SET -1

1. Write R code to implement to generate the sequence 1, 0, 3, 0, 5, 0, 7, . . . , 0, 49.

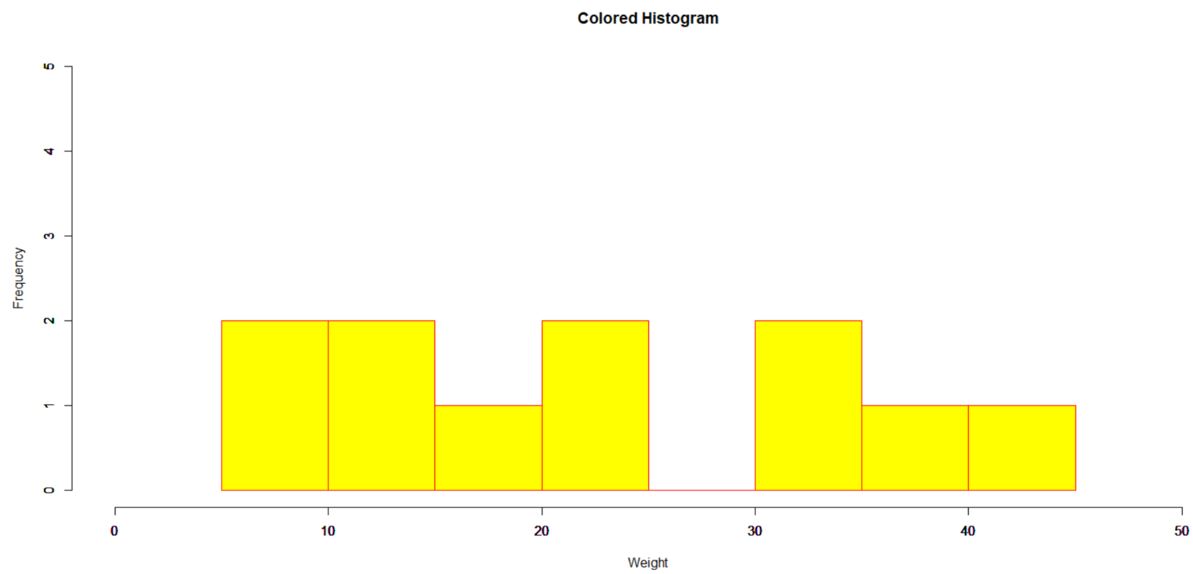
```
> x<-seq(1:49)
> x[x%%2==0]<-0
> x
 [1]  1  0  3  0  5  0  7  0  9  0 11  0 13  0 15  0 17  0 19  0 21  0 23  0 25  0 27  0 29  0 31  0 33  0
35  0 37  0 39  0 41  0 43  0 45  0 47  0 49
```

OR

```
sequence_length <- 49
sequence <- numeric(sequence_length)
for (i in 1:sequence_length) {
  if (i %% 2 == 0) {
    sequence[i] <- 0
  } else {
    sequence[i] <- i
  }
}
print(sequence)

 [1]  1  0  3  0  5  0  7  0  9  0 11  0 13  0 15  0 17  0 19  0 21  0 23  0 25  0 27  0 29  0 31  0 33  0
35  0 37  0 39  0 41  0 43  0 45  0 47  0 49
```

2. Given x = c(9,13,21,8,36,22,12,41,31,33,19). Write R command to draw a simple histogram, having label of x-axis = ”Weight”, label of y-axis = “ Frequency”, color of bars = ”yellow”, border color of bars = ”red“, title = “Colored Histogram”, limits of x-axis are 0 to 50, limits of y-axis are 0 to 5 and draw the output graph.

```
x <- c(9, 13, 21, 8, 36, 22, 12, 41, 31, 33, 19)
# Set up the plot area
hist(x,
     main = "Colored Histogram",
     xlab = "Weight",
     ylab = "Frequency",
     xlim = c(0, 50),
     ylim = c(0, 5),
     col = "yellow",
     border = "red"
)
```



