THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Submitted By:
Name – Shreeya Chatterji
Roll number -102103447
Batch - 3COE16

Submitted To:
Dr. Rajanish Rai

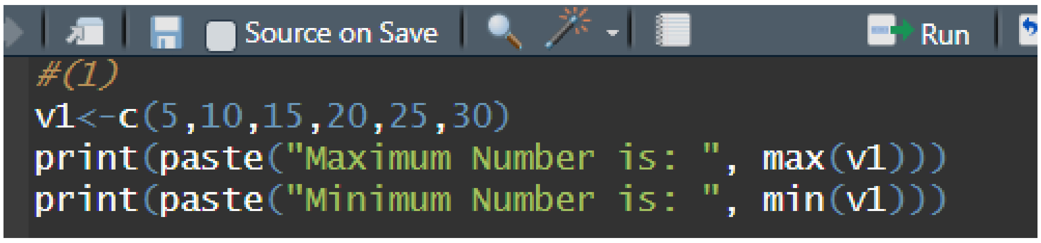
July 2022 – December 2022

Probability and Statistics (UCS410)

Experiment 1: Basics of R programming

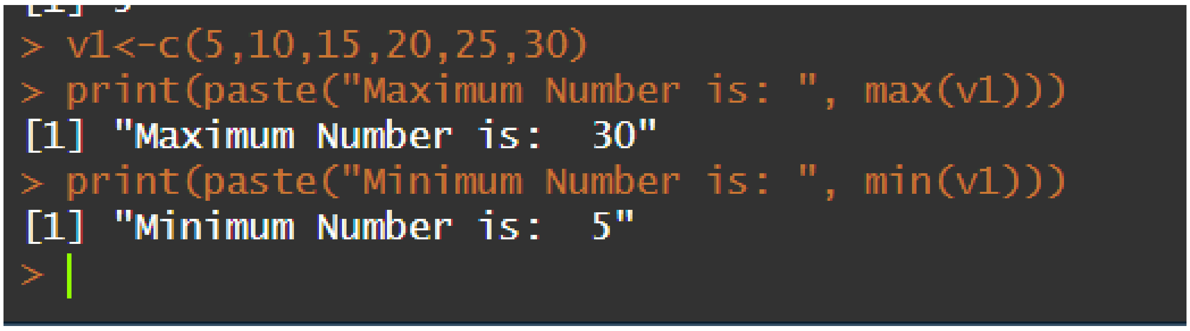
(1) Create a vector c = [5,10,15,20,25,30] and write a program which returns the maximum and minimum of this vector.

CODE:



```
#(1)
v1<-c(5,10,15,20,25,30)
print(paste("Maximum Number is: ", max(v1)))
print(paste("Minimum Number is: ", min(v1)))
```

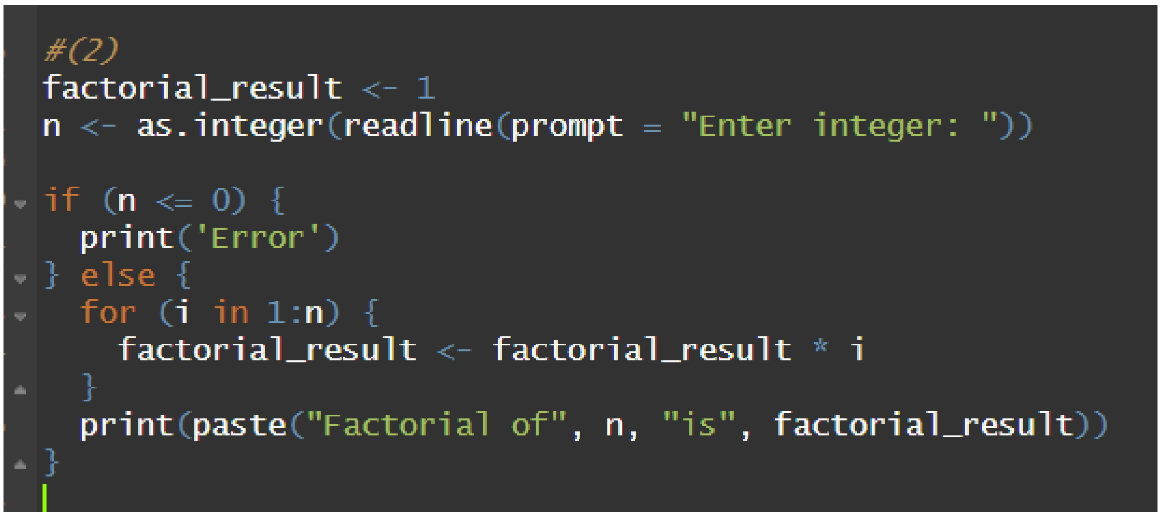
OUTPUT:



```
> v1<-c(5,10,15,20,25,30)
> print(paste("Maximum Number is: ", max(v1)))
[1] "Maximum Number is:  30"
> print(paste("Minimum Number is: ", min(v1)))
[1] "Minimum Number is:  5"
> 
```

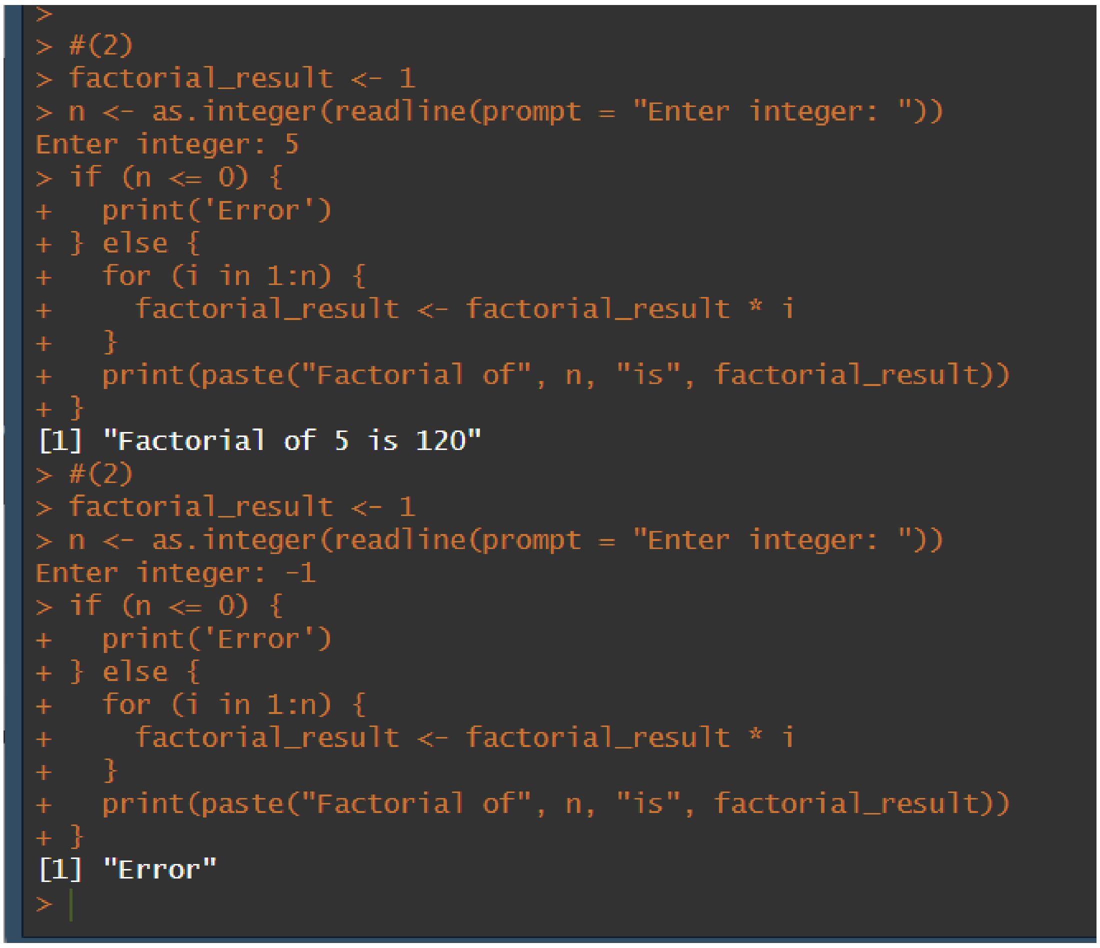
(2) Write a program in R to find factorial of a number by taking input from user. Please print error message if the input number is negative.

CODE:



```
#(2)
factorial_result <- 1
n <- as.integer(readline(prompt = "Enter integer: "))

if (n <= 0) {
  print('Error')
} else {
  for (i in 1:n) {
    factorial_result <- factorial_result * i
  }
  print(paste("Factorial of", n, "is", factorial_result))
}
```

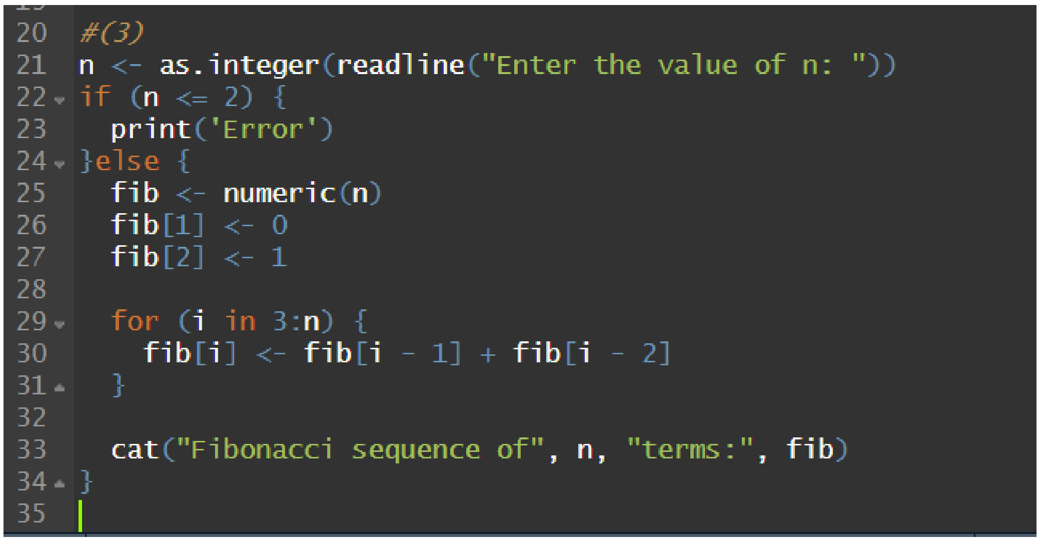
OUTPUT:



```
>
> #(2)
> factorial_result <- 1
> n <- as.integer(readline(prompt = "Enter integer: "))
Enter integer: 5
> if (n <= 0) {
+   print('Error')
+ } else {
+   for (i in 1:n) {
+     factorial_result <- factorial_result * i
+   }
+   print(paste("Factorial of", n, "is", factorial_result))
+ }
[1] "Factorial of 5 is 120"
> #(2)
> factorial_result <- 1
> n <- as.integer(readline(prompt = "Enter integer: "))
Enter integer: -1
> if (n <= 0) {
+   print('Error')
+ } else {
+   for (i in 1:n) {
+     factorial_result <- factorial_result * i
+   }
+   print(paste("Factorial of", n, "is", factorial_result))
+ }
[1] "Error"
>
```

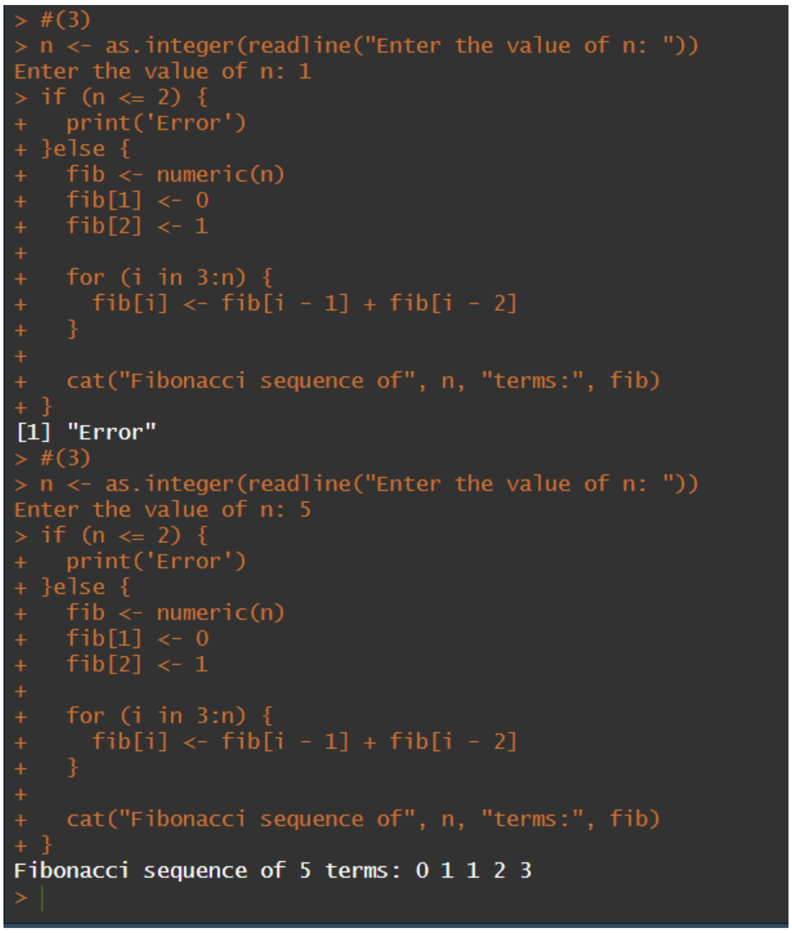
(3) Write a program to write first n terms of a Fibonacci sequence. You may take n as an input from the user.

CODE:



```
#(3)
n <- as.integer(readline("Enter the value of n: "))
if (n <= 2) {
  print('Error')
}else {
  fib <- numeric(n)
  fib[1] <- 0
  fib[2] <- 1

  for (i in 3:n) {
    fib[i] <- fib[i - 1] + fib[i - 2]
  }

  cat("Fibonacci sequence of", n, "terms:", fib)
}

```

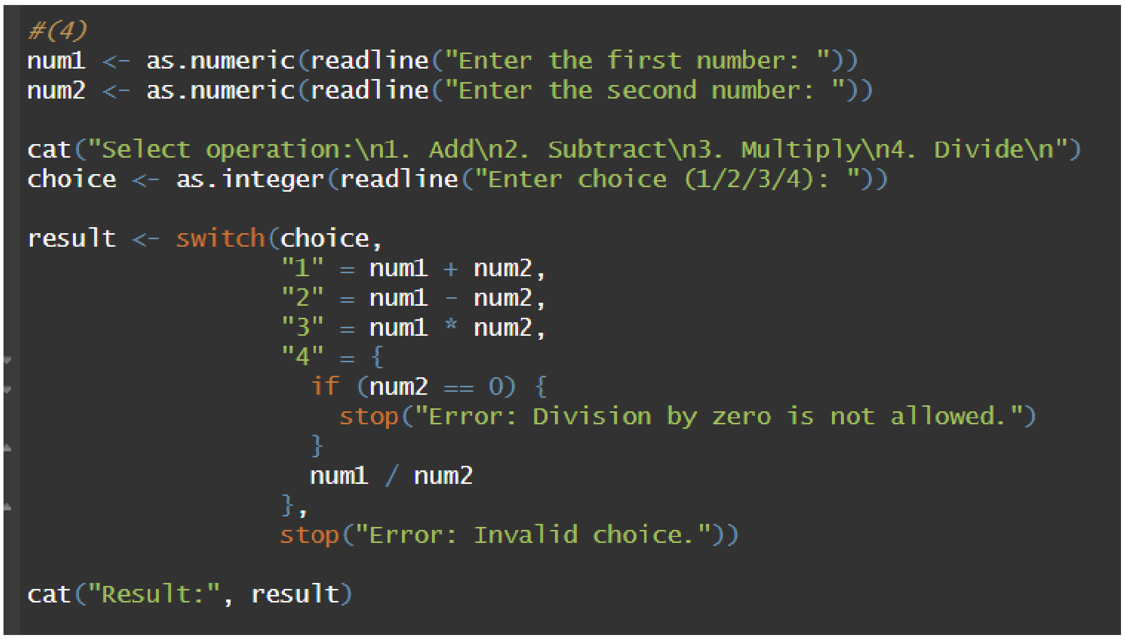
OUTPUT:



```
> #(3)
> n <- as.integer(readline("Enter the value of n: "))
Enter the value of n: 1
> if (n <= 2) {
+   print('Error')
+ }else {
+   fib <- numeric(n)
+   fib[1] <- 0
+   fib[2] <- 1
+ 
+   for (i in 3:n) {
+     fib[i] <- fib[i - 1] + fib[i - 2]
+   }
+ 
+   cat("Fibonacci sequence of", n, "terms:", fib)
+ }
[1] "Error"
> #(3)
> n <- as.integer(readline("Enter the value of n: "))
Enter the value of n: 5
> if (n <= 2) {
+   print('Error')
+ }else {
+   fib <- numeric(n)
+   fib[1] <- 0
+   fib[2] <- 1
+ 
+   for (i in 3:n) {
+     fib[i] <- fib[i - 1] + fib[i - 2]
+   }
+ 
+   cat("Fibonacci sequence of", n, "terms:", fib)
+ }
Fibonacci sequence of 5 terms: 0 1 1 2 3
> 
```

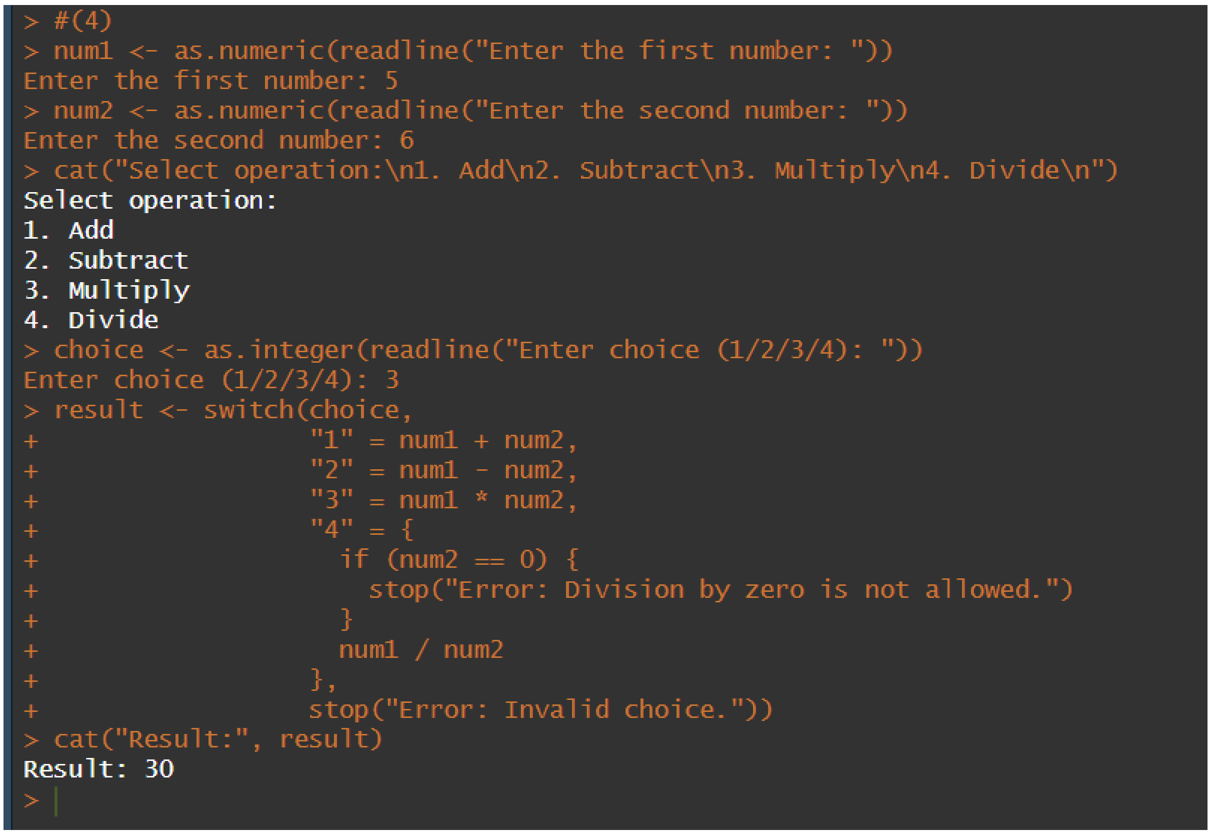
(4) Write an R program to make a simple calculator which can add, subtract, multiply and divide.

CODE:



```r
#(4)
num1 <- as.numeric(readline("Enter the first number: "))
num2 <- as.numeric(readline("Enter the second number: "))

cat("Select operation:\n1. Add\n2. Subtract\n3. Multiply\n4. Divide\n")
choice <- as.integer(readline("Enter choice (1/2/3/4): "))

result <- switch(choice,
                 "1" = num1 + num2,
                 "2" = num1 - num2,
                 "3" = num1 * num2,
                 "4" = {
                   if (num2 == 0) {
                     stop("Error: Division by zero is not allowed.")
                   }
                   num1 / num2
                 },
                 stop("Error: Invalid choice."))

cat("Result:", result)
```

OUTPUT:

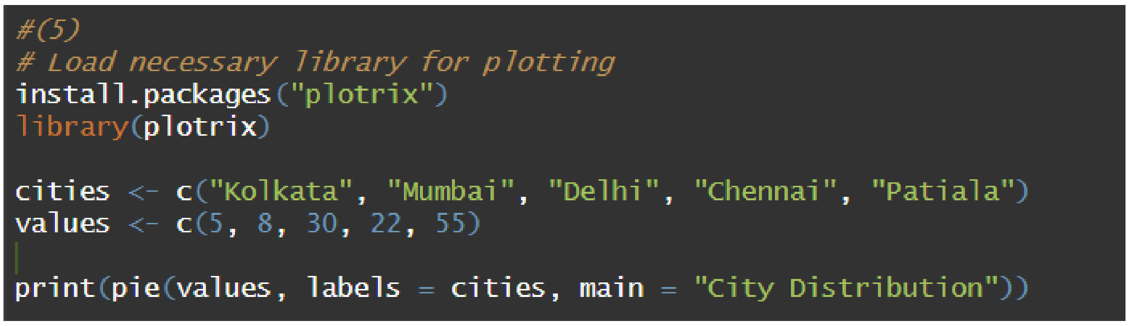


```
> #(4)
> num1 <- as.numeric(readline("Enter the first number: "))
Enter the first number: 5
> num2 <- as.numeric(readline("Enter the second number: "))
Enter the second number: 6
> cat("Select operation:\n1. Add\n2. Subtract\n3. Multiply\n4. Divide\n")
Select operation:
1. Add
2. Subtract
3. Multiply
4. Divide
> choice <- as.integer(readline("Enter choice (1/2/3/4): "))
Enter choice (1/2/3/4): 3
> result <- switch(choice,
+                  "1" = num1 + num2,
+                  "2" = num1 - num2,
+                  "3" = num1 * num2,
+                  "4" = {
+                    if (num2 == 0) {
+                      stop("Error: Division by zero is not allowed.")
+                    }
+                    num1 / num2
+                  },
+                  stop("Error: Invalid choice."))
> cat("Result:", result)
Result: 30
> 
```

(5) Explore plot, pie, barplot etc. (the plotting options) which are built-in functions in R.

