)
> print(result)
   Student_CGPA    Category
1           9.5 Outstanding
2           8.7   Excellent
3           7.8        Good
4           6.9     Average
5           5.2      Better
6           4.8        poor
7           9.9 Outstanding
8           7.3        Good
9           6.1     Average
10          8.2   Excellent
```

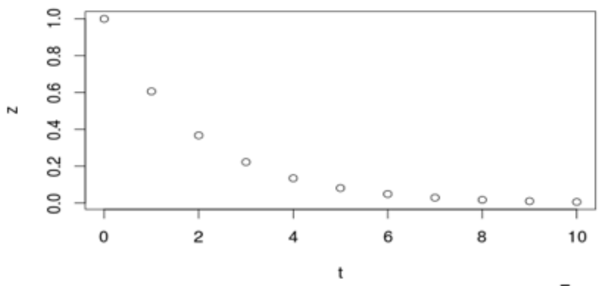
Practice 6

Plotting graph in R

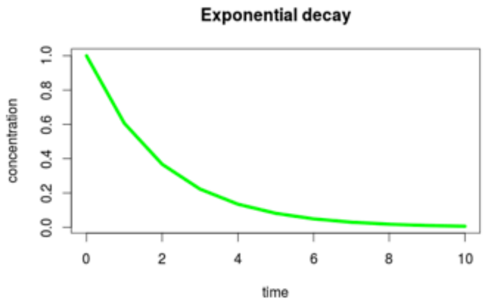
➢t=0:10

➢ z= exp(-t/2)

➢ plot(t,z)



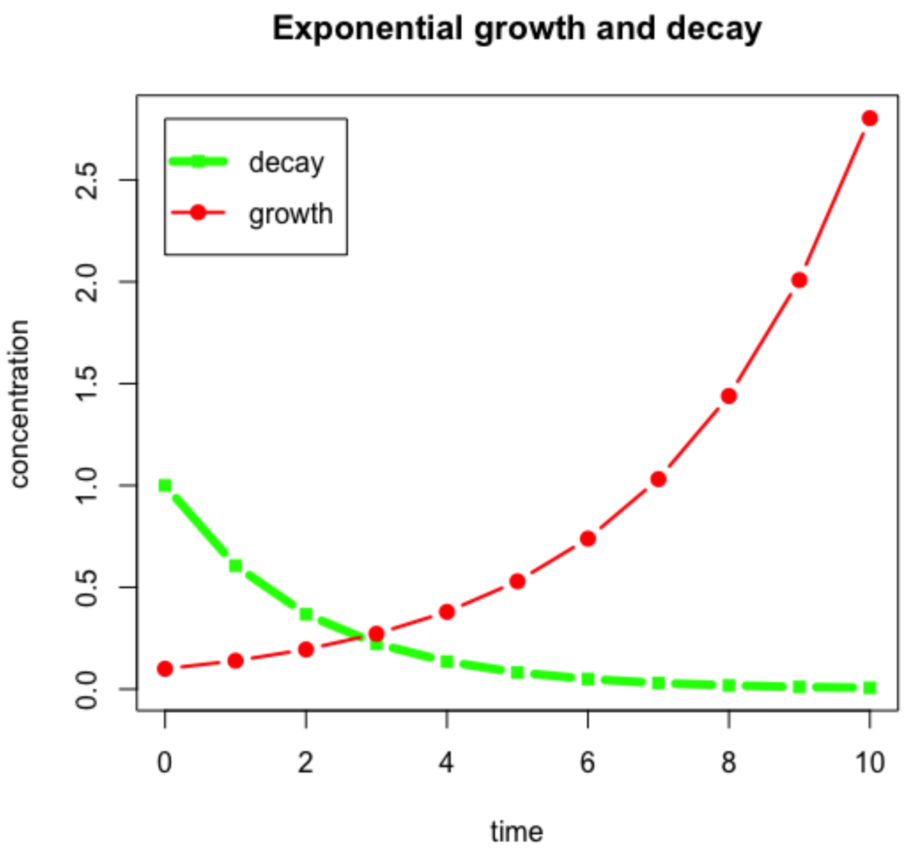
➢plot(t,z, type="l", col="green", lwd=5, xlab="time", ylab="concentration", main="Exponential decay")



CODE :

```
t=0:10
z= exp(-t/2)
plot(t,z)

plot(t,z, type="l", col="green", lwd=5, xlab="time", ylab="concentration",
main="Exponential decay")
w = 0.1*exp(t/3)
plot(t,z, type="l", col="green", lwd=5, xlab="time", ylab="concentration")
lines(t, w, col="red", lwd=2)

title("Exponential growth and decay")
legend(2,1,c("decay","growth"), lwd=c(5,2), col=c("green","red"),
y.intersp=1.5)

plot(t,z, type="b", col="green", lwd=5, pch=15, xlab="time",
ylab="concentration", ylim=range(w,z))
lines(t, w, type="b", col="red", lwd=2, pch=19)