3. write R code to fit a straight line y = a + bx to the following data by the method of least squares

x 0 1 3 6 8
y 1 3 2 5 4

```
x <- c(0, 1, 3, 6, 8)
y <- c(1, 3, 2, 5, 4)

# Create a data frame with the x and y values
data <- data.frame(x, y)

# Fit the linear regression model
model <- lm(y ~ x, data = data)

# Get the estimated coefficients
a <- coef(model)[1]
b <- coef(model)[2]

# Print the coefficients
cat("Estimated coefficients:\n")
Estimated coefficients:
cat("a =", a, "\n")
a = 1.646018
cat("b =", b, "\n")
b = 0.3761062
```

OR

```
> x <- c(0, 1, 3, 6, 8)
> y <- c(1, 3, 2, 5, 4)
> sqx <- x^2
> xy <- x*y
> df <- data.frame(x,y,xy,sqx)
> df
  x y xy sqx
1 0 1  0   0
2 1 3  3   1
3 3 2  6   9
4 6 5 30  36
5 8 4 32  64
> reg <- lm(y~x)
> print(reg)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
     1.6460       0.3761
```

4. Write an R program to generate first 30 prime numbers.

```
# Function to check if a number is prime
is_prime <- function(n) {
  if (n <= 1) {
    return(FALSE)
  }

  for (i in 2:sqrt(n)) {
    if (n %% i == 0) {
      return(FALSE)
    }
  }

  return(TRUE)
}
```

```r
# Generate the first 30 prime numbers
count <- 0
n <- 2
prime_numbers <- numeric(30)

while (count < 30) {
  if (is_prime(n)) {
    count <- count + 1
    prime_numbers[count] <- n
  }
  n <- n + 1
}

# Print the prime numbers
cat("First 30 prime numbers:\n")
print(prime_numbers)
```

Output:

```
First 30 prime numbers:
 [1]   3   5   7  11  13  17  19  23  29  31  37  41  43  47  53  59  61  67  71  73  79  83  89  97 101
103 107 109 113 127
```

SET - 4

1. Write R code to implement the problem :
Twenty students,graduates,undergraduates were enrolled in statistics course their ages were 18,19,19,19,19,20,20,20,20,20,21,21,21,21,22,23,24,27,30,36.

   (a) Find the median and mean of all the students
   (b) Find the median age of all students under age 25.
   (c) Find the modal age of all students.

```
# Student ages
ages <- c(18, 19, 19, 19, 19, 20, 20, 20, 20, 20, 21, 21, 21, 21, 22, 23, 24, 27, 30, 36)

# (a) Median and mean of all students
median_all <- median(ages)
mean_all <- mean(ages)

cat("Median of all students:", median_all, "\n")
cat("Mean of all students:", mean_all, "\n")

# (b) Median age of students under age 25
median_under_25 <- median(ages[ages < 25])

cat("Median age of students under age 25:", median_under_25, "\n")

# (c) Modal age of all students
mode<-which.max(table(ages))

cat("Modal age of all students:", "\n")
print(mode)
```

Output:

```
Median of all students: 20.5
Mean of all students: 22
Median age of students under age 25: 20
Modal age of all students:
20
 3
```

2. Create a 2x3 matrix M using R programming with following specifications. Values of the matrix should be randomly selected between 10 and 30. Elements of the matrix should be arranged in row order and Create a vector V with three values and add V as third row of the matrix.

```
# Create a 2x3 matrix M with random values between 10 and 30
M <- matrix(sample(10:30, 2*3, replace = TRUE), nrow = 2, ncol=3, byrow = TRUE)

# Print the matrix
print(M)

# Create a vector V with three values
V <- c(20, 25, 30)

# Add V as the third row of M
M <- rbind(M, V)

# Print the matrix
print(M)
```

Output:

```
     [,1] [,2] [,3]
[1,]   11   12   29
[2,]   14   16   20

  [,1] [,2] [,3]
    11   12   29
    14   16   20
V   20   25   30
```

3. Write R code for the following problem Heights(in cm) of father and son are given as follows fit a regression line predict the height of son given the height of father.