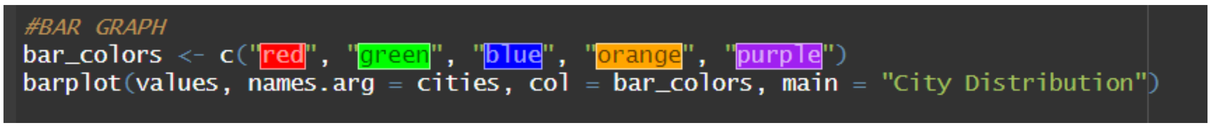
CODE:

PIE CHART:



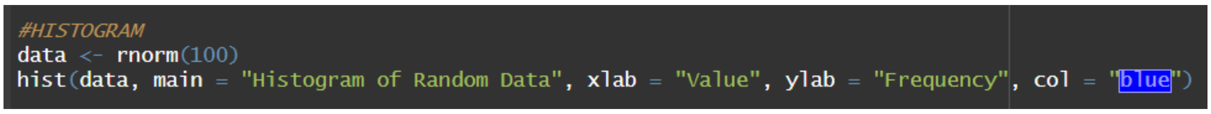
```r
#(5)
# Load necessary library for plotting
install.packages("plotrix")
library(plotrix)

cities <- c("Kolkata", "Mumbai", "Delhi", "Chennai", "Patiala")
values <- c(5, 8, 30, 22, 55)

print(pie(values, labels = cities, main = "City Distribution"))
```

BAR GRAPH:



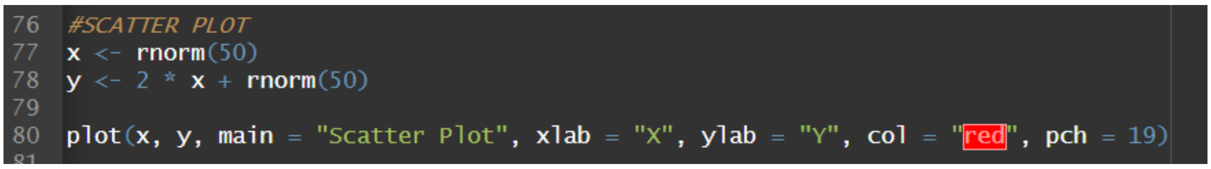
```r
#BAR GRAPH
bar_colors <- c("red", "green", "blue", "orange", "purple")
barplot(values, names.arg = cities, col = bar_colors, main = "City Distribution")
```

HISTOGRAM:



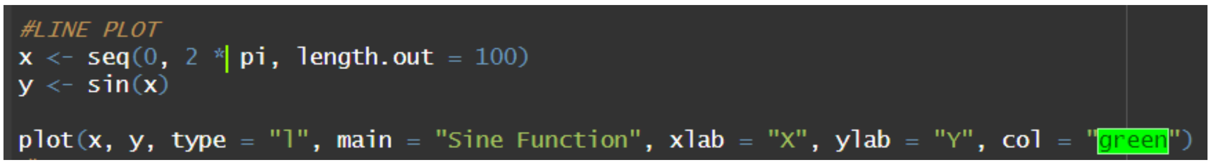
```r
#HISTOGRAM
data <- rnorm(100)
hist(data, main = "Histogram of Random Data", xlab = "Value", ylab = "Frequency", col = "blue")
```

SCATTER PLOT:



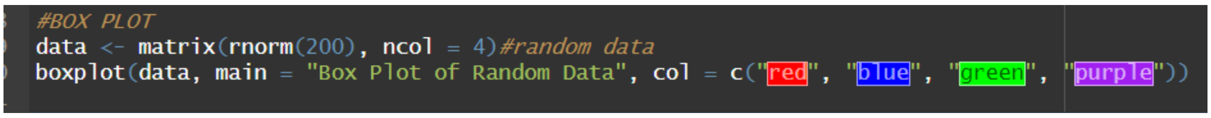
```
#SCATTER PLOT
x <- rnorm(50)
y <- 2 * x + rnorm(50)

plot(x, y, main = "Scatter Plot", xlab = "X", ylab = "Y", col = "red", pch = 19)
```

LINE PLOT:



```
#LINE PLOT
x <- seq(0, 2 * pi, length.out = 100)
y <- sin(x)

plot(x, y, type = "l", main = "Sine Function", xlab = "X", ylab = "Y", col = "green")
```

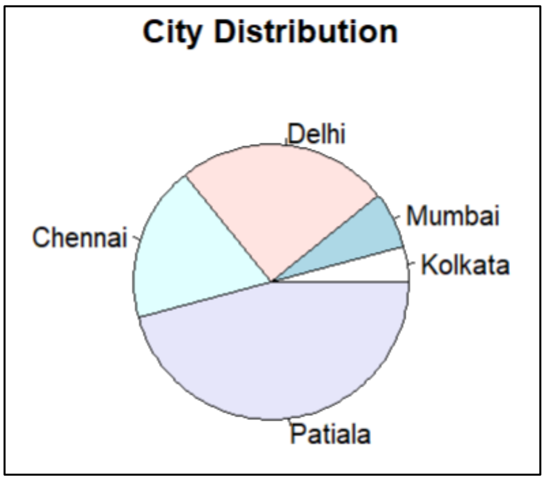
BOX PLOT:



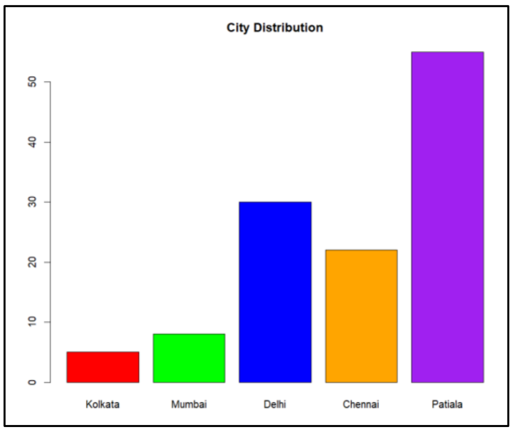
```
#BOX PLOT
data <- matrix(rnorm(200), ncol = 4)#random data
boxplot(data, main = "Box Plot of Random Data", col = c("red", "blue", "green", "purple"))
```

OUTPUT:

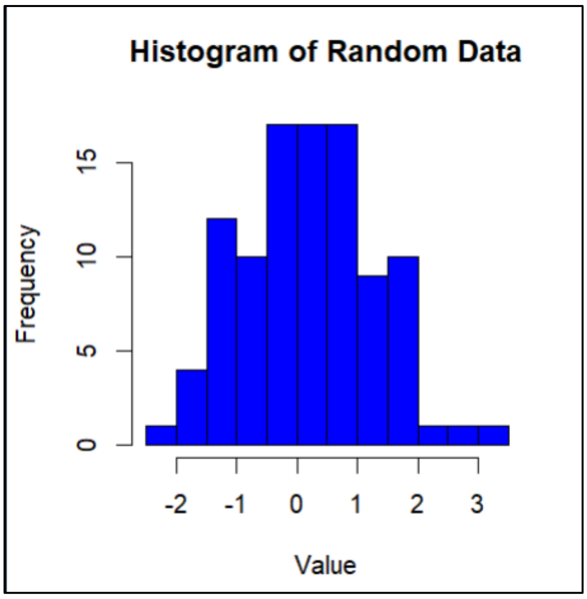
PIE CHART:



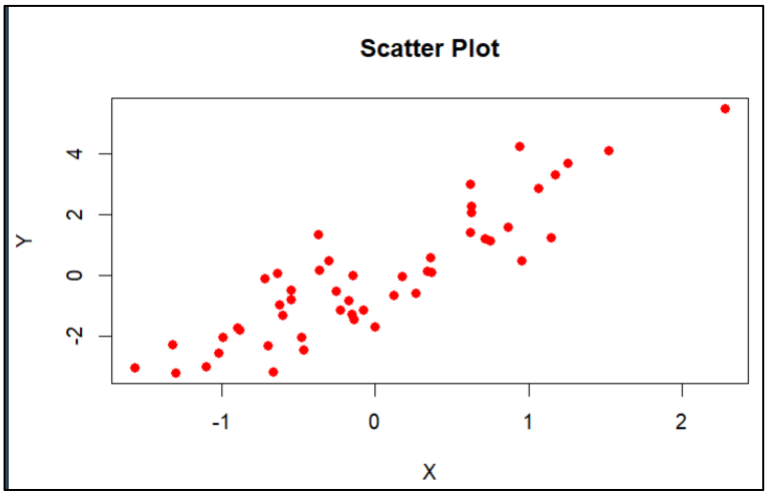
BAR GRAPH:



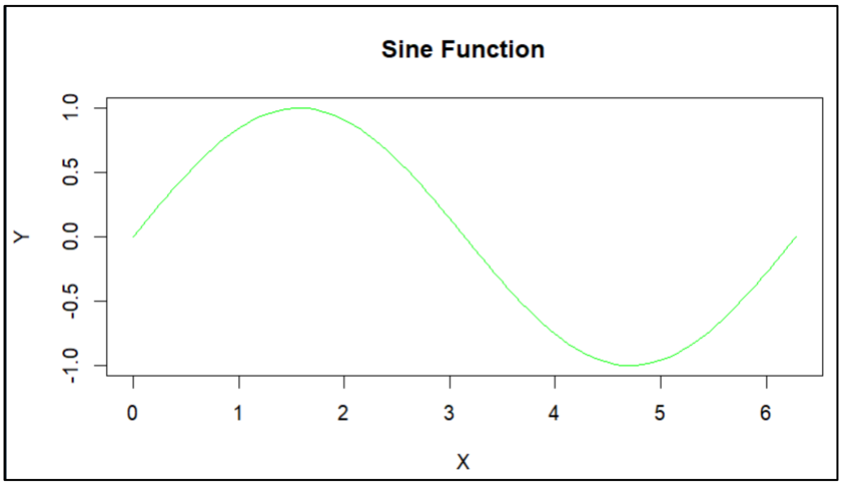
HISTOGRAM:



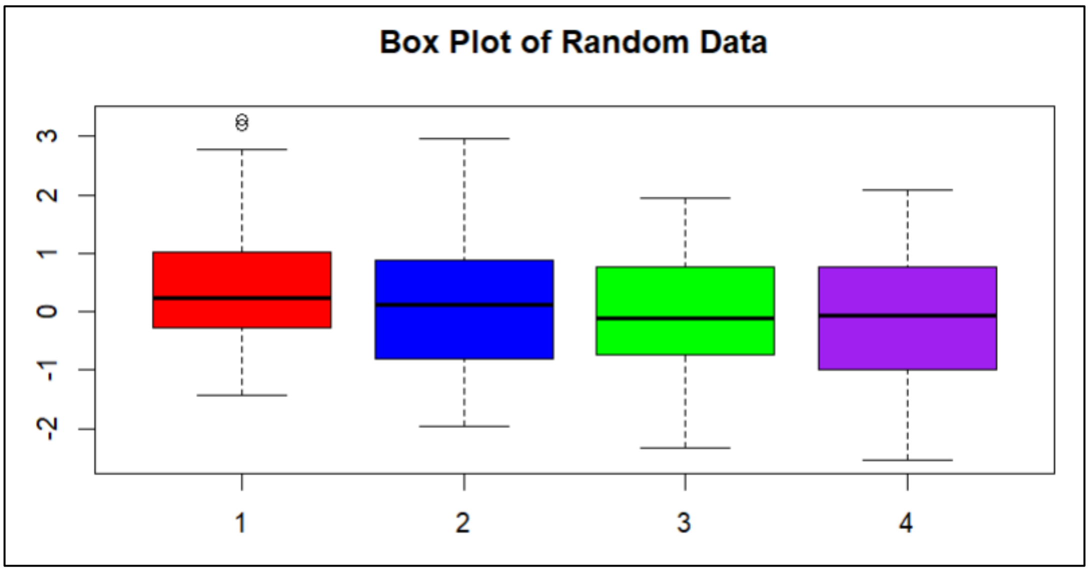
SCATTER PLOT:



LINE PLOT:



BOX PLOT:

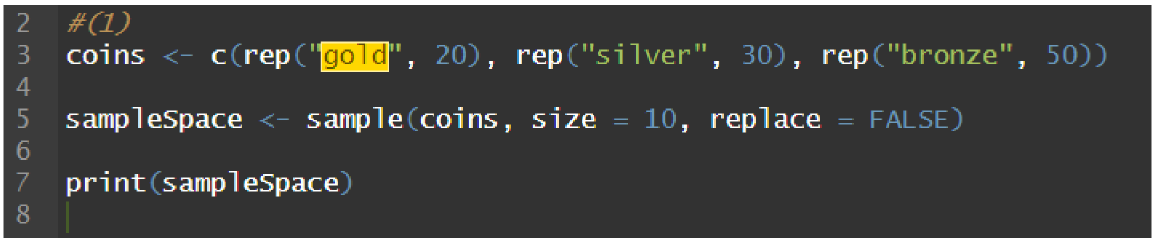


Probability and Statistics (UCS410)

Experiment 2: Descriptive statistics, Sample space, definition of Probability

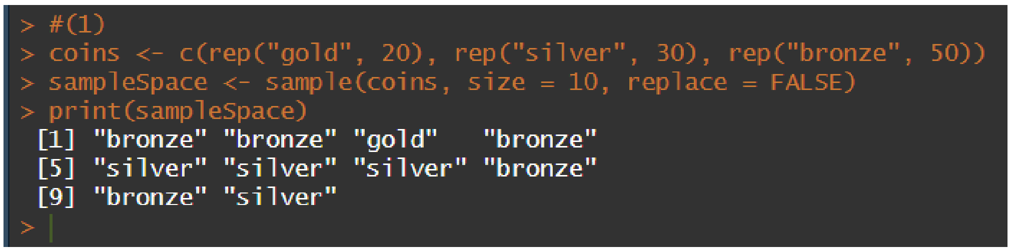
(1) (a) Suppose there is a chest of coins with 20 gold, 30 silver and 50 bronze coins. You randomly draw 10 coins from this chest. Write an R code which will give us the sample space for this experiment. (use of sample(): an in-built function in R)

CODE:



```r
#(1)
coins <- c(rep("gold", 20), rep("silver", 30), rep("bronze", 50))

sampleSpace <- sample(coins, size = 10, replace = FALSE)

print(sampleSpace)

```

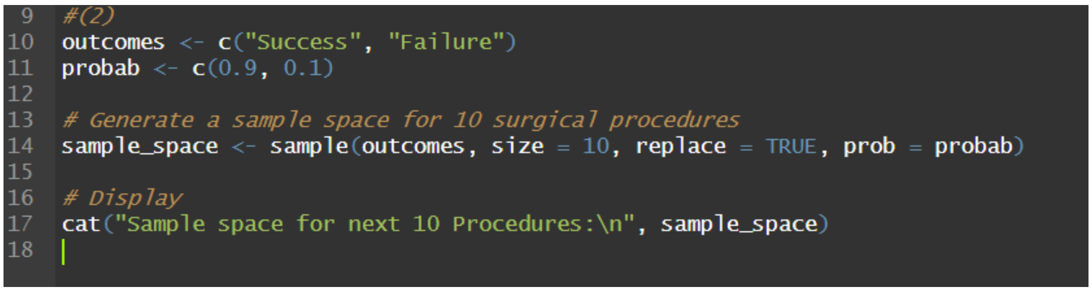
OUTPUT:



```
> #(1)
> coins <- c(rep("gold", 20), rep("silver", 30), rep("bronze", 50))
> sampleSpace <- sample(coins, size = 10, replace = FALSE)
> print(sampleSpace)
 [1] "bronze" "bronze" "gold"   "bronze"
 [5] "silver" "silver" "silver" "bronze"
 [9] "bronze" "silver"
> 
```

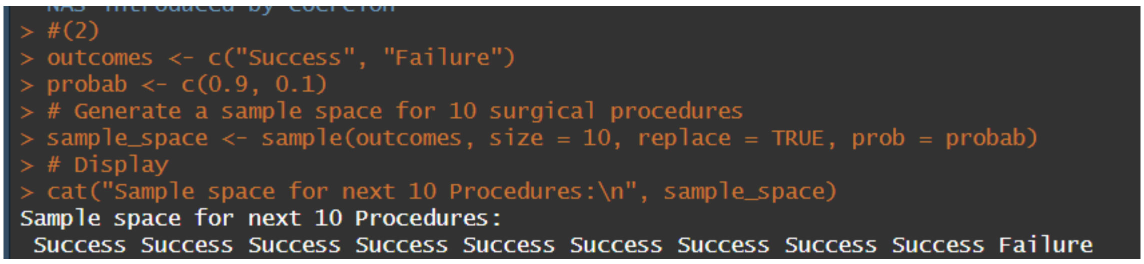
(b) In a surgical procedure, the chances of success and failure are 90% and 10% respectively. Generate a sample space for the next 10 surgical procedures performed. (use of prob(): an in-built function in R)

CODE:



```r
#(2)
outcomes <- c("Success", "Failure")
probab <- c(0.9, 0.1)

# Generate a sample space for 10 surgical procedures
sample_space <- sample(outcomes, size = 10, replace = TRUE, prob = probab)

# Display
cat("Sample space for next 10 Procedures:\n", sample_space)

```

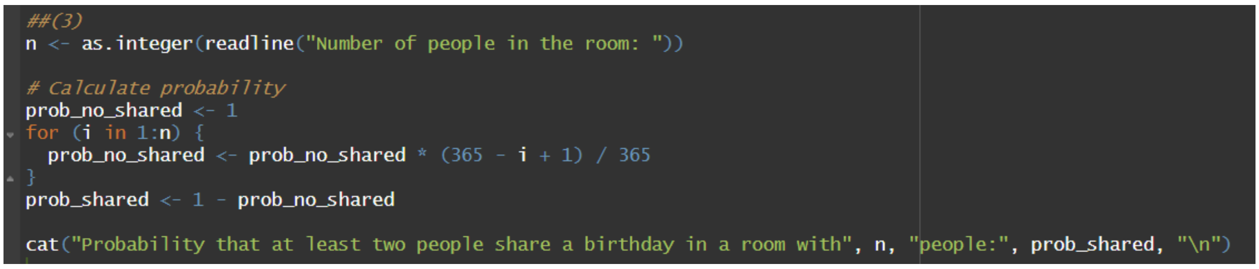
OUTPUT:



```
> #(2)
> outcomes <- c("Success", "Failure")
> probab <- c(0.9, 0.1)
> # Generate a sample space for 10 surgical procedures
> sample_space <- sample(outcomes, size = 10, replace = TRUE, prob = probab)
> # Display
> cat("Sample space for next 10 Procedures:\n", sample_space)
Sample space for next 10 Procedures:
 Success Success Success Success Success Success Success Success Success Failure
```

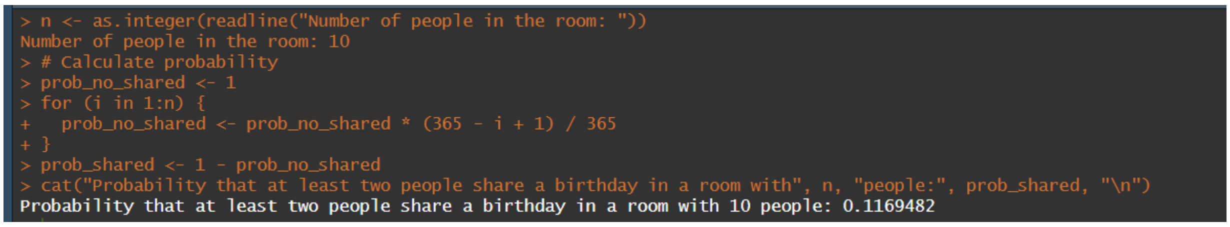
(2) A room has n people, and each has an equal chance of being born on any of the 365 days of the year. (For simplicity, we'll ignore leap years). What is the probability that two people in the room have the same birthday?