title("Exponential growth and decay")
legend(0,2.8,c("decay","growth"), lwd=c(5,2), col=c("green","red"),
pch=c(15,19), y.intersp=1.5)
```

OUTPUT :

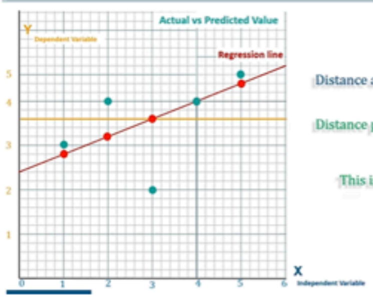


EXERCISE 1

Exercise-I

- For the given data set perform the linear regression analysis and find the regression line. Compute root mean square error and find out the number of points that will meet the regression lines using R studio.
- Plot the graphs for the given data & with the obtained regression line

	x	y
	1	3
	2	4
	3	2
	4	4
	5	5
mean	3	3.6



Distance actual - mean

vs

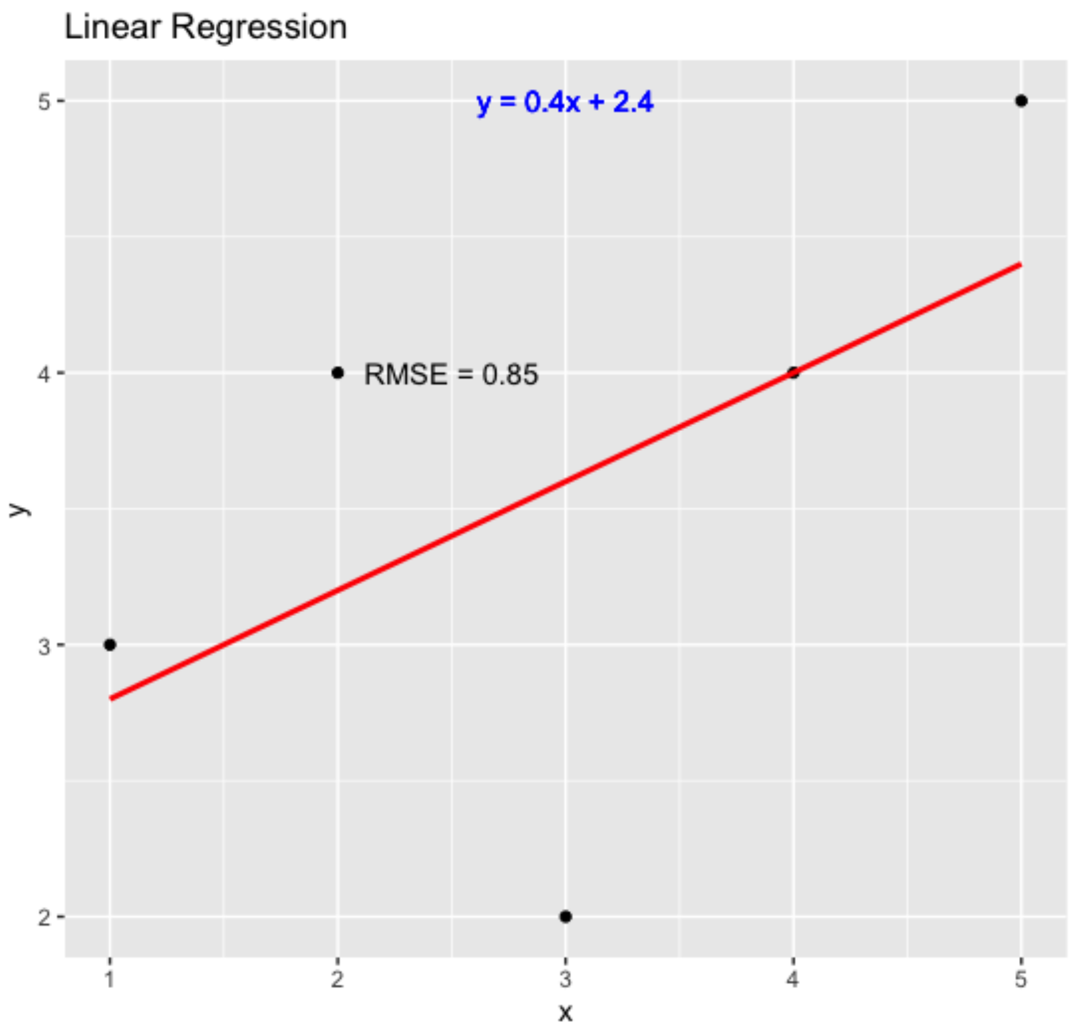
Distance predicted - mean

This is nothing but $R^2 = \dfrac{\Sigma\,(y_p - \bar{y})^2}{\Sigma\,(y - \bar{y})^2}$

CODE :

```
data <- data.frame(x = c(1, 2, 3, 4, 5), y = c(3, 4, 2, 4, 5))
model <- lm(y ~ x, data = data)

slope <- coef(model)[2]
intercept <- coef(model)[1]
regression_line <- paste0("y = ", round(slope, 2), "x + ", round(intercept, 2))
predicted_y <- predict(model, data)
rmse <- sqrt(mean((predicted_y - data$y)^2))
library(ggplot2)
ggplot(data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Linear Regression", x = "x", y = "y") +
  geom_text(x = 3, y = 5, label = regression_line, color = "blue") +
  annotate("text", x = 2.5, y = 4, label = paste0("RMSE = ", round(rmse, 2)))
```

OUTPUT :


Linear Regression
y = 0.4x + 2.4
RMSE = 0.85
y
x