Father(X) 152 155 157 160 161 164 165 150
Son(Y) 156 158 159 160 162 161 164 154

```
father <- c(152, 155, 157, 160, 161, 164, 165, 150)
son <- c(156, 158, 159, 160, 162, 161, 164, 154)

# Create a data frame with the father and son heights
heights <- data.frame(father, son)
```

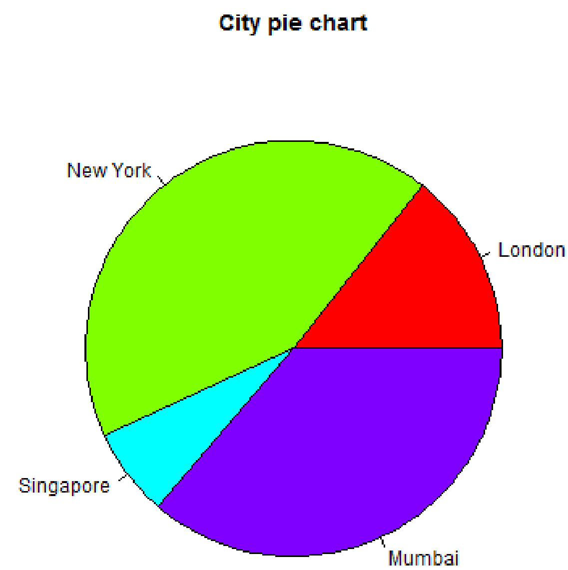
```
# Fit the linear regression model
model <- lm(son ~ father, data = heights)


# Get the estimated coefficients
intercept <- coef(model)[1]
slope <- coef(model)[2]

# Print the coefficients
cat("Estimated coefficients:\n")
cat("Intercept =", intercept, "\n")
cat("Slope =", slope, "\n")

# Predict the height of son for a given height of father
father_height <- 158
predicted_height <- predict(model, newdata = data.frame(father = father_height))

cat("Predicted height of son for father's height", father_height, "is", predicted_height, "\n")
```

Output:

```
Estimated coefficients:
Intercept = 68.85577
Slope = 0.5721154
Predicted height of son for father's height 158 is 159.25
```

4. Write R command to create the Pie Chart with 4 random rainbow colors, title as “City Pie Chart” and draw the output graph for the following data. x consisting of values 21, 62, 10, 53 and labels to corresponding values are ”London”, ”New York”, ”Singapore”, ”Mumbai”.

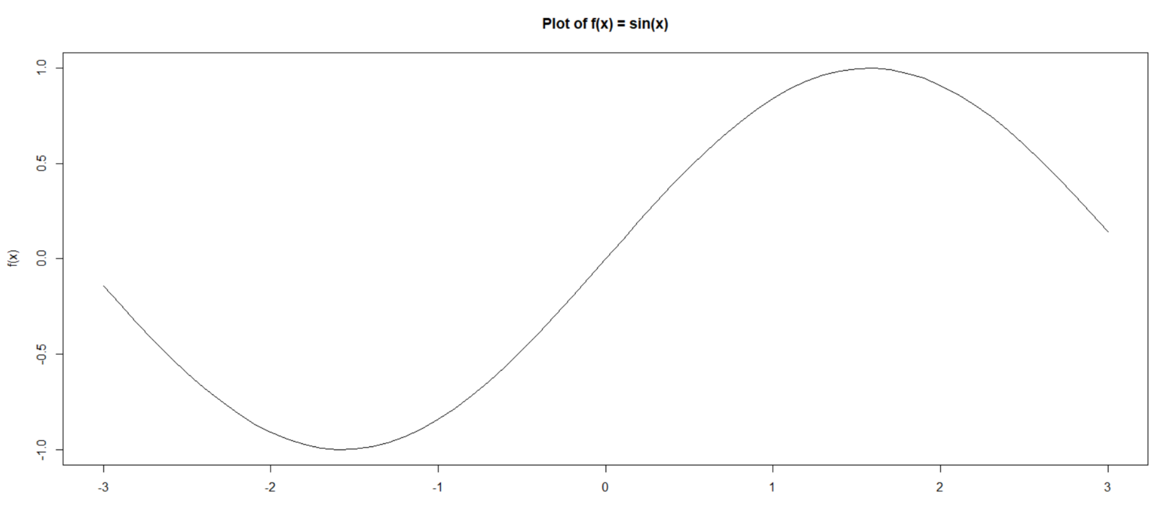
```
# Create data for the graph.
x <- c(21, 62, 10, 53)
labels <- c("London", "New York", "Singapore", "Mumbai")
rainbow <- c("violet", "blue", "green", "yellow", "orange", "red")
colors <- sample ( rainbow , 4)

# Plot the chart with title and rainbow color pallet.
pie (x, labels, main = "City Pie Chart", col=colors )
```