CODE:



```r
##(3)
n <- as.integer(readline("Number of people in the room: "))

# Calculate probability
prob_no_shared <- 1
for (i in 1:n) {
  prob_no_shared <- prob_no_shared * (365 - i + 1) / 365
}
prob_shared <- 1 - prob_no_shared

cat("Probability that at least two people share a birthday in a room with", n, "people:", prob_shared, "\n")
```

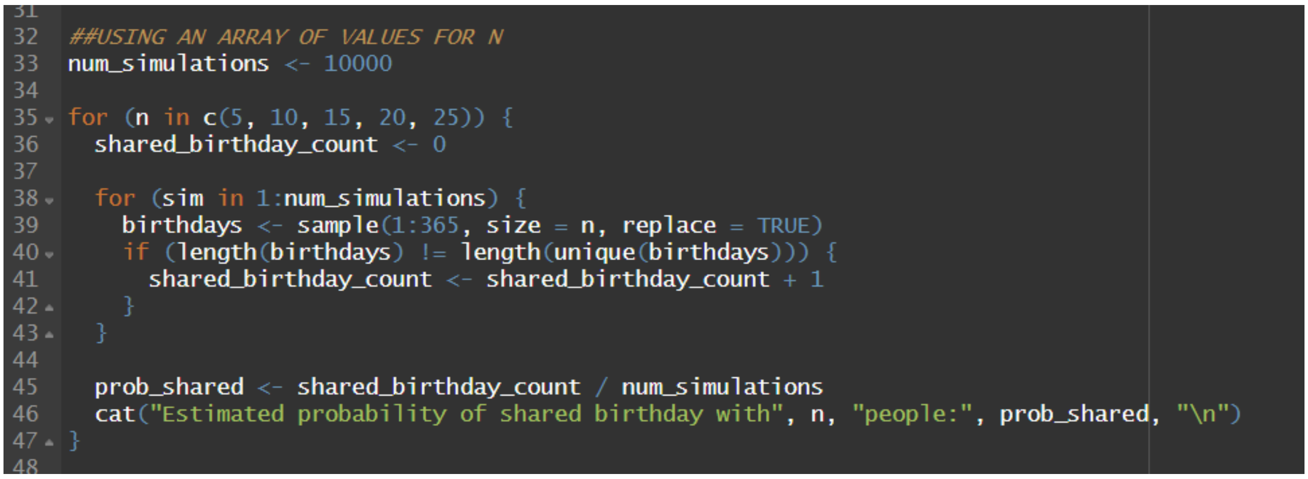
OUTPUT:



```
> n <- as.integer(readline("Number of people in the room: "))
Number of people in the room: 10
> # Calculate probability
> prob_no_shared <- 1
> for (i in 1:n) {
+   prob_no_shared <- prob_no_shared * (365 - i + 1) / 365
+ }
> prob_shared <- 1 - prob_no_shared
> cat("Probability that at least two people share a birthday in a room with", n, "people:", prob_shared, "\n")
Probability that at least two people share a birthday in a room with 10 people: 0.1169482
```

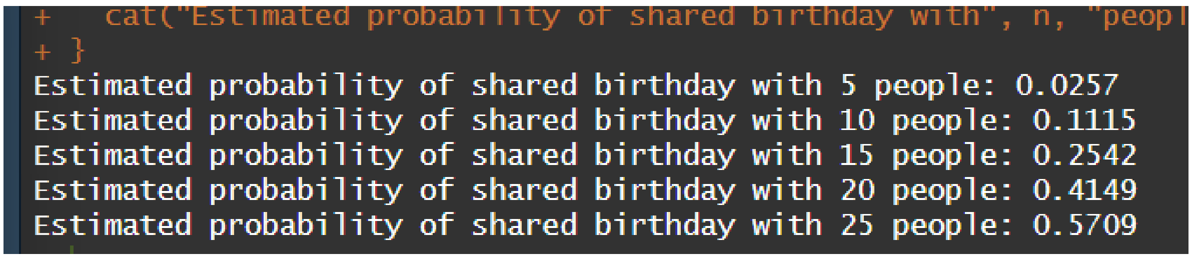
(a) Use an R simulation to estimate this for various n.

CODE:



```r

##USING AN ARRAY OF VALUES FOR N
num_simulations <- 10000

for (n in c(5, 10, 15, 20, 25)) {
  shared_birthday_count <- 0

  for (sim in 1:num_simulations) {
    birthdays <- sample(1:365, size = n, replace = TRUE)
    if (length(birthdays) != length(unique(birthdays))) {
      shared_birthday_count <- shared_birthday_count + 1
    }
  }

  prob_shared <- shared_birthday_count / num_simulations
  cat("Estimated probability of shared birthday with", n, "people:", prob_shared, "\n")
}

```

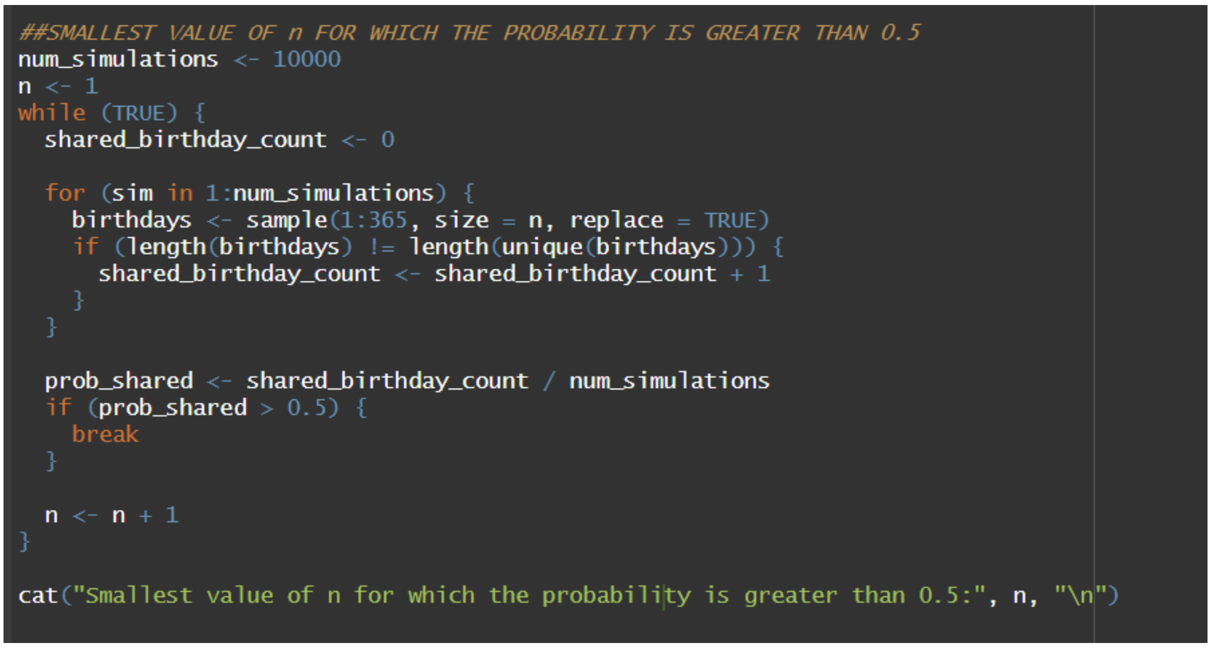
OUTPUT:



```
+   cat("Estimated probability of shared birthday with", n, "peopl
+ }
Estimated probability of shared birthday with 5 people: 0.0257
Estimated probability of shared birthday with 10 people: 0.1115
Estimated probability of shared birthday with 15 people: 0.2542
Estimated probability of shared birthday with 20 people: 0.4149
Estimated probability of shared birthday with 25 people: 0.5709
```

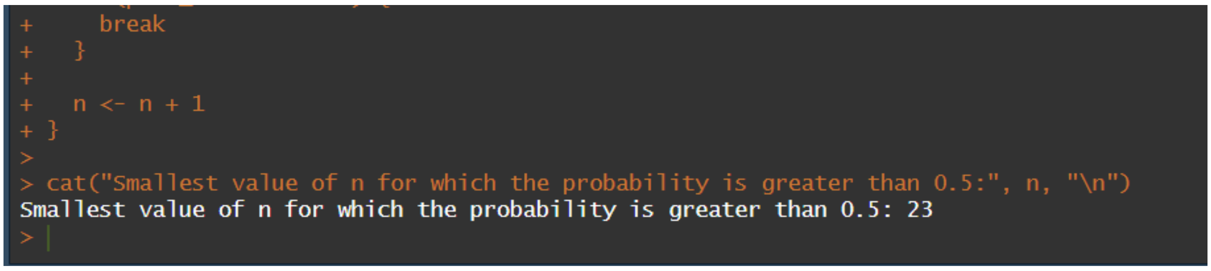
(b) Find the smallest value of n for which the probability of a match is greater than .5.

CODE:



```r
##SMALLEST VALUE OF n FOR WHICH THE PROBABILITY IS GREATER THAN 0.5
num_simulations <- 10000
n <- 1
while (TRUE) {
  shared_birthday_count <- 0

  for (sim in 1:num_simulations) {
    birthdays <- sample(1:365, size = n, replace = TRUE)
    if (length(birthdays) != length(unique(birthdays))) {
      shared_birthday_count <- shared_birthday_count + 1
    }
  }

  prob_shared <- shared_birthday_count / num_simulations
  if (prob_shared > 0.5) {
    break
  }

  n <- n + 1
}

cat("Smallest value of n for which the probability is greater than 0.5:", n, "\n")
```

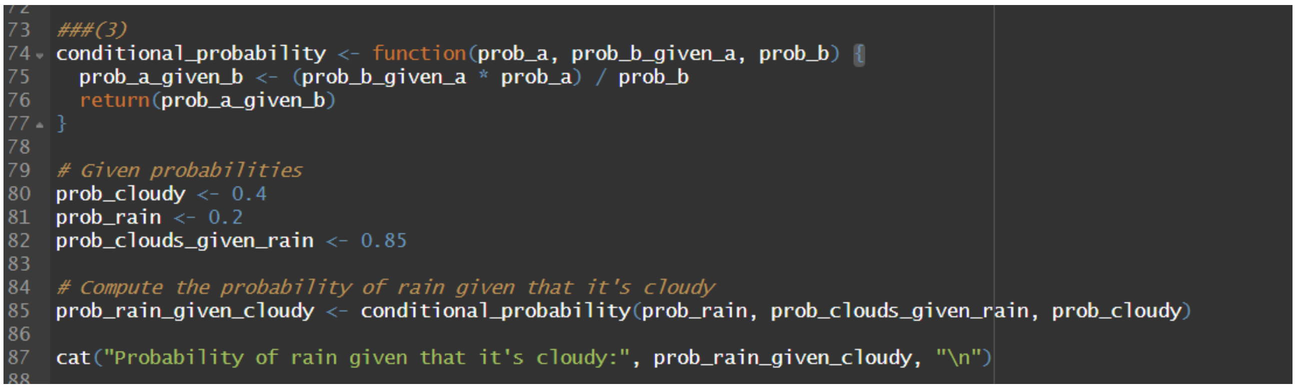
OUTPUT:



```
+     break
+   }
+
+   n <- n + 1
+ }
>
> cat("Smallest value of n for which the probability is greater than 0.5:", n, "\n")
Smallest value of n for which the probability is greater than 0.5: 23
>
```

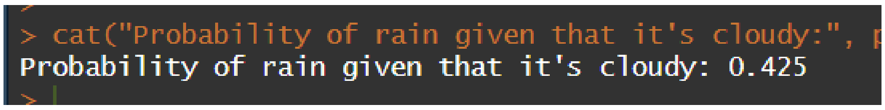
(3) Write an R function for computing conditional probability. Call this function to do the following problem: Suppose the probability of the weather being cloudy is 40%. Also suppose the probability of rain on a given day is 20% and that the probability of clouds on a rainy day is 85%. If it's cloudy outside on a given day, what is the probability that it will rain that day?

CODE:



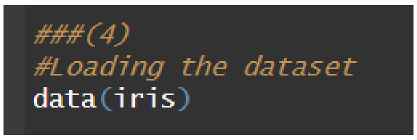
```r
###(3)
conditional_probability <- function(prob_a, prob_b_given_a, prob_b) {
  prob_a_given_b <- (prob_b_given_a * prob_a) / prob_b
  return(prob_a_given_b)
}

# Given probabilities
prob_cloudy <- 0.4
prob_rain <- 0.2
prob_clouds_given_rain <- 0.85

# Compute the probability of rain given that it's cloudy
prob_rain_given_cloudy <- conditional_probability(prob_rain, prob_clouds_given_rain, prob_cloudy)

cat("Probability of rain given that it's cloudy:", prob_rain_given_cloudy, "\n")
```

OUTPUT:



```
> cat("Probability of rain given that it's cloudy:", p
Probability of rain given that it's cloudy: 0.425
>
```

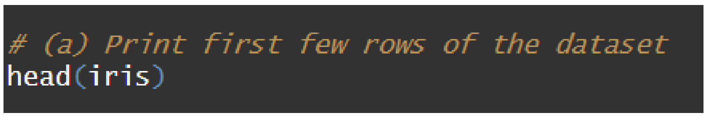
(4) The iris dataset is a built-in dataset in R that contains measurements on 4 different attributes (in centimeters) for 150 flowers from 3 different species. Load this dataset and do the following:



```
###(4)
#Loading the dataset
data(iris)
```

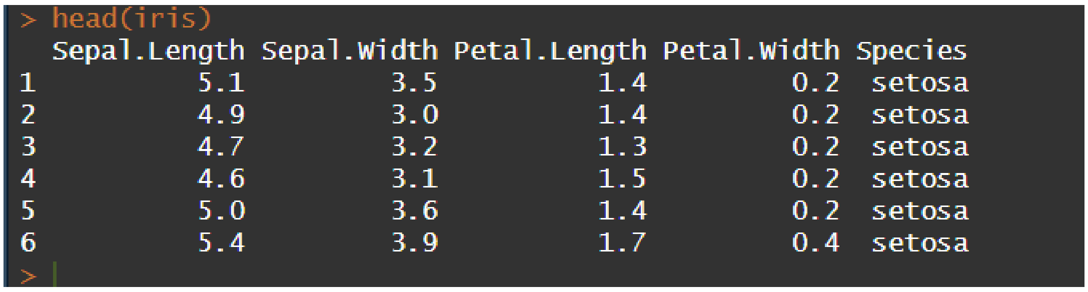
(a) Print first few rows of this dataset.

CODE:



```
# (a) Print first few rows of the dataset
head(iris)
```

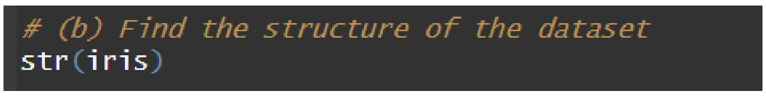
OUTPUT:



```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
>
```

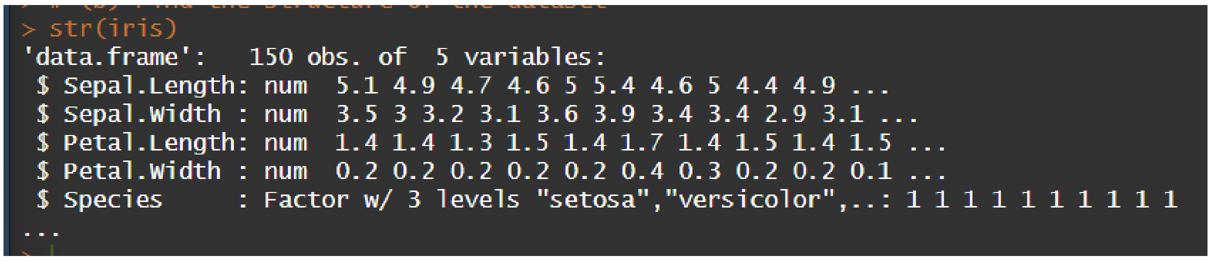
(b) Find the structure of this dataset.

CODE:



```
# (b) Find the structure of the dataset
str(iris)
```

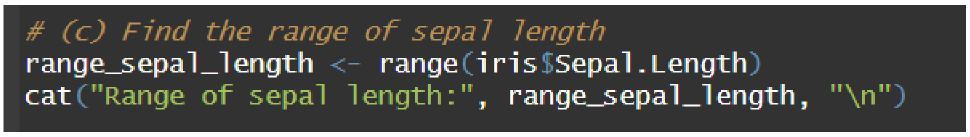
OUTPUT:



```
> str(iris)
'data.frame':	150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1
...
```

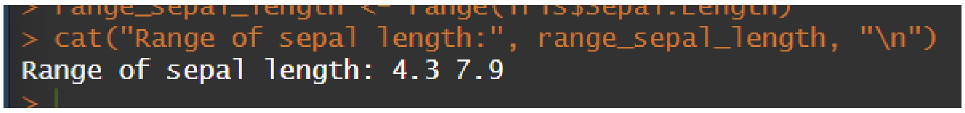
(c) Find the range of the data regarding the sepal length of flowers.

CODE:



```
# (c) Find the range of sepal length
range_sepal_length <- range(iris$Sepal.Length)
cat("Range of sepal length:", range_sepal_length, "\n")
```

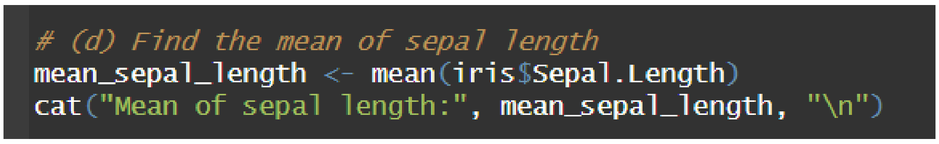
OUTPUT:



```
> range_sepal_length <- range(iris$Sepal.Length)
> cat("Range of sepal length:", range_sepal_length, "\n")
Range of sepal length: 4.3 7.9
> 
```

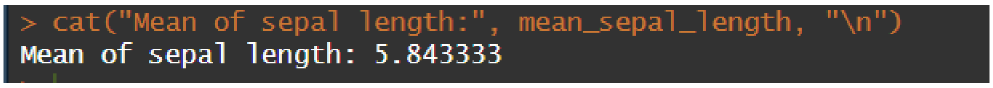
(d) Find the mean of the sepal length.

CODE:



```
# (d) Find the mean of sepal length
mean_sepal_length <- mean(iris$Sepal.Length)
cat("Mean of sepal length:", mean_sepal_length, "\n")
```

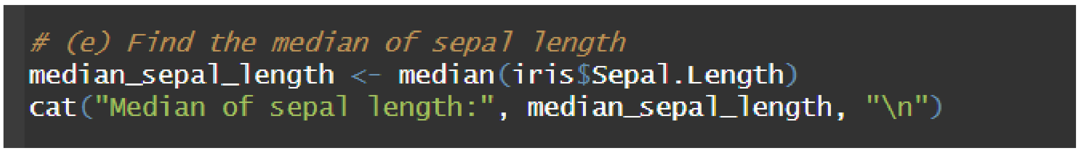
OUTPUT:



```
> cat("Mean of sepal length:", mean_sepal_length, "\n")
Mean of sepal length: 5.843333
```

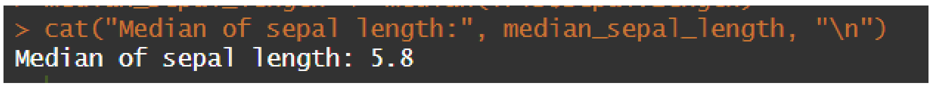
(e) Find the median of the sepal length.

CODE:



```
# (e) Find the median of sepal length
median_sepal_length <- median(iris$Sepal.Length)
cat("Median of sepal length:", median_sepal_length, "\n")
```

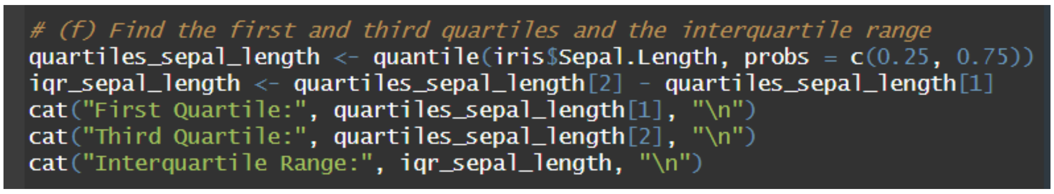
OUTPUT:



```
> median_sepal_length <- median(iris$Sepal.Length)
> cat("Median of sepal length:", median_sepal_length, "\n")
Median of sepal length: 5.8
```

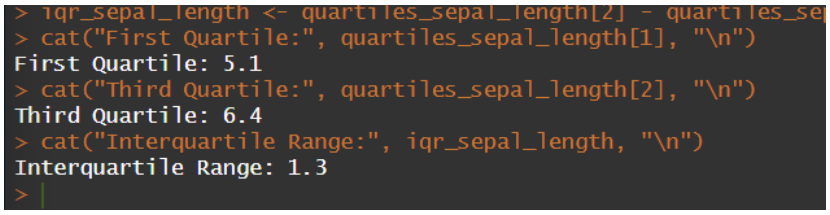
(f) Find the first and the third quartiles and hence the interquartile range.

CODE:



```
# (f) Find the first and third quartiles and the interquartile range
quartiles_sepal_length <- quantile(iris$Sepal.Length, probs = c(0.25, 0.75))
iqr_sepal_length <- quartiles_sepal_length[2] - quartiles_sepal_length[1]
cat("First Quartile:", quartiles_sepal_length[1], "\n")
cat("Third Quartile:", quartiles_sepal_length[2], "\n")
cat("Interquartile Range:", iqr_sepal_length, "\n")
```

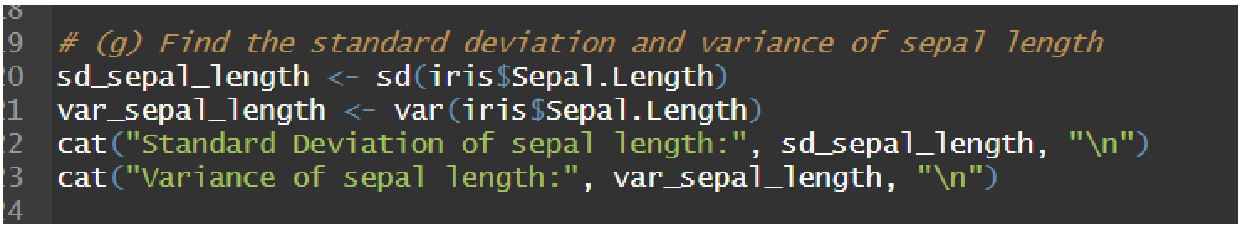
OUTPUT:



```
> iqr_sepal_length <- quartiles_sepal_length[2] - quartiles_se
> cat("First Quartile:", quartiles_sepal_length[1], "\n")
First Quartile: 5.1
> cat("Third Quartile:", quartiles_sepal_length[2], "\n")
Third Quartile: 6.4
> cat("Interquartile Range:", iqr_sepal_length, "\n")
Interquartile Range: 1.3
> 
```

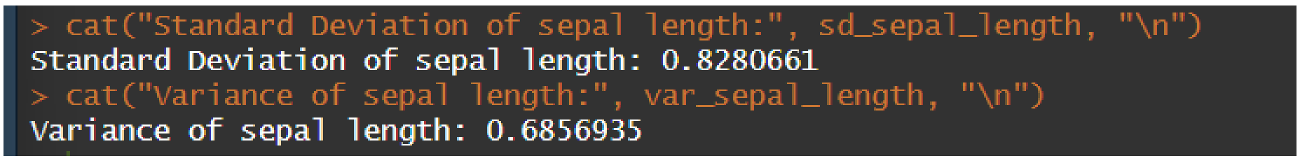
(g) Find the standard deviation and variance.