SET -2

1. Write R program to plot the function f(x)= sin(x) in the interval (-3,3) in steps of 0.1 .

```r
# Define the range and step size
x <- seq(-3, 3, by = 0.1)
# Calculate the corresponding function values
y <- sin(x)
# Plot the function
plot(x, y, type = "l", xlab = "x", ylab = "f(x)", main = "Plot of f(x) = sin(x)")
```



SET - 11

1. Write R program to compute mode for the following frequency distribution.

height in cm	145 -150	155 -160	155 -160	160 -165	165 -170	170 -175	175 -180	180 -185
no of adult men	4	6	28	58	64	30	5	5

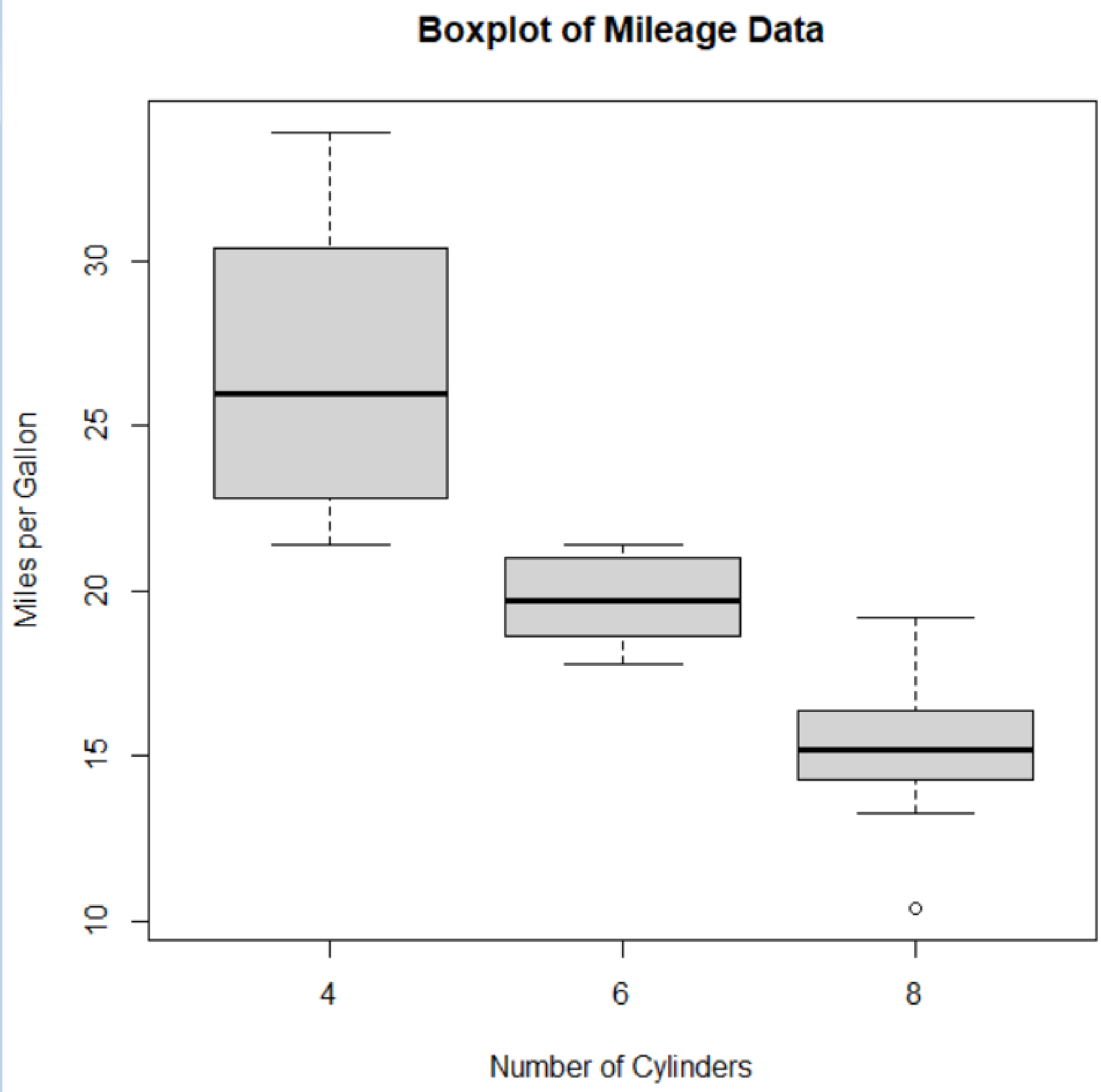
```
> serial_no<-seq(1:8)
> x<-seq(145,185,5)
> lower_limit<-seq(145,180,5)
> f<-c(4,6,28,58,64,30,5,5)
> n<-sum(f)
> class_interval<-c("145 - 150","150 - 155","155 - 160","160 - 165","165 - 170",
+                   "170 - 175","175 - 180","180 - 185")
> df<-data.frame(sno,class_interval,lower_limit,f)
> df
  sno class_interval lower_limit  f
1   1      145 - 150         145  4
2   2      150 - 155         150  6
3   3      155 - 160         155 28
4   4      160 - 165         160 58
5   5      165 - 170         165 64
6   6      170 - 175         170 30
7   7      175 - 180         175  5
8   8      180 - 185         180  5
> s1<- which(f==max(f))
> f1<-f[s1]
> f0<-f[s1-1]
> f2<-f[s1+1]
> L<-x[s1]
> h<-5
> mode<- L+((f1-f0)/(2*f1-f0-f2))*h
> mode
[1] 165.75
```

2. Write R program to compute the factorial of a given number using Recursion and User defined function.

```
# User-defined function to calculate factorial using recursion
factorial <- function(n) {
  if (n == 0) {
    return(1)
  } else {
    return(n * factorial(n - 1))
  }
}

# Prompt user for input
num <- as.integer(readline("Enter a positive integer: "))

# Check if the input is valid
if (num < 0) {
  cat("Error: Please enter a positive integer.")
} else {
  # Calculate and print the factorial
  result <- factorial(num)
  cat("Factorial of", num, "is", result)
}
```

3. Write R program to generate boxplot for the relation between “mpg”(miles per gallon) and “cyl”(number of cylinders) using mtcars dataset.

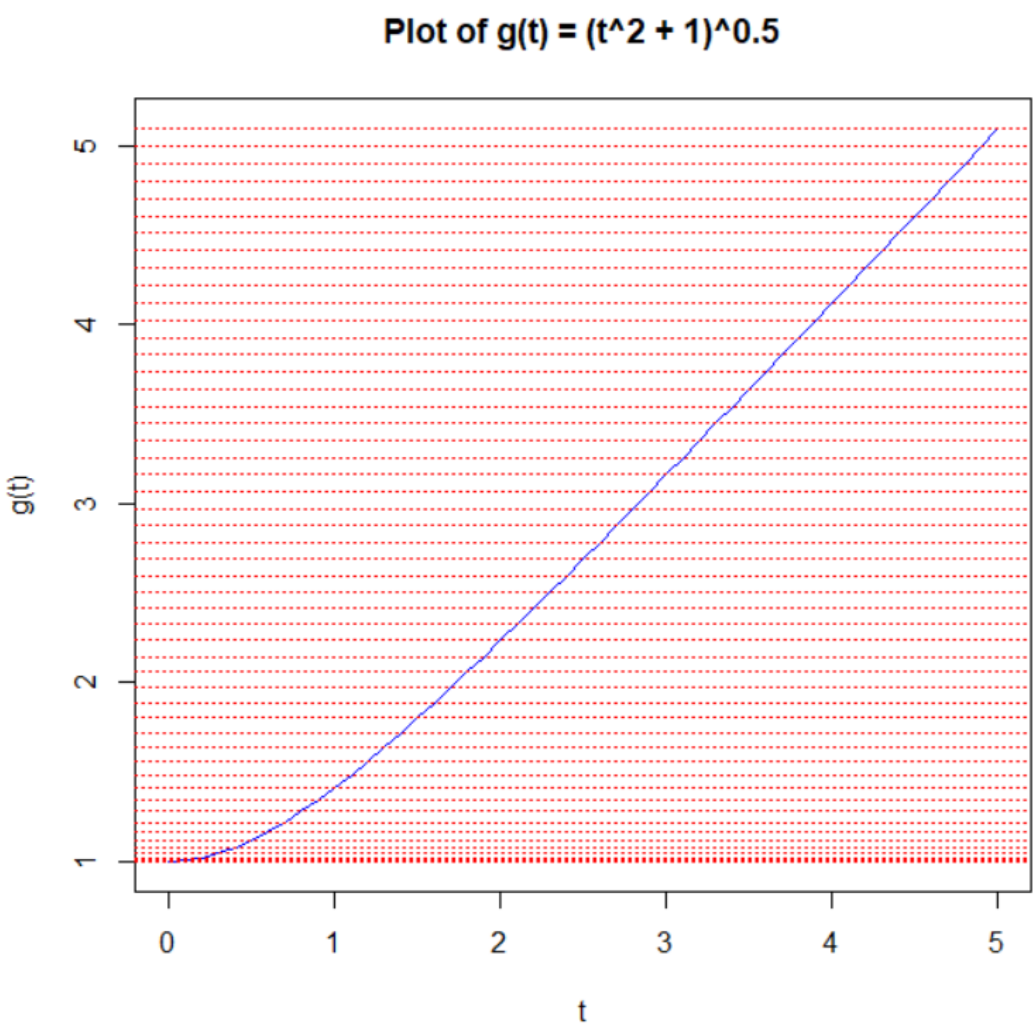
```
# Load the mtcars dataset
data(mtcars)

# Generate boxplot
boxplot(mpg ~ cyl, data = mtcars, xlab = "Number of Cylinders", ylab = "Miles per Gallon", main
= "Boxplot of Mileage Data")
```



4. plot the function g(t) = (t^2 + 1)^0.5 for t between 0 and 5. using dotted line and color of dotted lines is ”red”.
(This is ChatGPT generated, so not sure if it's correct or not as the output is different)

```
# Define the function
g <- function(t) {
  sqrt(t^2 + 1)
}
```

```r
# Generate values for t
t <- seq(0, 5, by = 0.1)

# Calculate values for g(t)
g_values <- g(t)

# Plot the function with solid line
plot(t, g_values, type = "l", col = "blue", xlab = "t", ylab = "g(t)", main = "Plot of g(t) = (t^2 +
1)^0.5")

# Add red dotted lines
abline(h = g_values, lty = "dotted", col = "red")