CODE:



```
# (g) Find the standard deviation and variance of sepal length
sd_sepal_length <- sd(iris$Sepal.Length)
var_sepal_length <- var(iris$Sepal.Length)
cat("Standard Deviation of sepal length:", sd_sepal_length, "\n")
cat("Variance of sepal length:", var_sepal_length, "\n")
```

OUTPUT:

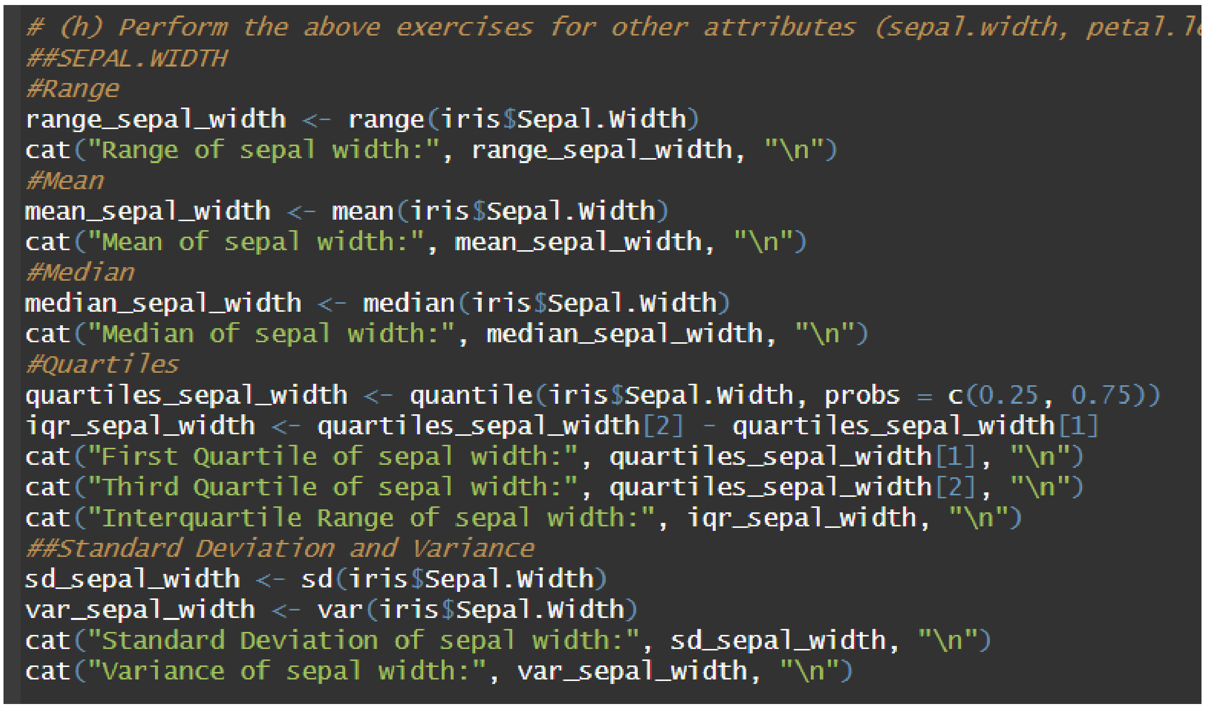


```
> cat("Standard Deviation of sepal length:", sd_sepal_length, "\n")
Standard Deviation of sepal length: 0.8280661
> cat("Variance of sepal length:", var_sepal_length, "\n")
Variance of sepal length: 0.6856935
```

(h) Try doing the above exercises for sepal.width, petal.length and petal.width.

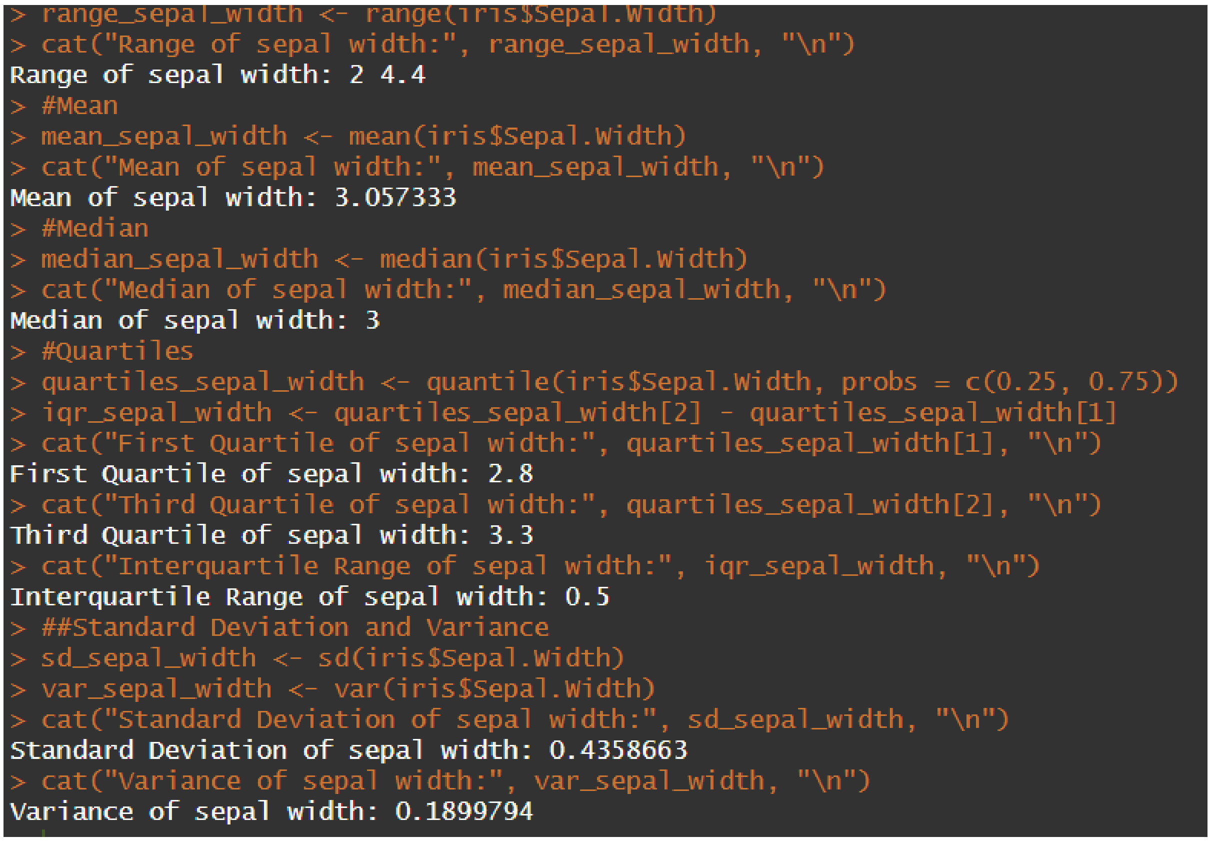
SEPAL.WIDTH

CODE:



```
# (h) Perform the above exercises for other attributes (sepal.width, petal.l
##SEPAL.WIDTH
#Range
range_sepal_width <- range(iris$Sepal.Width)
cat("Range of sepal width:", range_sepal_width, "\n")
#Mean
mean_sepal_width <- mean(iris$Sepal.Width)
cat("Mean of sepal width:", mean_sepal_width, "\n")
#Median
median_sepal_width <- median(iris$Sepal.Width)
cat("Median of sepal width:", median_sepal_width, "\n")
#Quartiles
quartiles_sepal_width <- quantile(iris$Sepal.Width, probs = c(0.25, 0.75))
iqr_sepal_width <- quartiles_sepal_width[2] - quartiles_sepal_width[1]
cat("First Quartile of sepal width:", quartiles_sepal_width[1], "\n")
cat("Third Quartile of sepal width:", quartiles_sepal_width[2], "\n")
cat("Interquartile Range of sepal width:", iqr_sepal_width, "\n")
##Standard Deviation and Variance
sd_sepal_width <- sd(iris$Sepal.Width)
var_sepal_width <- var(iris$Sepal.Width)
cat("Standard Deviation of sepal width:", sd_sepal_width, "\n")
cat("Variance of sepal width:", var_sepal_width, "\n")
```

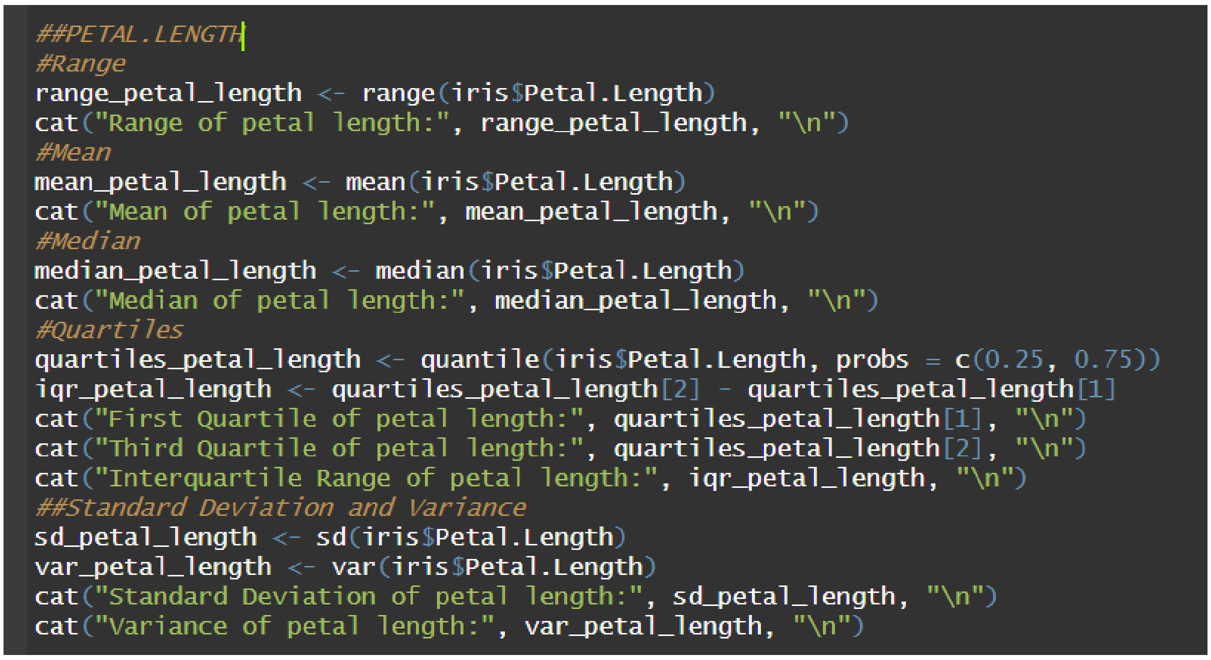
OUTPUT:



```
> range_sepal_width <- range(iris$Sepal.Width)
> cat("Range of sepal width:", range_sepal_width, "\n")
Range of sepal width: 2 4.4
> #Mean
> mean_sepal_width <- mean(iris$Sepal.Width)
> cat("Mean of sepal width:", mean_sepal_width, "\n")
Mean of sepal width: 3.057333
> #Median
> median_sepal_width <- median(iris$Sepal.Width)
> cat("Median of sepal width:", median_sepal_width, "\n")
Median of sepal width: 3
> #Quartiles
> quartiles_sepal_width <- quantile(iris$Sepal.Width, probs = c(0.25, 0.75))
> iqr_sepal_width <- quartiles_sepal_width[2] - quartiles_sepal_width[1]
> cat("First Quartile of sepal width:", quartiles_sepal_width[1], "\n")
First Quartile of sepal width: 2.8
> cat("Third Quartile of sepal width:", quartiles_sepal_width[2], "\n")
Third Quartile of sepal width: 3.3
> cat("Interquartile Range of sepal width:", iqr_sepal_width, "\n")
Interquartile Range of sepal width: 0.5
> ##Standard Deviation and Variance
> sd_sepal_width <- sd(iris$Sepal.Width)
> var_sepal_width <- var(iris$Sepal.Width)
> cat("Standard Deviation of sepal width:", sd_sepal_width, "\n")
Standard Deviation of sepal width: 0.4358663
> cat("Variance of sepal width:", var_sepal_width, "\n")
Variance of sepal width: 0.1899794
```

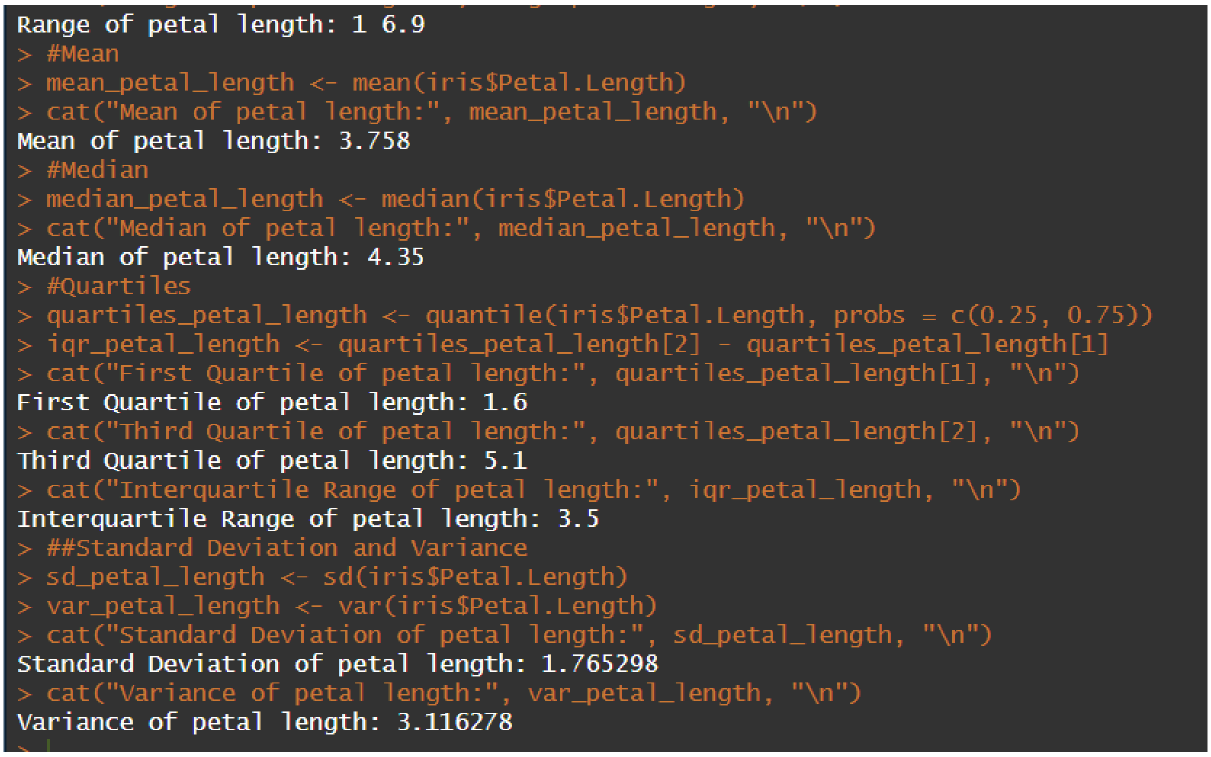
PETAL.LENGTH

INPUT:



```
##PETAL.LENGTH
#Range
range_petal_length <- range(iris$Petal.Length)
cat("Range of petal length:", range_petal_length, "\n")
#Mean
mean_petal_length <- mean(iris$Petal.Length)
cat("Mean of petal length:", mean_petal_length, "\n")
#Median
median_petal_length <- median(iris$Petal.Length)
cat("Median of petal length:", median_petal_length, "\n")
#Quartiles
quartiles_petal_length <- quantile(iris$Petal.Length, probs = c(0.25, 0.75))
iqr_petal_length <- quartiles_petal_length[2] - quartiles_petal_length[1]
cat("First Quartile of petal length:", quartiles_petal_length[1], "\n")
cat("Third Quartile of petal length:", quartiles_petal_length[2], "\n")
cat("Interquartile Range of petal length:", iqr_petal_length, "\n")
##Standard Deviation and Variance
sd_petal_length <- sd(iris$Petal.Length)
var_petal_length <- var(iris$Petal.Length)
cat("Standard Deviation of petal length:", sd_petal_length, "\n")
cat("Variance of petal length:", var_petal_length, "\n")
```

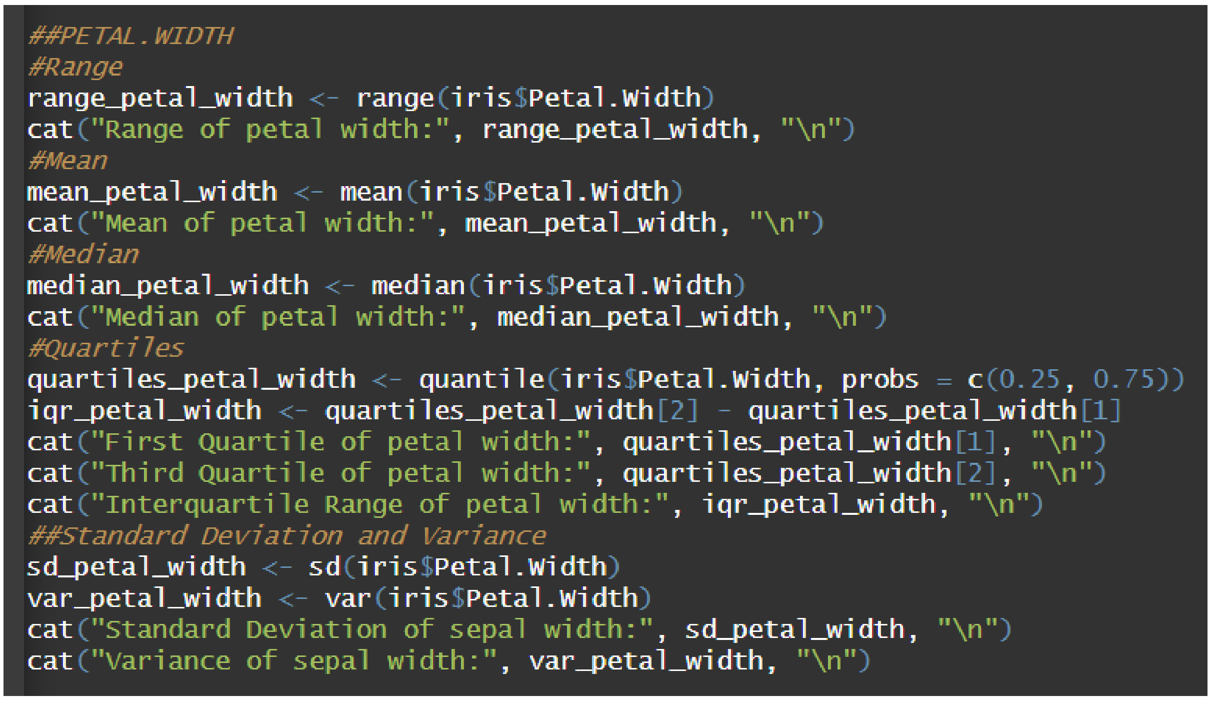
OUTPUT:



```
Range of petal length: 1 6.9
> #Mean
> mean_petal_length <- mean(iris$Petal.Length)
> cat("Mean of petal length:", mean_petal_length, "\n")
Mean of petal length: 3.758
> #Median
> median_petal_length <- median(iris$Petal.Length)
> cat("Median of petal length:", median_petal_length, "\n")
Median of petal length: 4.35
> #Quartiles
> quartiles_petal_length <- quantile(iris$Petal.Length, probs = c(0.25, 0.75))
> iqr_petal_length <- quartiles_petal_length[2] - quartiles_petal_length[1]
> cat("First Quartile of petal length:", quartiles_petal_length[1], "\n")
First Quartile of petal length: 1.6
> cat("Third Quartile of petal length:", quartiles_petal_length[2], "\n")
Third Quartile of petal length: 5.1
> cat("Interquartile Range of petal length:", iqr_petal_length, "\n")
Interquartile Range of petal length: 3.5
> ##Standard Deviation and Variance
> sd_petal_length <- sd(iris$Petal.Length)
> var_petal_length <- var(iris$Petal.Length)
> cat("Standard Deviation of petal length:", sd_petal_length, "\n")
Standard Deviation of petal length: 1.765298
> cat("Variance of petal length:", var_petal_length, "\n")
Variance of petal length: 3.116278
```

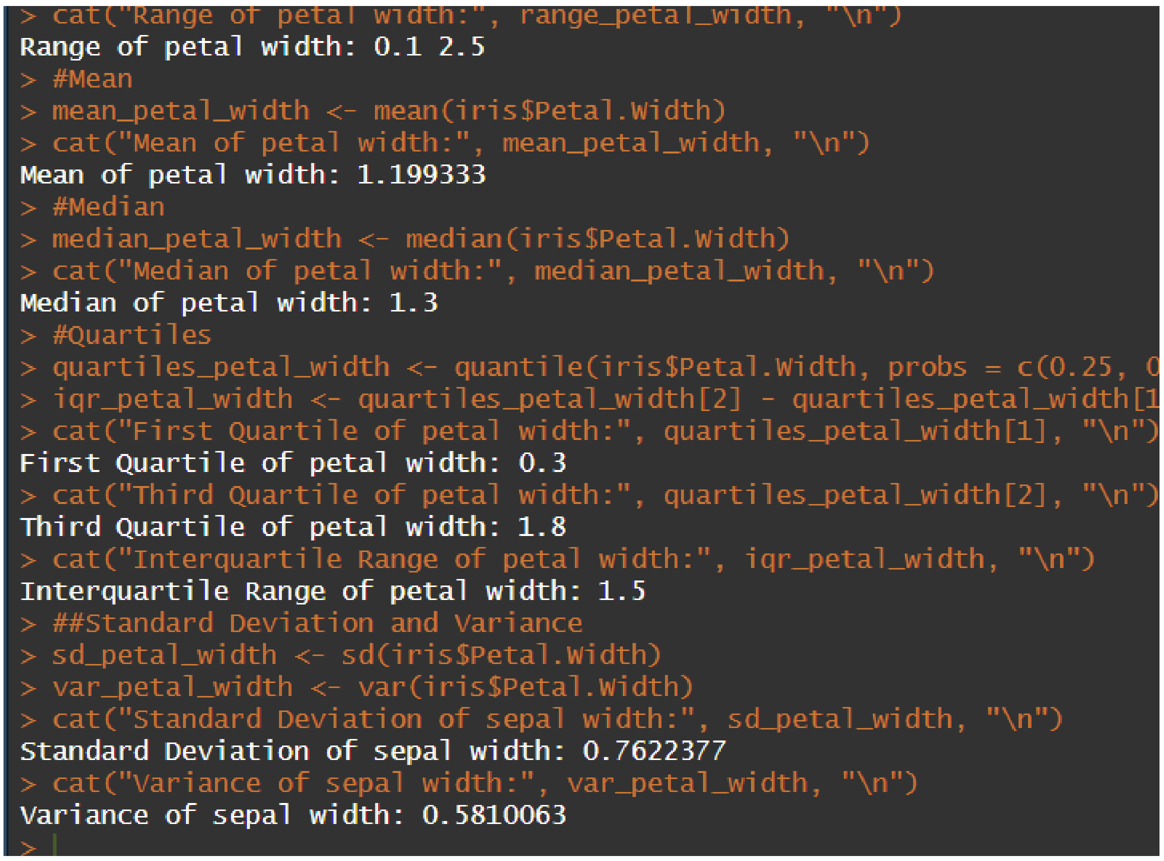
PETAL.WIDTH

INPUT:



```
##PETAL.WIDTH
#Range
range_petal_width <- range(iris$Petal.Width)
cat("Range of petal width:", range_petal_width, "\n")
#Mean
mean_petal_width <- mean(iris$Petal.Width)
cat("Mean of petal width:", mean_petal_width, "\n")
#Median
median_petal_width <- median(iris$Petal.Width)
cat("Median of petal width:", median_petal_width, "\n")
#Quartiles
quartiles_petal_width <- quantile(iris$Petal.Width, probs = c(0.25, 0.75))
iqr_petal_width <- quartiles_petal_width[2] - quartiles_petal_width[1]
cat("First Quartile of petal width:", quartiles_petal_width[1], "\n")
cat("Third Quartile of petal width:", quartiles_petal_width[2], "\n")
cat("Interquartile Range of petal width:", iqr_petal_width, "\n")
##Standard Deviation and Variance
sd_petal_width <- sd(iris$Petal.Width)
var_petal_width <- var(iris$Petal.Width)
cat("Standard Deviation of sepal width:", sd_petal_width, "\n")
cat("Variance of sepal width:", var_petal_width, "\n")
```