```



SET - 15

1. Write R program that reads a number and check whether the given number is even or odd

```
# Read the number from the user
number <- as.integer(readline("Enter a number: "))
# Check if the number is even or odd
if (number %% 2 == 0) {
  cat(number, "is an even number.\n")
} else {
  cat(number, "is an odd number.\n")
}
```

2. use apply() function to find sum of columns in a matrix.

```
# Create a matrix
matrix <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3, ncol = 3, byrow = TRUE)

# Use apply() to find the sum of columns
col_sums <- apply(matrix, MARGIN = 2, FUN = sum)

# Print the column sums
col_sums
```

```
> m1<-matrix(c(1:9),3,3,byrow=TRUE)
> #printing matrix
> m1
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
> #applying sum row wise
> x<-apply(m1,1,sum)
> x
[1]  6 15 24
> #applying sum colomn wise
> y<-apply(m1,2,sum)
> y
[1] 12 15 18
> 
```

3. consider the following data frame

Ozone Solar.R Wind Temp Month Day
41 190 7.4 67 5 1
36 118 8 72 5 2
12 149 12.6 74 5 3
18 313 11.5 62 5 4
NA NA 14.3 56 5 5
28 NA 14.9 66 5 6

(a) Write a R command to create above data frame “air”
(b) Create a vector v1 by sorting the column Solar.R from the data frame in descending order with NA values at the front.
(c) Write R command to check missing values in the Ozone column and Number of missing values in each column.
(d) Write R command to find a mean of the Solar.R column after removing missing values.
(e) Write R command to replace missing values of Solar.R column by mean of the available values.

(a) To create the data frame "air" with the given values, you can use the following R command:

```R
# Create the data frame "air"
air <- data.frame(
  Ozone = c(41, 36, 12, 18, NA, 28),
  Solar.R = c(190, 118, 149, 313, NA, NA),
  Wind = c(7.4, 8, 12.6, 11.5, 14.3, 14.9),
  Temp = c(67, 72, 74, 62, 56, 66),
  Month = c(5, 5, 5, 5, 5, 5),
  Day = c(1, 2, 3, 4, 5, 6)
)

# Print the "air" data frame
air
```

```
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
```

(b) To create a vector "v1" by sorting the "Solar.R" column from the data frame "air" in descending order with NA values at the front, you can use the following R command:

```R
# Create the vector "v1" by sorting the "Solar.R" column in descending order with NA values at the front
v1 <- sort(air$Solar.R, decreasing = TRUE, na.last = FALSE)

# Print the vector "v1"
v1
```

```
[1]  NA  NA 313 190 149 118
```

(c) To check missing values in the "Ozone" column and the number of missing values in each column, you can use the following R command:

```R
# Check missing values in the "Ozone" column
missing_values <- is.na(air$Ozone)

# Print the number of missing values in each column
colSums(is.na(air))
```

```
 Ozone Solar.R    Wind    Temp   Month     Day
     1       2       0       0       0       0
```

(d) To find the mean of the "Solar.R" column after removing missing values, you can use the following R command:

```R
# Calculate the mean of the "Solar.R" column after removing missing values
mean_solar_r <- mean(air$Solar.R, na.rm = TRUE)

# Print the mean of the "Solar.R" column
mean_solar_r
```

```
[1] 192.5
```

(e) To replace missing values of the "Solar.R" column by the mean of the available values, you can use the following R command:

```R
# Replace missing values of the "Solar.R" column by the mean of the available values
air$Solar.R[is.na(air$Solar.R)] <- mean(air$Solar.R, na.rm = TRUE)

# Print the updated "air" data frame
air
```

```
  Ozone Solar.R Wind Temp Month Day
1    41   190.0  7.4   67     5   1
2    36   118.0  8.0   72     5   2
3    12   149.0 12.6   74     5   3
4    18   313.0 11.5   62     5   4
5    NA   192.5 14.3   56     5   5
6    28   192.5 14.9   66     5   6
```

In this code, the missing values in the "Solar.R" column are replaced by the mean of the available values using the assignment operator (`<-`) and subsetting with `is.na(air$Solar.R)`. The `mean()` function is used to calculate the mean of the "Solar.R" column with `na.rm = TRUE` to exclude missing values from the calculation. The updated "air" data frame is then printed.

4. write R code to fit a straight line y = a + bx to the following data by the method of least squares

x	0	1	3	6	8
y	1	3	2	5	4

```
x <- c(0, 1, 3, 6, 8)
y <- c(1, 3, 2, 5, 4)

# Create a data frame with the x and y values
data <- data.frame(x, y)

# Fit the linear regression model
model <- lm(y ~ x, data = data)

# Get the estimated coefficients
a <- coef(model)[1]
b <- coef(model)[2]
```

```
# Print the coefficients
cat("Estimated coefficients:\n")
Estimated coefficients:
cat("a =", a, "\n")
a = 1.646018
cat("b =", b, "\n")
b = 0.3761062
```

OR

```
> x <- c(0, 1, 3, 6, 8)
> y <- c(1, 3, 2, 5, 4)
> sqx <- x^2
> xy <- x*y
> df <- data.frame(x,y,xy,sqx)
> df
  x y xy sqx
1 0 1  0   0
2 1 3  3   1
3 3 2  6   9
4 6 5 30  36
5 8 4 32  64
> reg <- lm(y~x)
> print(reg)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
     1.6460       0.3761
```