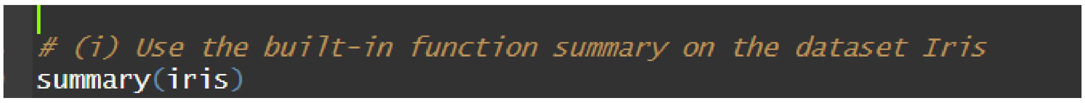
OUTPUT:



```
> cat("Range of petal width:", range_petal_width, "\n")
Range of petal width: 0.1 2.5
> #Mean
> mean_petal_width <- mean(iris$Petal.Width)
> cat("Mean of petal width:", mean_petal_width, "\n")
Mean of petal width: 1.199333
> #Median
> median_petal_width <- median(iris$Petal.Width)
> cat("Median of petal width:", median_petal_width, "\n")
Median of petal width: 1.3
> #Quartiles
> quartiles_petal_width <- quantile(iris$Petal.Width, probs = c(0.25, 0
> iqr_petal_width <- quartiles_petal_width[2] - quartiles_petal_width[1
> cat("First Quartile of petal width:", quartiles_petal_width[1], "\n")
First Quartile of petal width: 0.3
> cat("Third Quartile of petal width:", quartiles_petal_width[2], "\n")
Third Quartile of petal width: 1.8
> cat("Interquartile Range of petal width:", iqr_petal_width, "\n")
Interquartile Range of petal width: 1.5
> ##Standard Deviation and Variance
> sd_petal_width <- sd(iris$Petal.Width)
> var_petal_width <- var(iris$Petal.Width)
> cat("Standard Deviation of sepal width:", sd_petal_width, "\n")
Standard Deviation of sepal width: 0.7622377
> cat("Variance of sepal width:", var_petal_width, "\n")
Variance of sepal width: 0.5810063
>
```

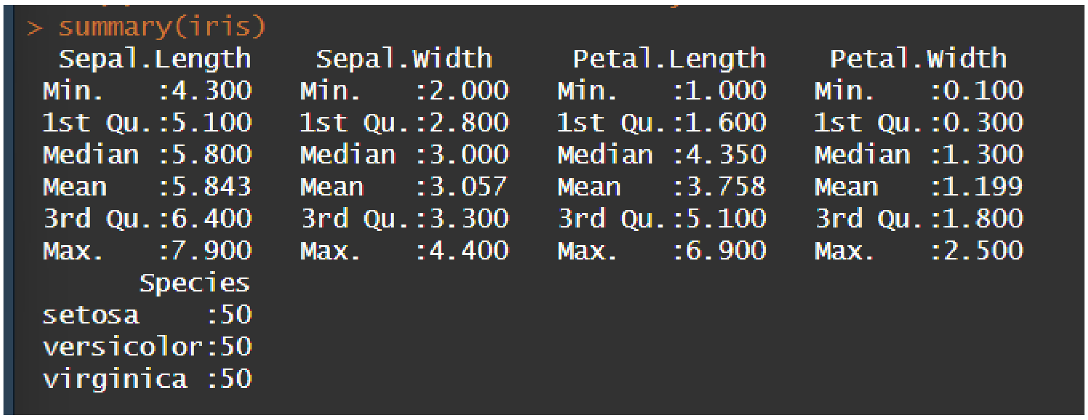
(i) Use the built-in function summary on the dataset Iris.

CODE:



```
# (i) Use the built-in function summary on the dataset Iris
summary(iris)
```

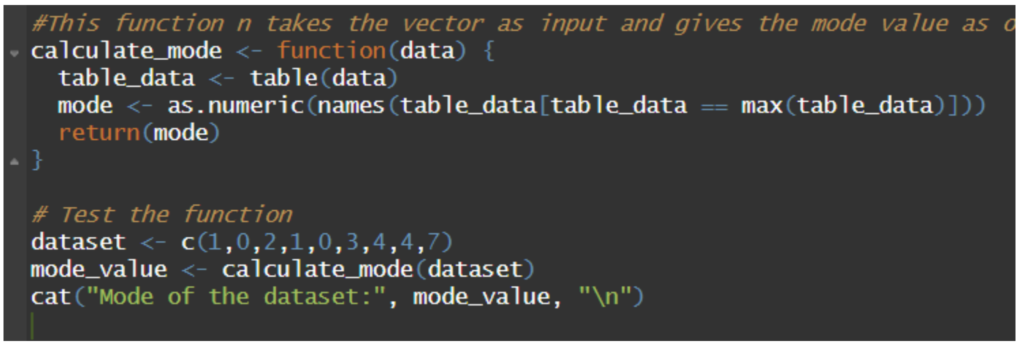
OUTPUT:



```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
 Median :5.800   Median :3.000   Median :4.350   Median :1.300  
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
       Species  
 setosa    :50  
 versicolor:50  
 virginica :50  
```

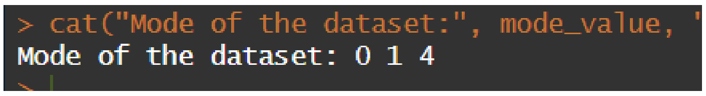
(5) R does not have a standard in-built function to calculate mode. So, we create a user function to calculate mode of a data set in R. This function n takes the vector as input and gives the mode value as output.

CODE:



```
#This function n takes the vector as input and gives the mode value as o
calculate_mode <- function(data) {
  table_data <- table(data)
  mode <- as.numeric(names(table_data[table_data == max(table_data)]))
  return(mode)
}

# Test the function
dataset <- c(1,0,2,1,0,3,4,4,7)
mode_value <- calculate_mode(dataset)
cat("Mode of the dataset:", mode_value, "\n")
```

OUTPUT:



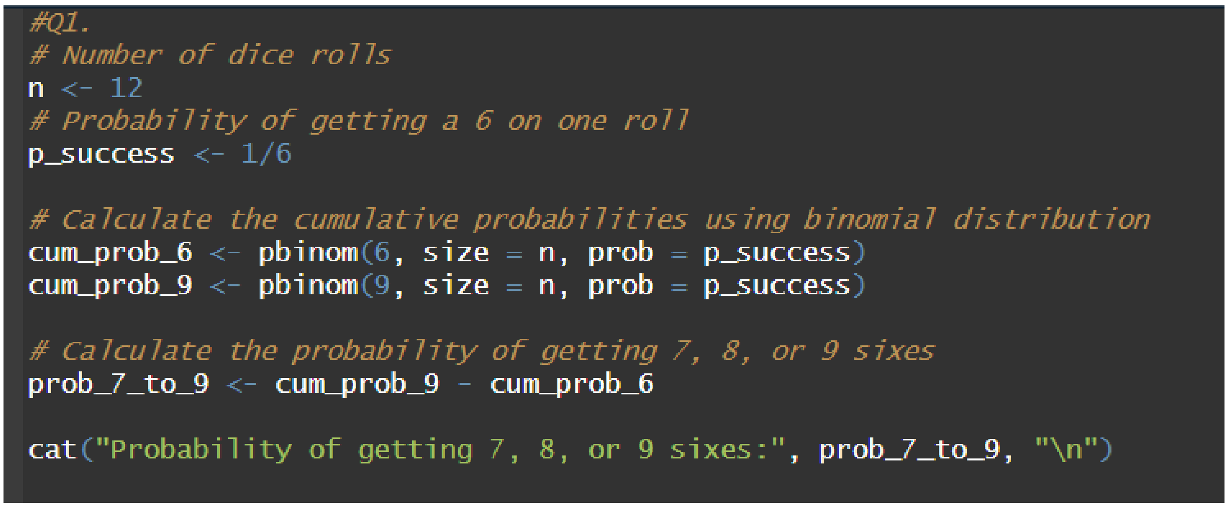
```
> cat("Mode of the dataset:", mode_value, '
Mode of the dataset: 0 1 4
>
```

Probability and Statistics (UCS410)

Experiment 3: Probability distributions

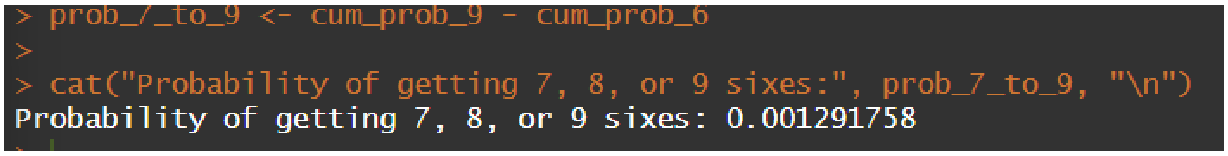
(1) Roll 12 dice simultaneously, and let X denotes the number of 6's that appear. Calculate the probability of getting 7, 8 or 9, 6's using R. (Try using the function pbinom; If we set S = {get a 6 on one roll}, P(S) = 1/6 and the rolls constitute Bernoulli trials; thus X ~ binom(size=12, prob=1/6) and we are looking for P(7 ≤ X ≤ 9).

CODE:



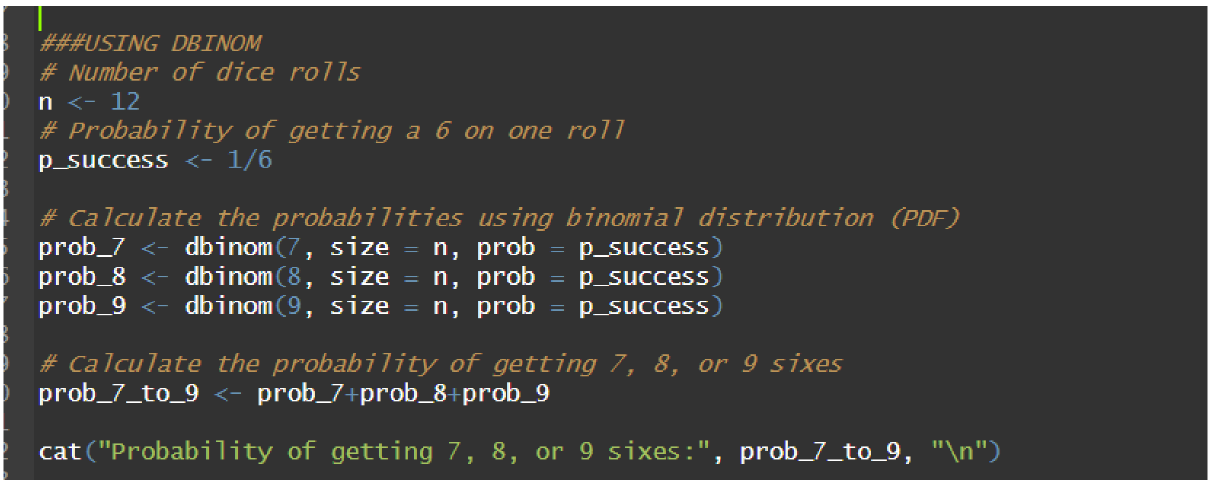
```r
#Q1.
# Number of dice rolls
n <- 12
# Probability of getting a 6 on one roll
p_success <- 1/6

# Calculate the cumulative probabilities using binomial distribution
cum_prob_6 <- pbinom(6, size = n, prob = p_success)
cum_prob_9 <- pbinom(9, size = n, prob = p_success)

# Calculate the probability of getting 7, 8, or 9 sixes
prob_7_to_9 <- cum_prob_9 - cum_prob_6

cat("Probability of getting 7, 8, or 9 sixes:", prob_7_to_9, "\n")
```

OUTPUT:



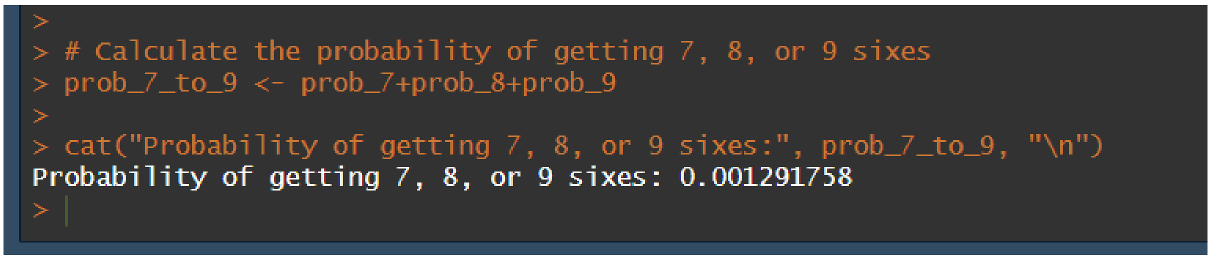
```
> prob_7_to_9 <- cum_prob_9 - cum_prob_6
>
> cat("Probability of getting 7, 8, or 9 sixes:", prob_7_to_9, "\n")
Probability of getting 7, 8, or 9 sixes: 0.001291758
```

CODE(USING dbinom):



```r
###USING DBINOM
# Number of dice rolls
n <- 12
# Probability of getting a 6 on one roll
p_success <- 1/6

# Calculate the probabilities using binomial distribution (PDF)
prob_7 <- dbinom(7, size = n, prob = p_success)
prob_8 <- dbinom(8, size = n, prob = p_success)
prob_9 <- dbinom(9, size = n, prob = p_success)

# Calculate the probability of getting 7, 8, or 9 sixes
prob_7_to_9 <- prob_7+prob_8+prob_9

cat("Probability of getting 7, 8, or 9 sixes:", prob_7_to_9, "\n")
```

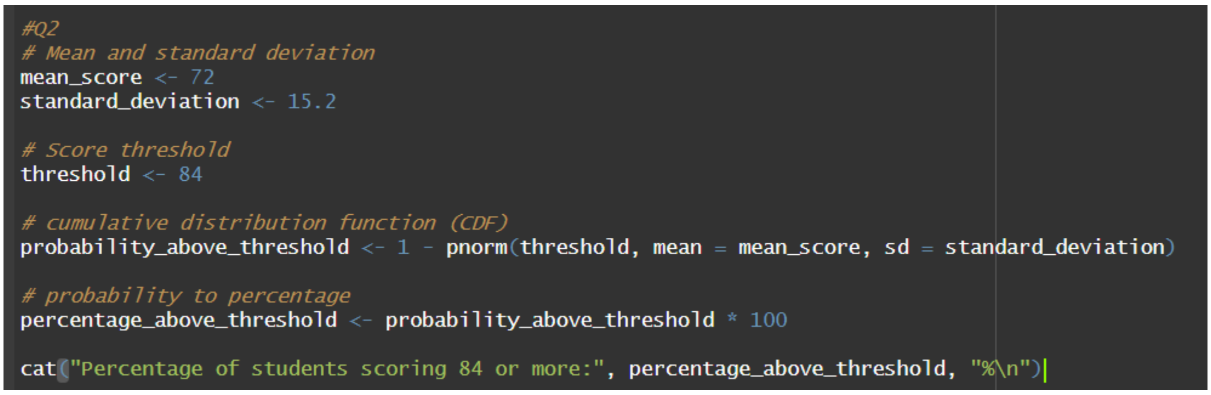
OUTPUT:



```
>
> # Calculate the probability of getting 7, 8, or 9 sixes
> prob_7_to_9 <- prob_7+prob_8+prob_9
>
> cat("Probability of getting 7, 8, or 9 sixes:", prob_7_to_9, "\n")
Probability of getting 7, 8, or 9 sixes: 0.001291758
>
```

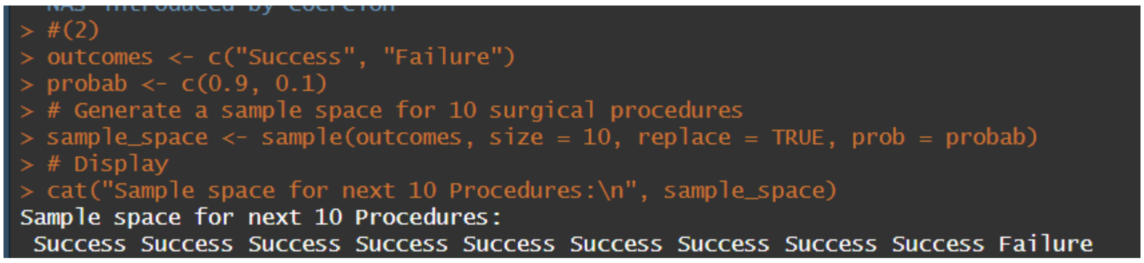
(2) Assume that the test scores of a college entrance exam fits a normal distribution. Furthermore, the mean test score is 72, and the standard deviation is 15.2. What is the percentage of students scoring 84 or more in the exam?

CODE:

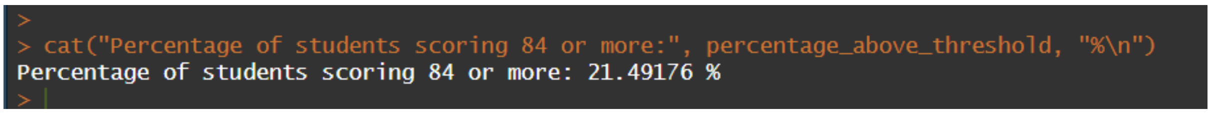


```
#Q2
# Mean and standard deviation
mean_score <- 72
standard_deviation <- 15.2

# Score threshold
threshold <- 84

# cumulative distribution function (CDF)
probability_above_threshold <- 1 - pnorm(threshold, mean = mean_score, sd = standard_deviation)

# probability to percentage
percentage_above_threshold <- probability_above_threshold * 100

cat("Percentage of students scoring 84 or more:", percentage_above_threshold, "%\n")
```

OUTPUT:



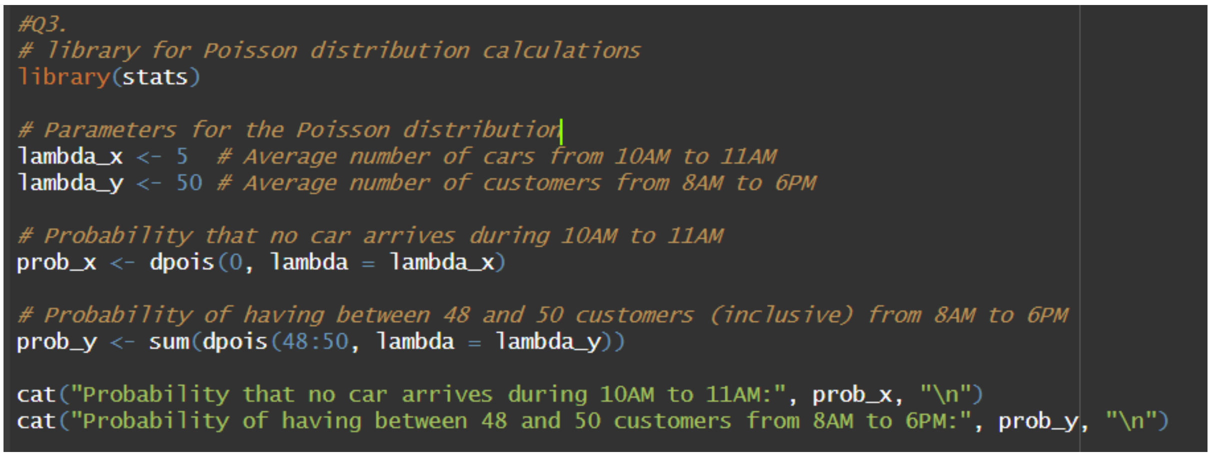
```
  NAs introduced by coercion
> #(2)
> outcomes <- c("Success", "Failure")
> probab <- c(0.9, 0.1)
> # Generate a sample space for 10 surgical procedures
> sample_space <- sample(outcomes, size = 10, replace = TRUE, prob = probab)
> # Display
> cat("Sample space for next 10 Procedures:\n", sample_space)
Sample space for next 10 Procedures:
 Success Success Success Success Success Success Success Success Success Failure
```



```
>
> cat("Percentage of students scoring 84 or more:", percentage_above_threshold, "%\n")
Percentage of students scoring 84 or more: 21.49176 %
>
```

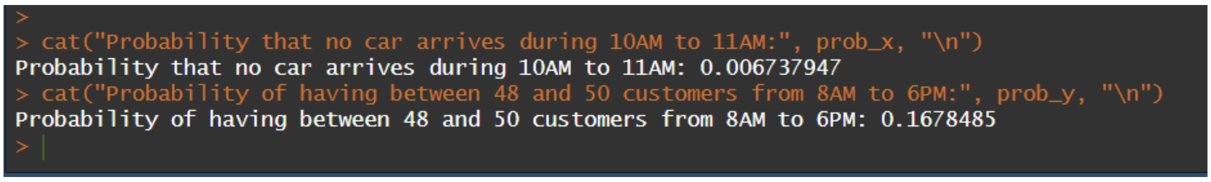
(3) On the average, five cars arrive at a particular car wash every hour. Let X count the number of cars that arrive from 10AM to 11AM, then X ~Poisson(λ = 5). What is probability that no car arrives during this time. Next, suppose the car wash above is in operation from 8AM to 6PM, and we let Y be the number of customers that appear in this period. Since this period covers a total of 10 hours, we get that Y ~ Poisson(λ = 5×10 = 50). What is the probability that there are between 48 and 50 customers, inclusive?

CODE:



```
#Q3.
# library for Poisson distribution calculations
library(stats)

# Parameters for the Poisson distribution
lambda_x <- 5   # Average number of cars from 10AM to 11AM
lambda_y <- 50 # Average number of customers from 8AM to 6PM

# Probability that no car arrives during 10AM to 11AM
prob_x <- dpois(0, lambda = lambda_x)

# Probability of having between 48 and 50 customers (inclusive) from 8AM to 6PM
prob_y <- sum(dpois(48:50, lambda = lambda_y))

cat("Probability that no car arrives during 10AM to 11AM:", prob_x, "\n")
cat("Probability of having between 48 and 50 customers from 8AM to 6PM:", prob_y, "\n")
```

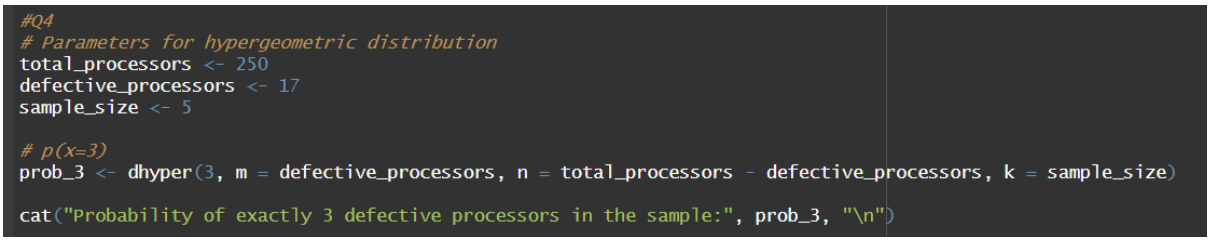
OUTPUT:



```
>
> cat("Probability that no car arrives during 10AM to 11AM:", prob_x, "\n")
Probability that no car arrives during 10AM to 11AM: 0.006737947
> cat("Probability of having between 48 and 50 customers from 8AM to 6PM:", prob_y, "\n")
Probability of having between 48 and 50 customers from 8AM to 6PM: 0.1678485
> 
```

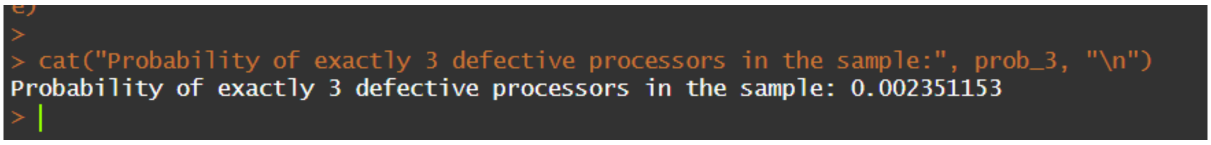
(4) Suppose in a certain shipment of 250 Pentium processors there are 17 defective processors. A quality control consultant randomly collects 5 processors for inspection to determine whether or not they are defective. Let X denote the number of defectives in the sample. Find the probability of exactly 3 defectives in the sample, that is, find P(X = 3).

CODE:



```
#Q4
# Parameters for hypergeometric distribution
total_processors <- 250
defective_processors <- 17
sample_size <- 5

# p(x=3)
prob_3 <- dhyper(3, m = defective_processors, n = total_processors - defective_processors, k = sample_size)

cat("Probability of exactly 3 defective processors in the sample:", prob_3, "\n")
```

OUTPUT:



```
e)
>
> cat("Probability of exactly 3 defective processors in the sample:", prob_3, "\n")
Probability of exactly 3 defective processors in the sample: 0.002351153
> 
```

(5) A recent national study showed that approximately 44.7% of college students have used Wikipedia as a source in at least one of their term papers. Let X equal the number of students in a random sample of size n = 31 who have used Wikipedia as a source.

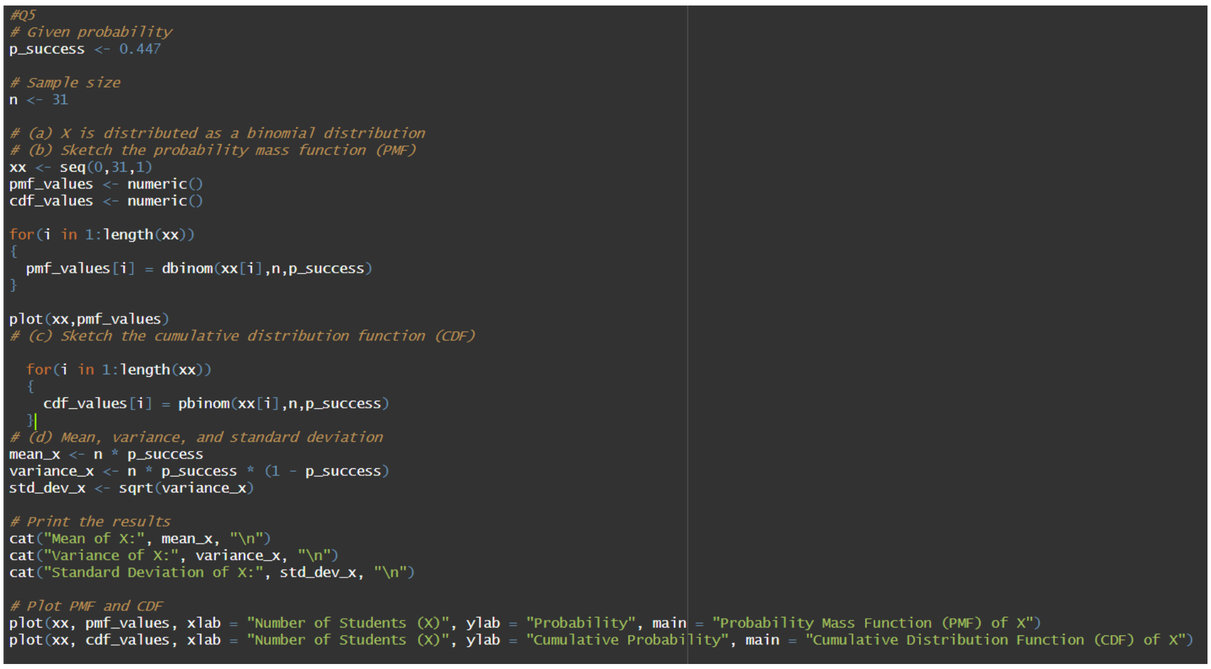
(a) How is X distributed?

(b) Sketch the probability mass function.

(c) Sketch the cumulative distribution function.

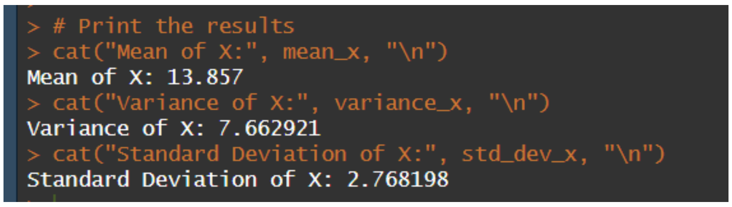
(d) Find mean, variance and standard deviation of X.

CODE:



```
#Q5
# Given probability
p_success <- 0.447

# Sample size
n <- 31

# (a) X is distributed as a binomial distribution
# (b) Sketch the probability mass function (PMF)
xx <- seq(0,31,1)
pmf_values <- numeric()
cdf_values <- numeric()

for(i in 1:length(xx))
{
  pmf_values[i] = dbinom(xx[i],n,p_success)
}

plot(xx,pmf_values)
# (c) Sketch the cumulative distribution function (CDF)

  for(i in 1:length(xx))
  {
    cdf_values[i] = pbinom(xx[i],n,p_success)
  }
# (d) Mean, variance, and standard deviation
mean_x <- n * p_success
variance_x <- n * p_success * (1 - p_success)
std_dev_x <- sqrt(variance_x)

# Print the results
cat("Mean of X:", mean_x, "\n")
cat("Variance of X:", variance_x, "\n")
cat("Standard Deviation of X:", std_dev_x, "\n")

# Plot PMF and CDF
plot(xx, pmf_values, xlab = "Number of Students (X)", ylab = "Probability", main = "Probability Mass Function (PMF) of X")
plot(xx, cdf_values, xlab = "Number of Students (X)", ylab = "Cumulative Probability", main = "Cumulative Distribution Function (CDF) of X")
```

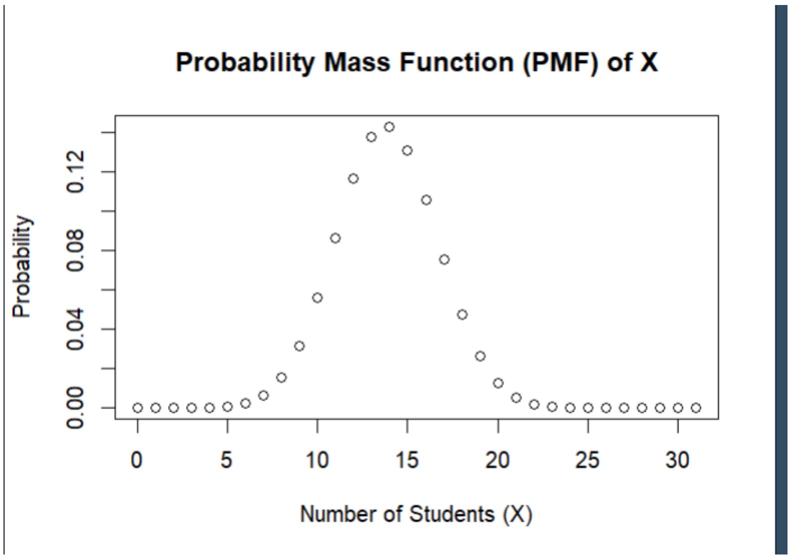
OUTPUT:



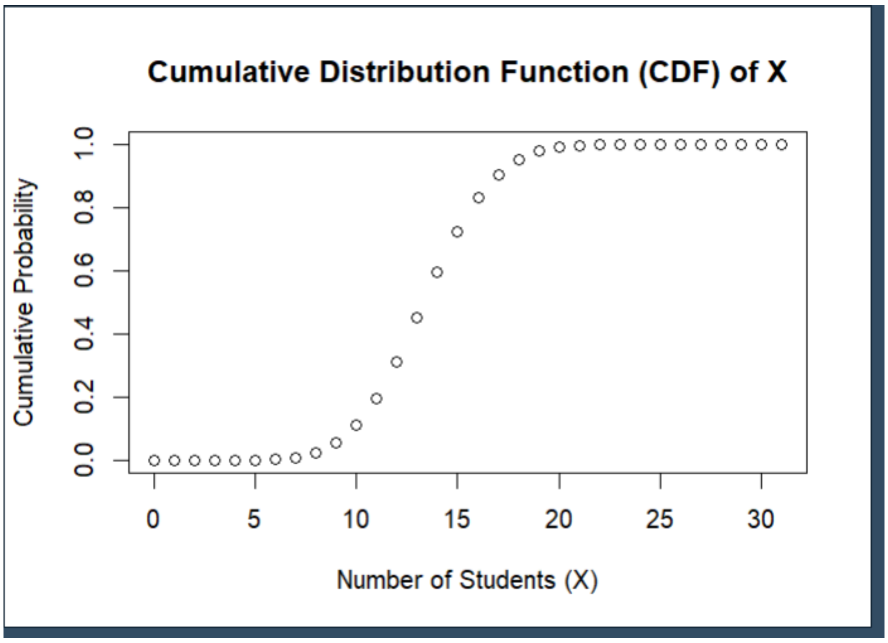
```
> # Print the results
> cat("Mean of X:", mean_x, "\n")
Mean of X: 13.857
> cat("Variance of X:", variance_x, "\n")
Variance of X: 7.662921
> cat("Standard Deviation of X:", std_dev_x, "\n")
Standard Deviation of X: 2.768198
```

PLOTS:

PDF: plot(xx, pmf_values, xlab = "Number of Students (X)", ylab = "Probability", main = "Probability Mass Function (PMF) of X")



CDF: plot(xx, cdf_values, xlab = "Number of Students (X)", ylab = "Cumulative Probability", main = "Cumulative Distribution Function (CDF) of X")



Probability and Statistics (UCS410)

Experiment 4: Mathematical Expectation, Moments and Functions of Random Variables

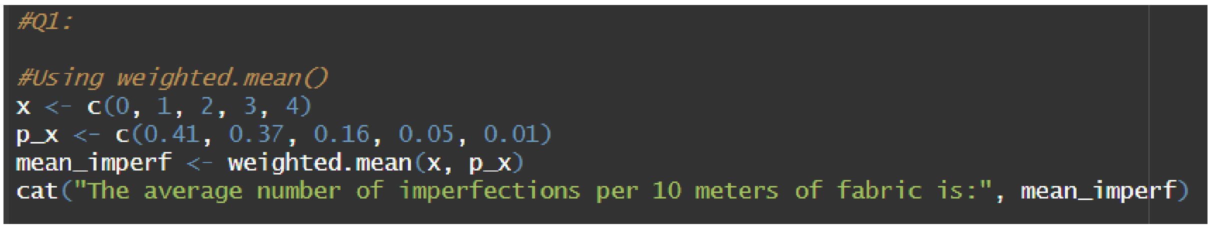
1. The probability distribution of X, the number of imperfections per 10 meters of a synthetic fabric in continuous rolls of uniform width, is given as

x	0	1	2	3	4
$p(x)$	0.41	0.37	0.16	0.05	0.01

Find the average number of imperfections per 10 meters of this fabric.

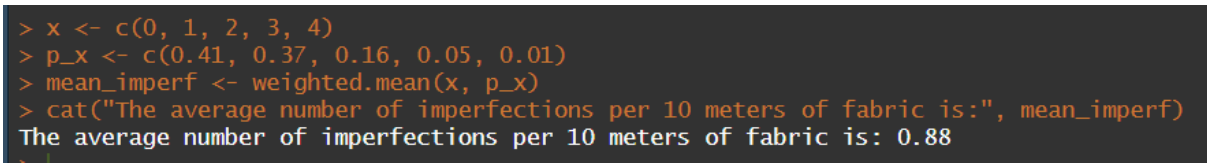
(Try functions **sum( )**, **weighted.mean( )**, c(a **%*%** b) to find expected value/mean.

CODE (using weighted.mean( )):



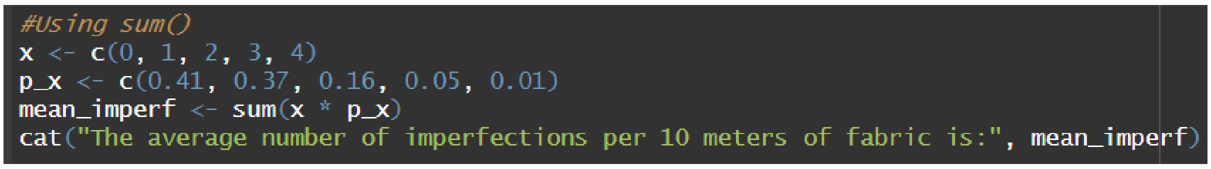
```r
#Q1:

#Using weighted.mean()
x <- c(0, 1, 2, 3, 4)
p_x <- c(0.41, 0.37, 0.16, 0.05, 0.01)
mean_imperf <- weighted.mean(x, p_x)
cat("The average number of imperfections per 10 meters of fabric is:", mean_imperf)
```

OUTPUT:



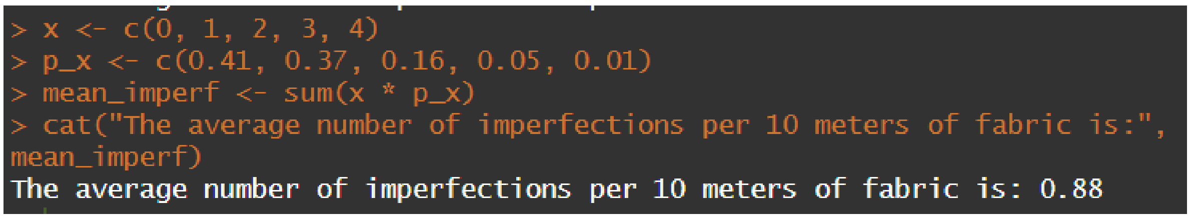
```
> x <- c(0, 1, 2, 3, 4)
> p_x <- c(0.41, 0.37, 0.16, 0.05, 0.01)
> mean_imperf <- weighted.mean(x, p_x)
> cat("The average number of imperfections per 10 meters of fabric is:", mean_imperf)
The average number of imperfections per 10 meters of fabric is: 0.88
```

CODE(sum()):



```r
#Using sum()
x <- c(0, 1, 2, 3, 4)
p_x <- c(0.41, 0.37, 0.16, 0.05, 0.01)
mean_imperf <- sum(x * p_x)
cat("The average number of imperfections per 10 meters of fabric is:", mean_imperf)
```

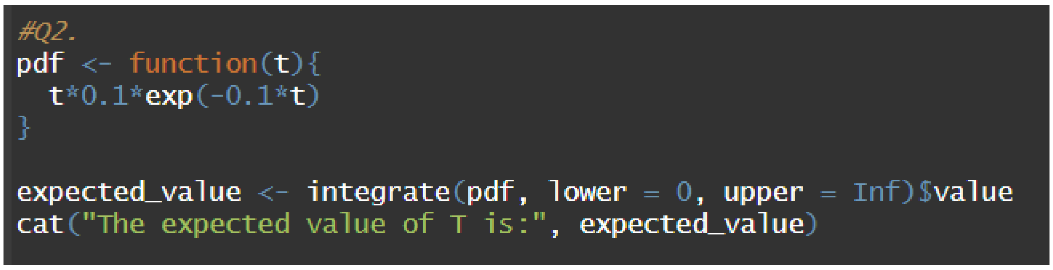
OUTPUT:



```
> x <- c(0, 1, 2, 3, 4)
> p_x <- c(0.41, 0.37, 0.16, 0.05, 0.01)
> mean_imperf <- sum(x * p_x)
> cat("The average number of imperfections per 10 meters of fabric is:",
mean_imperf)
The average number of imperfections per 10 meters of fabric is: 0.88
```

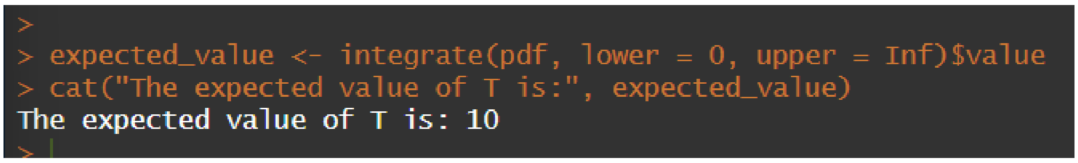
2. The time T, in days, required for the completion of a contracted project is a random variable with probability density function $f(t) = 0.1\ e^{(-0.1t)}$ for $t > 0$ and 0 otherwise. Find the expected value of T.
Use function **integrate( )** to find the expected value of continuous random variable T.

CODE:



```
#Q2.
pdf <- function(t){
  t*0.1*exp(-0.1*t)
}

expected_value <- integrate(pdf, lower = 0, upper = Inf)$value
cat("The expected value of T is:", expected_value)
```

OUTPUT:



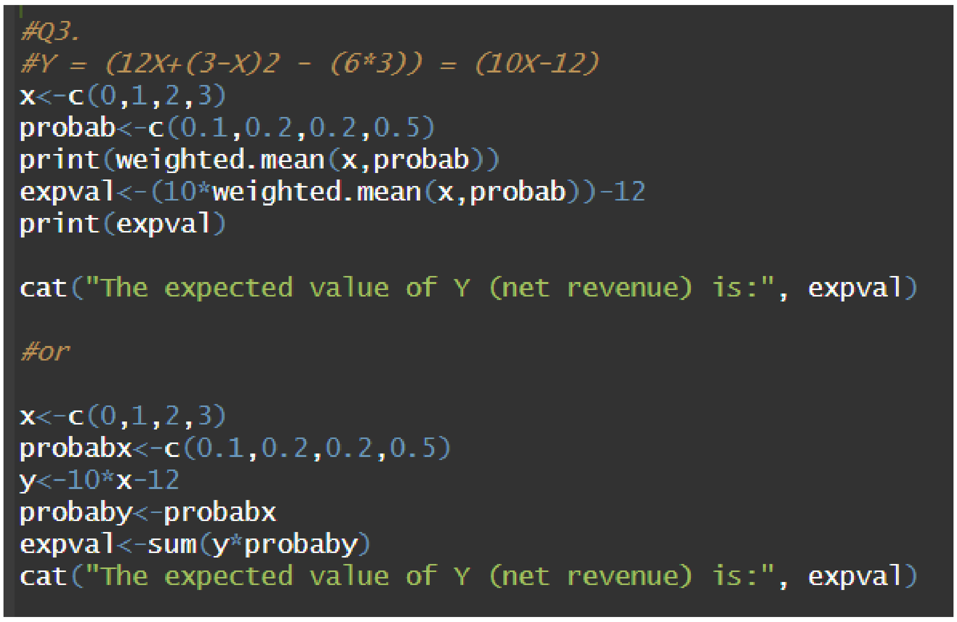
```
>
> expected_value <- integrate(pdf, lower = 0, upper = Inf)$value
> cat("The expected value of T is:", expected_value)
The expected value of T is: 10
>
```

3. A bookstore purchases three copies of a book at \$6.00 each and sells them for \$12.00 each. Unsold copies are returned for \$2.00 each. Let X = {number of copies sold} and Y = {net revenue}. If the probability mass function of X is

x	0	1	2	3
$p(x)$	0.1	0.2	0.2	0.5

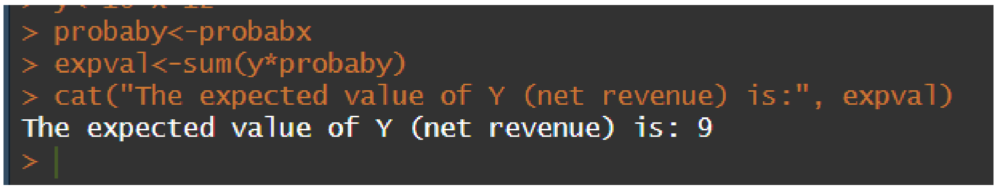
Find the expected value of Y.

CODE:



```
#Q3.
#Y = (12X+(3-X)2 - (6*3)) = (10X-12)
x<-c(0,1,2,3)
probab<-c(0.1,0.2,0.2,0.5)
print(weighted.mean(x,probab))
expval<-(10*weighted.mean(x,probab))-12
print(expval)

cat("The expected value of Y (net revenue) is:", expval)

#or

x<-c(0,1,2,3)
probabx<-c(0.1,0.2,0.2,0.5)
y<-10*x-12
probaby<-probabx
expval<-sum(y*probaby)
cat("The expected value of Y (net revenue) is:", expval)
```

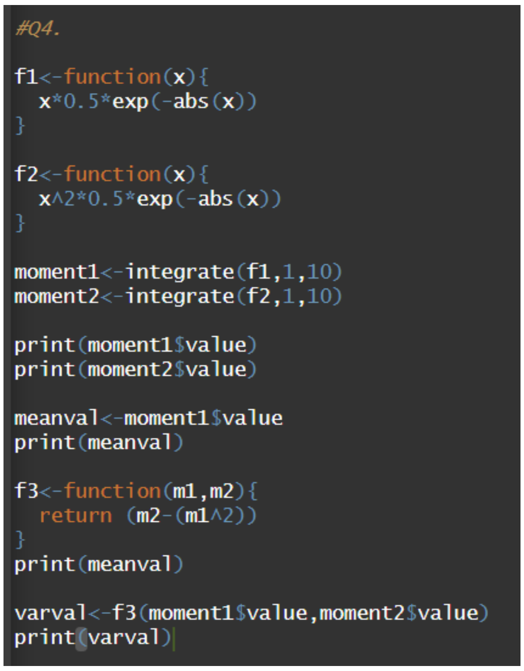
OUTPUT:



```
> probaby<-probabx
> expval<-sum(y*probaby)
> cat("The expected value of Y (net revenue) is:", expval)
The expected value of Y (net revenue) is: 9
> 
```

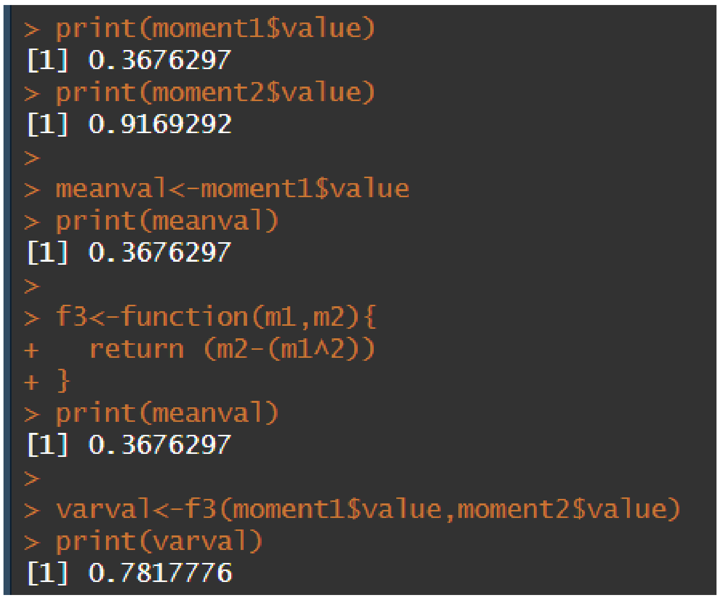
4. Find the first and second moments about the origin of the random variable X with probability density function $f(x) = 0.5e^{-|x|}$, $1 < x < 10$ and 0 otherwise. Further use the results to find Mean and Variance.
   (kth moment = E(X^k), Mean = first moment and Variance = second moment – Mean^2.

CODE:



```
#Q4.

f1<-function(x){
  x*0.5*exp(-abs(x))
}

f2<-function(x){
  x^2*0.5*exp(-abs(x))
}

moment1<-integrate(f1,1,10)
moment2<-integrate(f2,1,10)

print(moment1$value)
print(moment2$value)

meanval<-moment1$value
print(meanval)

f3<-function(m1,m2){
  return (m2-(m1^2))
}
print(meanval)

varval<-f3(moment1$value,moment2$value)
print(varval)
```

OUTPUT:



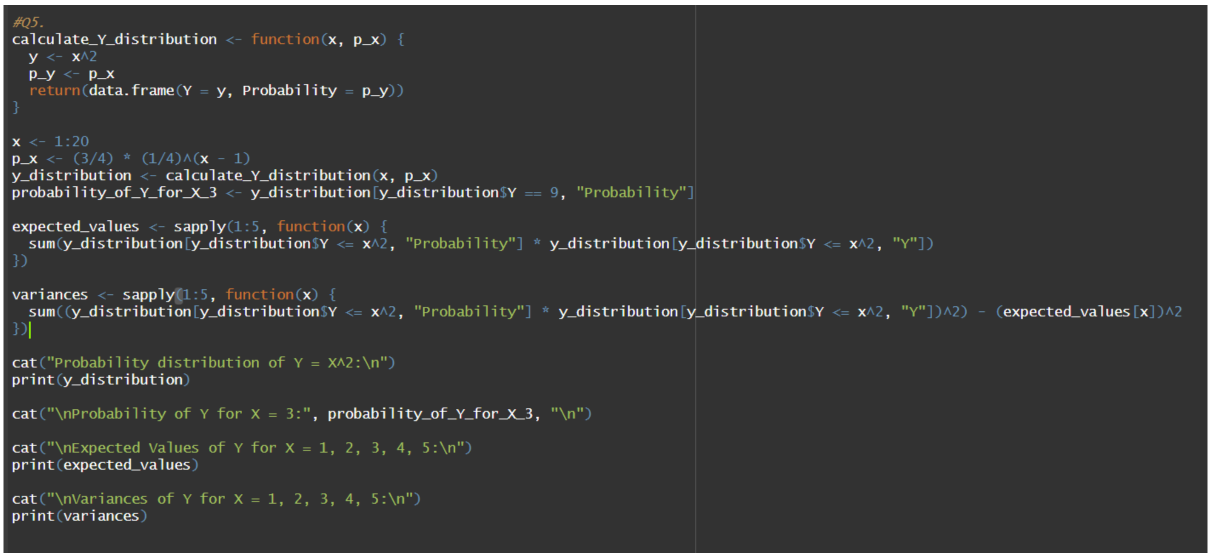
```
> print(moment1$value)
[1] 0.3676297
> print(moment2$value)
[1] 0.9169292
> 
> meanval<-moment1$value
> print(meanval)
[1] 0.3676297
> 
> f3<-function(m1,m2){
+   return (m2-(m1^2))
+ }
> print(meanval)
[1] 0.3676297
> 
> varval<-f3(moment1$value,moment2$value)
> print(varval)
[1] 0.7817776
```

5. Let X be a geometric random variable with probability distribution

$$f(x) = \frac{3}{4}\left(\frac{1}{4}\right)^{x-1}, x = 1,2,3, \ldots$$

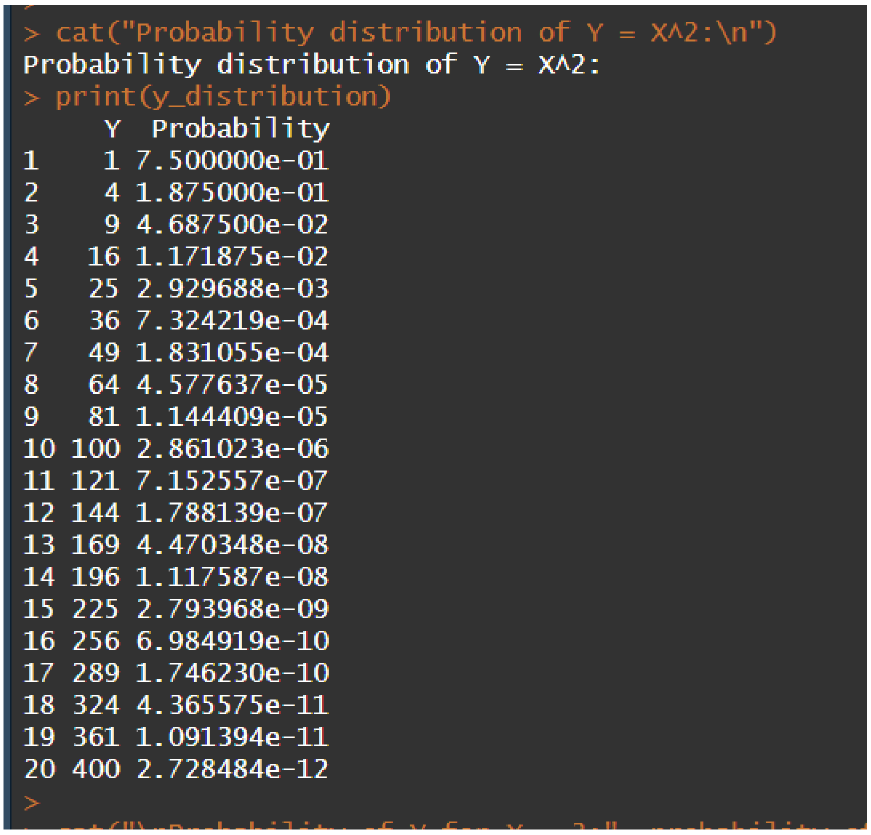
Write a function to find the probability distribution of the random variable Y = X^2 and find probability of Y for X = 3. Further, use it to find the expected value and variance of Y for X = 1,2,3,4,5.

CODE:

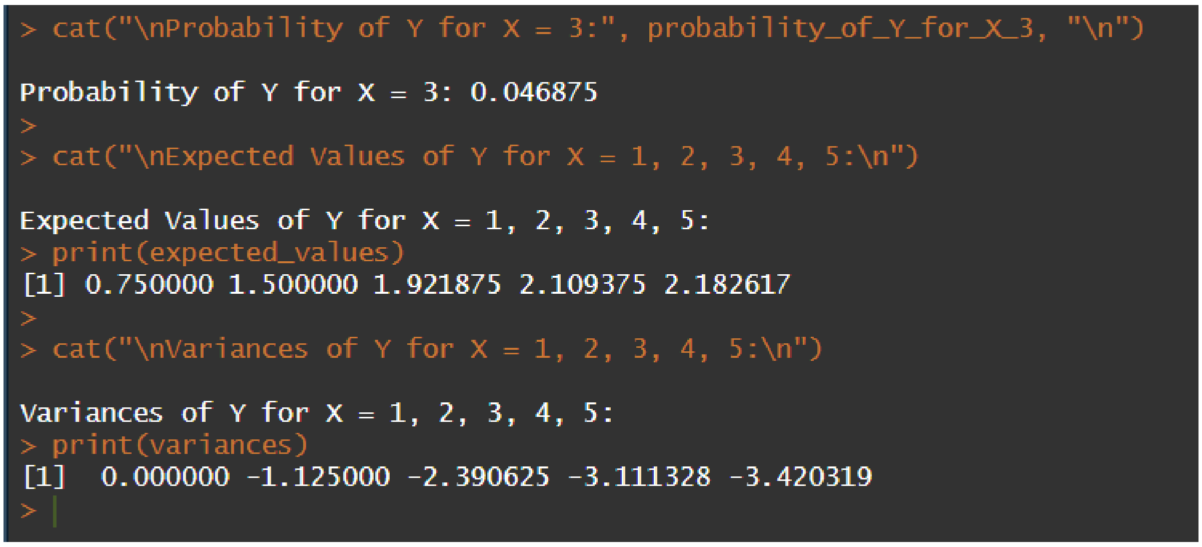


```
#Q5.
calculate_Y_distribution <- function(x, p_x) {
  y <- x^2
  p_y <- p_x
  return(data.frame(Y = y, Probability = p_y))
}

x <- 1:20
p_x <- (3/4) * (1/4)^(x - 1)
y_distribution <- calculate_Y_distribution(x, p_x)
probability_of_Y_for_X_3 <- y_distribution[y_distribution$Y == 9, "Probability"]

expected_values <- sapply(1:5, function(x) {
  sum(y_distribution[y_distribution$Y <= x^2, "Probability"] * y_distribution[y_distribution$Y <= x^2, "Y"])
})

variances <- sapply(1:5, function(x) {
  sum((y_distribution[y_distribution$Y <= x^2, "Probability"] * y_distribution[y_distribution$Y <= x^2, "Y"])^2) - (expected_values[x])^2
})

cat("Probability distribution of Y = X^2:\n")
print(y_distribution)

cat("\nProbability of Y for X = 3:", probability_of_Y_for_X_3, "\n")

cat("\nExpected Values of Y for X = 1, 2, 3, 4, 5:\n")
print(expected_values)

cat("\nVariances of Y for X = 1, 2, 3, 4, 5:\n")
print(variances)
```

OUTPUT:



```
> cat("Probability distribution of Y = X^2:\n")
Probability distribution of Y = X^2:
> print(y_distribution)
     Y  Probability
1    1 7.500000e-01
2    4 1.875000e-01
3    9 4.687500e-02
4   16 1.171875e-02
5   25 2.929688e-03
6   36 7.324219e-04
7   49 1.831055e-04
8   64 4.577637e-05
9   81 1.144409e-05
10 100 2.861023e-06
11 121 7.152557e-07
12 144 1.788139e-07
13 169 4.470348e-08
14 196 1.117587e-08
15 225 2.793968e-09
16 256 6.984919e-10
17 289 1.746230e-10
18 324 4.365575e-11
19 361 1.091394e-11
20 400 2.728484e-12
>
```


```
> cat("\nProbability of Y for X = 3:", probability_of_Y_for_X_3, "\n")

Probability of Y for X = 3: 0.046875
>
> cat("\nExpected Values of Y for X = 1, 2, 3, 4, 5:\n")

Expected Values of Y for X = 1, 2, 3, 4, 5:
> print(expected_values)
[1] 0.750000 1.500000 1.921875 2.109375 2.182617
>
> cat("\nVariances of Y for X = 1, 2, 3, 4, 5:\n")

Variances of Y for X = 1, 2, 3, 4, 5:
> print(variances)
[1]  0.000000 -1.125000 -2.390625 -3.111328 -3.420319
>
```

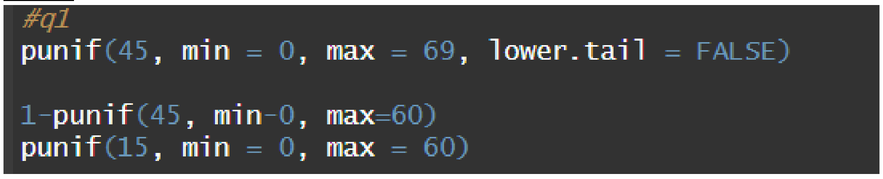
Probability and Statistics (UCS410)

Experiment 5: Continuous Probability Distributions

1. **Consider that X is the time (in minutes) that a person has to wait in order to take a flight. If each flight takes off each hour X ~ U(0, 60). Find the probability that**

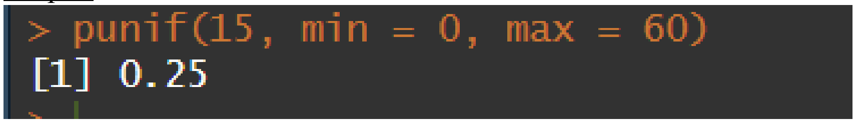
   (a) waiting time is more than 45 minutes, and

   Code:



```
#q1
punif(45, min = 0, max = 69, lower.tail = FALSE)

1-punif(45, min=0, max=60)
punif(15, min = 0, max = 60)
```

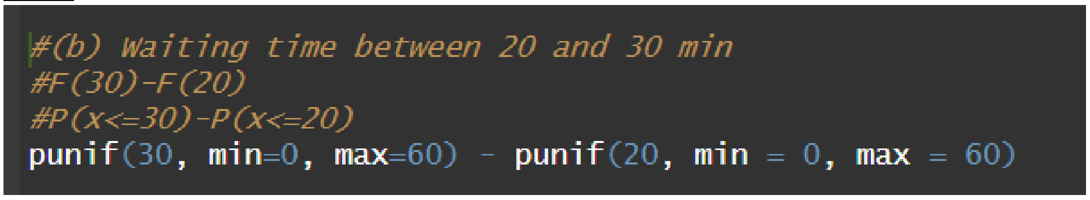
   Output:



```
> punif(15, min = 0, max = 60)
[1] 0.25
```

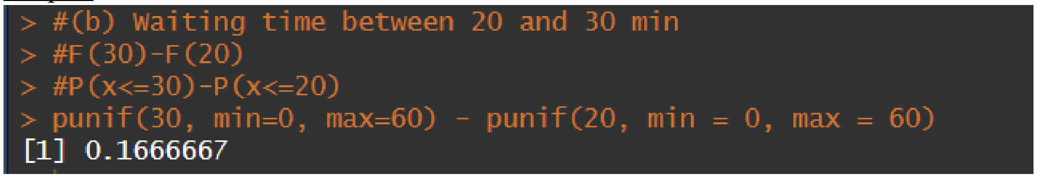
   (b) waiting time lies between 20 and 30 minutes.

   Code:



```
#(b) Waiting time between 20 and 30 min
#F(30)-F(20)
#P(x<=30)-P(x<=20)
punif(30, min=0, max=60) - punif(20, min = 0, max = 60)
```

   Output:

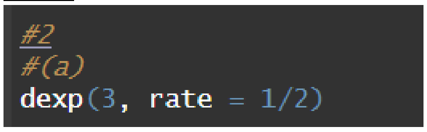


```
> #(b) Waiting time between 20 and 30 min
> #F(30)-F(20)
> #P(x<=30)-P(x<=20)
> punif(30, min=0, max=60) - punif(20, min = 0, max = 60)
[1] 0.1666667
```

2. **The time (in hours) required to repair a machine is an exponential distributed random variable with parameter $\lambda = 1/2$.**

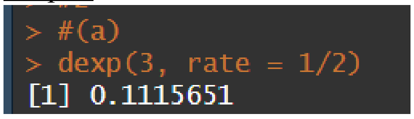
   a) Find the value of density function at x = 3.

   Code:



```
#2
#(a)
dexp(3, rate = 1/2)
```

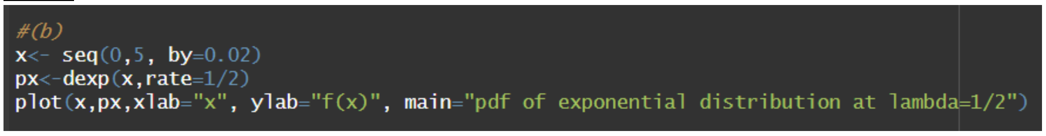
   Output:



```
> #(a)
> dexp(3, rate = 1/2)
[1] 0.1115651
```

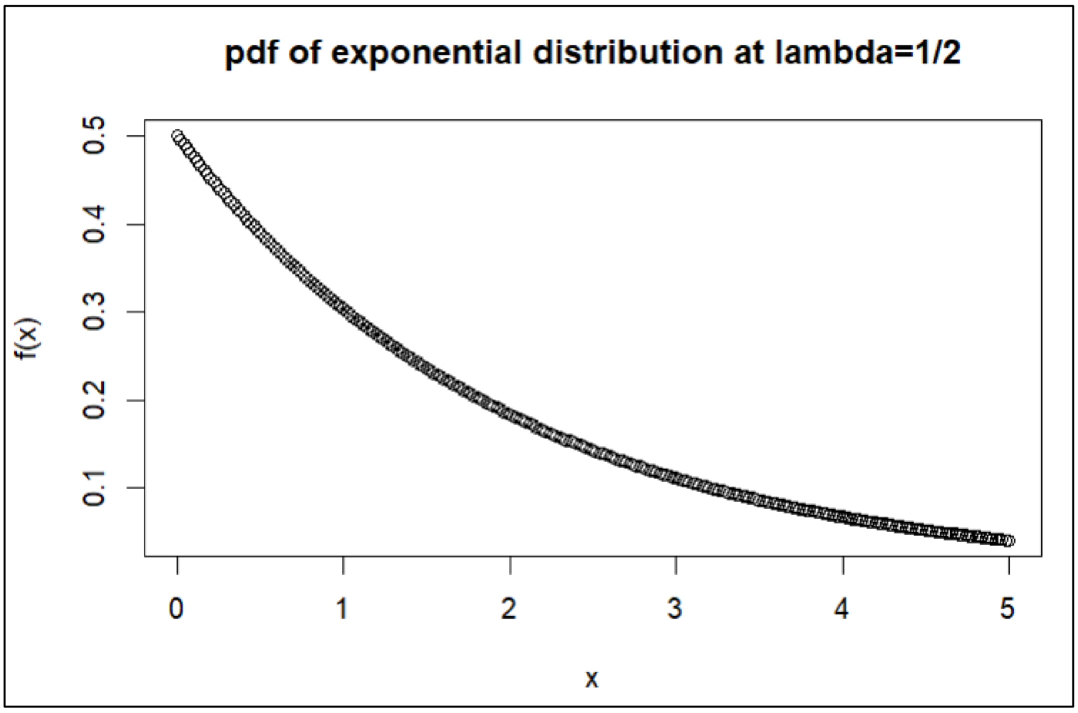
   b) Plot the graph of exponential probability distribution for $0 \le x \le 5$.

   Code:



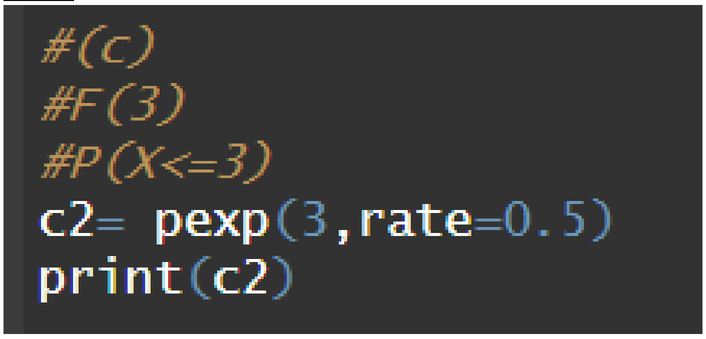
```
#(b)
x<- seq(0,5, by=0.02)
px<-dexp(x,rate=1/2)
plot(x,px,xlab="x", ylab="f(x)", main="pdf of exponential distribution at lambda=1/2")
```

   Output:



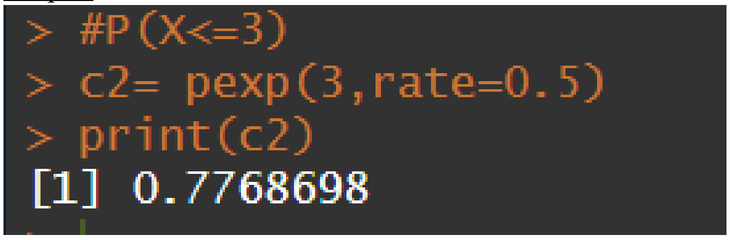
c) Find the probability that a repair time takes at most 3 hours.

Code:



```
#(c)
#F(3)
#P(X<=3)
c2= pexp(3,rate=0.5)
print(c2)
```

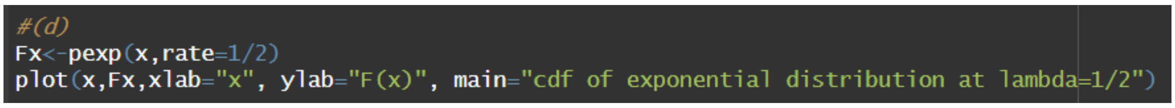
Output:



```
> #P(X<=3)
> c2= pexp(3,rate=0.5)
> print(c2)
[1] 0.7768698
```

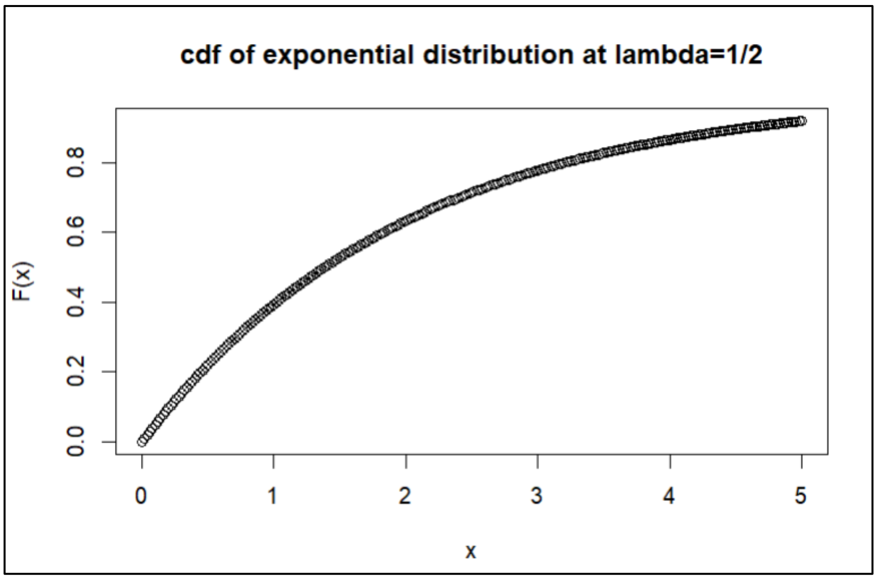
d) Plot the graph of cumulative exponential probabilities for 0 ≤ x ≤ 5.

Code:



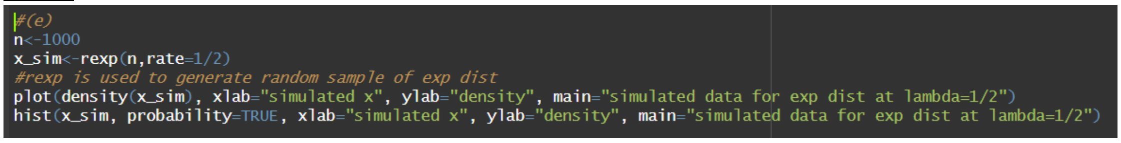
```
#(d)
Fx<-pexp(x,rate=1/2)
plot(x,Fx,xlab="x", ylab="F(x)", main="cdf of exponential distribution at lambda=1/2")
```

Output:



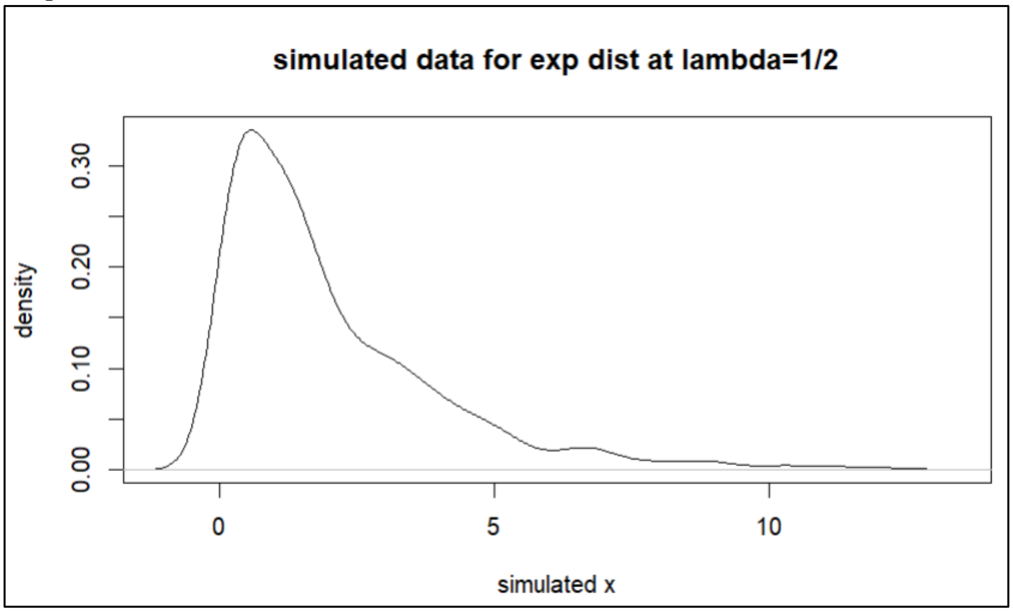
e) **Simulate 1000 exponential distributed random numbers with λ = 1⁄2 and plot the simulated data.**

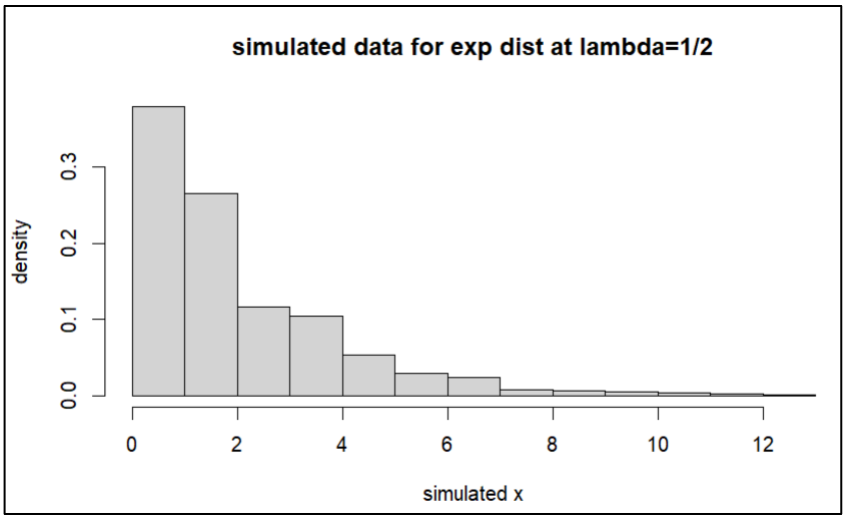
Code:



```
#(e)
n<-1000
x_sim<-rexp(n,rate=1/2)
#rexp is used to generate random sample of exp dist
plot(density(x_sim), xlab="simulated x", ylab="density", main="simulated data for exp dist at lambda=1/2")
hist(x_sim, probability=TRUE, xlab="simulated x", ylab="density", main="simulated data for exp dist at lambda=1/2")
```

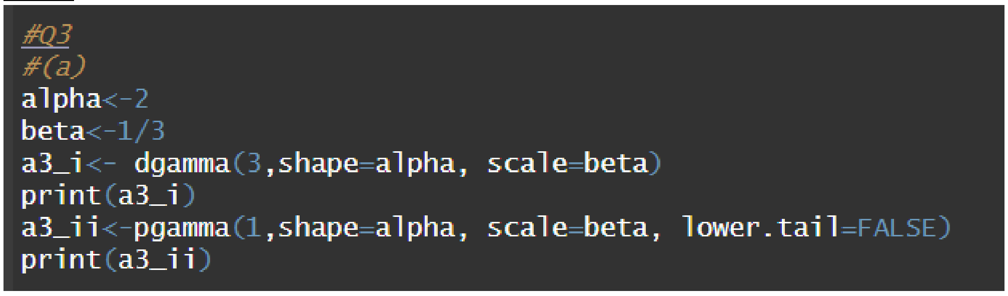
Output:





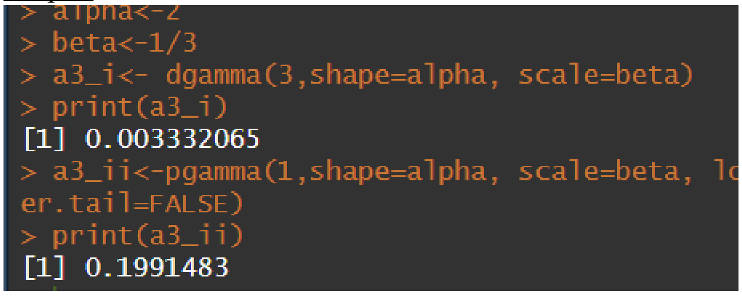
3. **The lifetime of certain equipment is described by a random variable X that follows Gamma distribution with parameters α = 2 and β = 1/3.**
   a) **Find the probability that the lifetime of equipment is at least 1 unit of time.**

   Code:



```
#Q3
#(a)
alpha<-2
beta<-1/3
a3_i<- dgamma(3,shape=alpha, scale=beta)
print(a3_i)
a3_ii<-pgamma(1,shape=alpha, scale=beta, lower.tail=FALSE)
print(a3_ii)
```

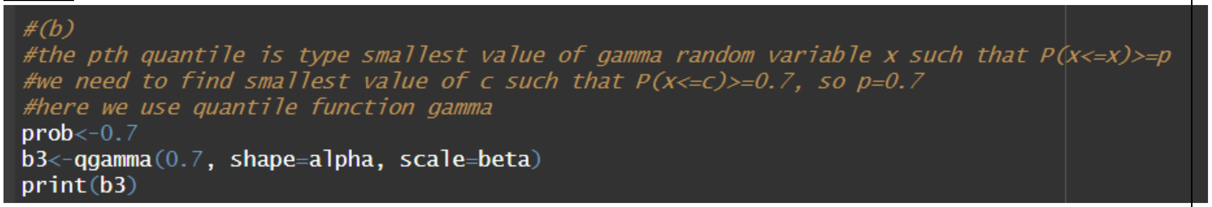
   Output:



```
> alpha<-2
> beta<-1/3
> a3_i<- dgamma(3,shape=alpha, scale=beta)
> print(a3_i)
[1] 0.003332065
> a3_ii<-pgamma(1,shape=alpha, scale=beta, lo
er.tail=FALSE)
> print(a3_ii)
[1] 0.1991483
```

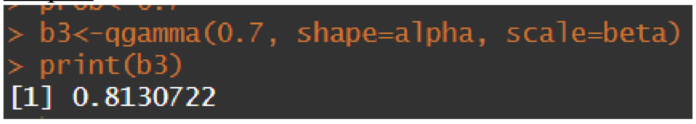
   b) **What is the value of c, if P(X ≤ c) ≥ 0.70? (Hint: try quantile function qgamma())**

   Code:



```
#(b)
#the pth quantile is type smallest value of gamma random variable x such that P(x<=x)>=p
#we need to find smallest value of c such that P(x<=c)>=0.7, so p=0.7
#here we use quantile function gamma
prob<-0.7
b3<-qgamma(0.7, shape=alpha, scale=beta)
print(b3)
```

   Output:



```
> prob<-0.7
> b3<-qgamma(0.7, shape=alpha, scale=beta)
> print(b3)
[1] 0.8130722
```

Probability and Statistics (UCS410)

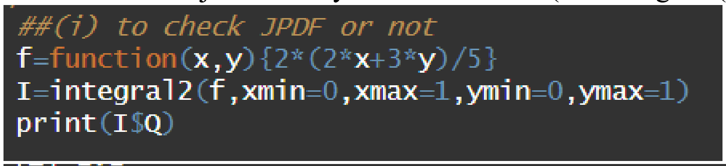
Experiment 6: Joint probability mass and density functions

(1) The joint probability density of two random variables X and Y is

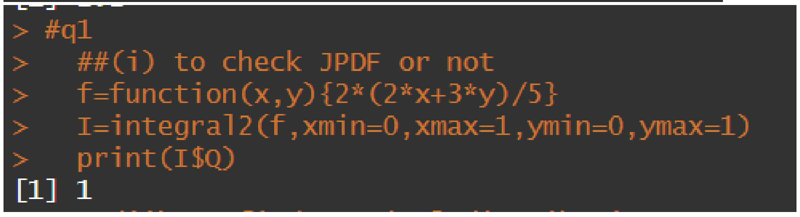
$$f(x,y) = \begin{cases} 2(2x+3y)/5; & 0 \le x, y \le 1 \\ 0; & elsewhere \end{cases}$$

Then write a R-code to

(i) check that it is a joint density function or not? (Use integral2())

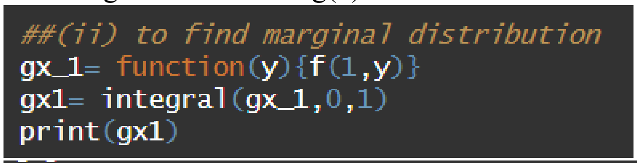


```
##(i) to check JPDF or not
f=function(x,y){2*(2*x+3*y)/5}
I=integral2(f,xmin=0,xmax=1,ymin=0,ymax=1)
print(I$Q)
```

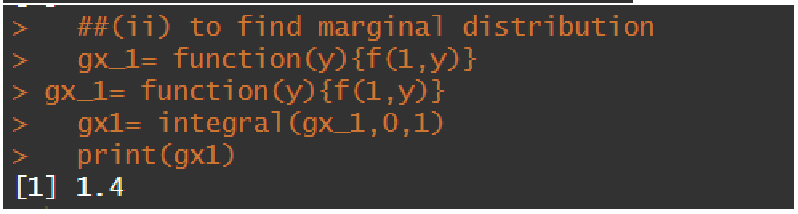


```
> #q1
>   ##(i) to check JPDF or not
>   f=function(x,y){2*(2*x+3*y)/5}
>   I=integral2(f,xmin=0,xmax=1,ymin=0,ymax=1)
>   print(I$Q)
[1] 1
```

(ii) find marginal distribution g(x) at x = 1.

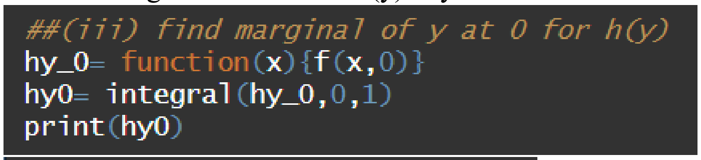


```
##(ii) to find marginal distribution
gx_1= function(y){f(1,y)}
gx1= integral(gx_1,0,1)
print(gx1)
```

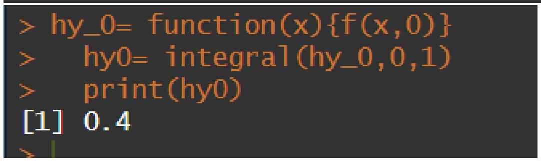


```
>   ##(ii) to find marginal distribution
>   gx_1= function(y){f(1,y)}
> gx_1= function(y){f(1,y)}
>   gx1= integral(gx_1,0,1)
>   print(gx1)
[1] 1.4
```

(iii) find the marginal distribution h(y) at y = 0.

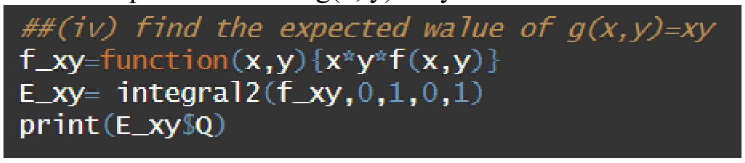


```
##(iii) find marginal of y at 0 for h(y)
hy_0= function(x){f(x,0)}
hy0= integral(hy_0,0,1)
print(hy0)
```

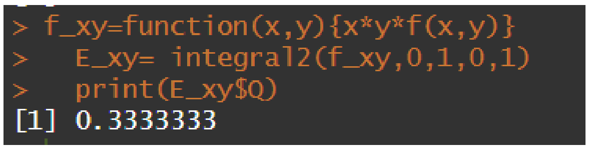


```
> hy_0= function(x){f(x,0)}
>   hy0= integral(hy_0,0,1)
>   print(hy0)
[1] 0.4
```

(iv) find the expected value of g(x, y) = xy.



```
##(iv) find the expected walue of g(x,y)=xy
f_xy=function(x,y){x*y*f(x,y)}
E_xy= integral2(f_xy,0,1,0,1)
print(E_xy$Q)
```



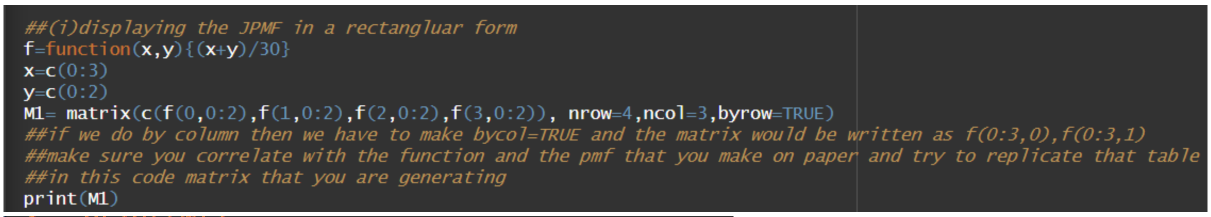
```
> f_xy=function(x,y){x*y*f(x,y)}
>   E_xy= integral2(f_xy,0,1,0,1)
>   print(E_xy$Q)
[1] 0.3333333
```

(2) The joint probability mass function of two random variables X and Y is

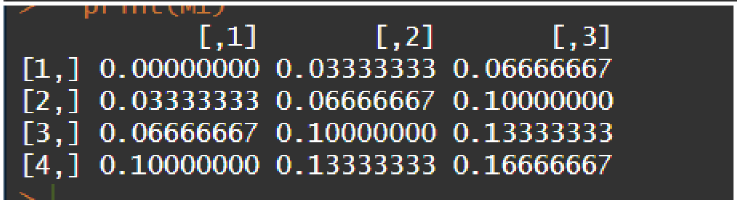
$$f(x, y) = \{(x + y)/30;\ \ x = 0, 1, 2, 3;\ \ y = 0, 1, 2\}$$

Then write a R-code to

(i) display the joint mass function in rectangular (matrix) form.

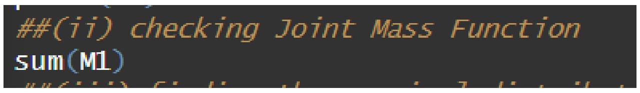


```
##(i)displaying the JPMF in a rectangluar form
f=function(x,y){(x+y)/30}
x=c(0:3)
y=c(0:2)
M1= matrix(c(f(0,0:2),f(1,0:2),f(2,0:2),f(3,0:2)), nrow=4,ncol=3,byrow=TRUE)
##if we do by column then we have to make bycol=TRUE and the matrix would be written as f(0:3,0),f(0:3,1)
##make sure you correlate with the function and the pmf that you make on paper and try to replicate that table
##in this code matrix that you are generating
print(M1)
```

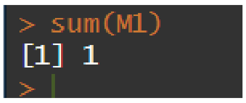


```
           [,1]       [,2]       [,3]
[1,] 0.00000000 0.03333333 0.06666667
[2,] 0.03333333 0.06666667 0.10000000
[3,] 0.06666667 0.10000000 0.13333333
[4,] 0.10000000 0.13333333 0.16666667
```

(ii) check that it is joint mass function or not? (use: Sum())

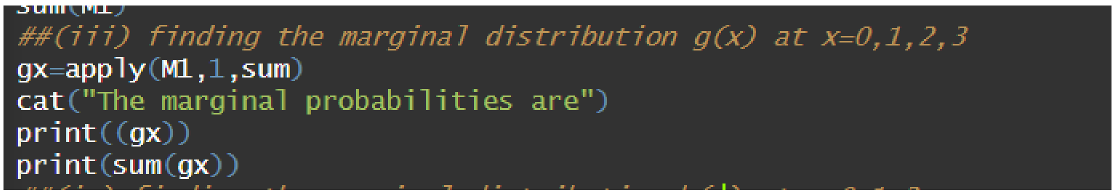


```
##(ii) checking Joint Mass Function
sum(M1)
```

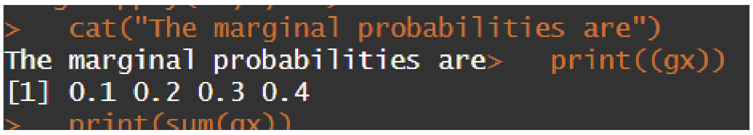


```
> sum(M1)
[1] 1
>
```

(iii) find the marginal distribution g(x) for x = 0, 1, 2, 3. (Use:apply())

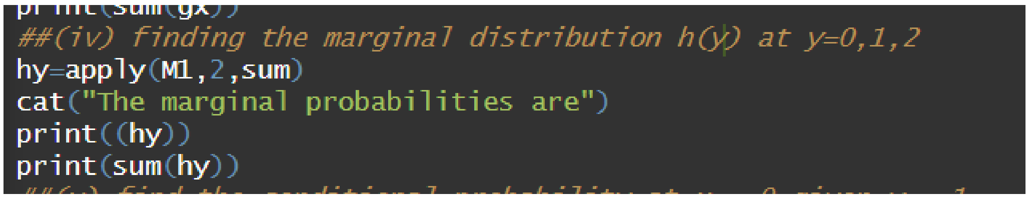


```
##(iii) finding the marginal distribution g(x) at x=0,1,2,3
gx=apply(M1,1,sum)
cat("The marginal probabilities are")
print((gx))
print(sum(gx))
```

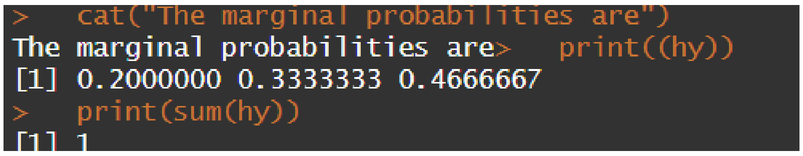


```
>   cat("The marginal probabilities are")
The marginal probabilities are>   print((gx))
[1] 0.1 0.2 0.3 0.4
>   print(sum(gx))
```

(iv) find the marginal distribution h(y) for y = 0, 1, 2. (Use:apply())

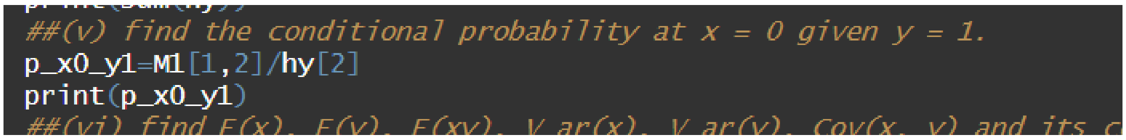


```
##(iv) finding the marginal distribution h(y) at y=0,1,2
hy=apply(M1,2,sum)
cat("The marginal probabilities are")
print((hy))
print(sum(hy))
```

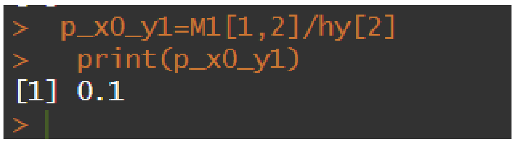


```
>   cat("The marginal probabilities are")
The marginal probabilities are>   print((hy))
[1] 0.2000000 0.3333333 0.4666667
>   print(sum(hy))
[1] 1
```

(v) find the conditional probability at x = 0 given y = 1.

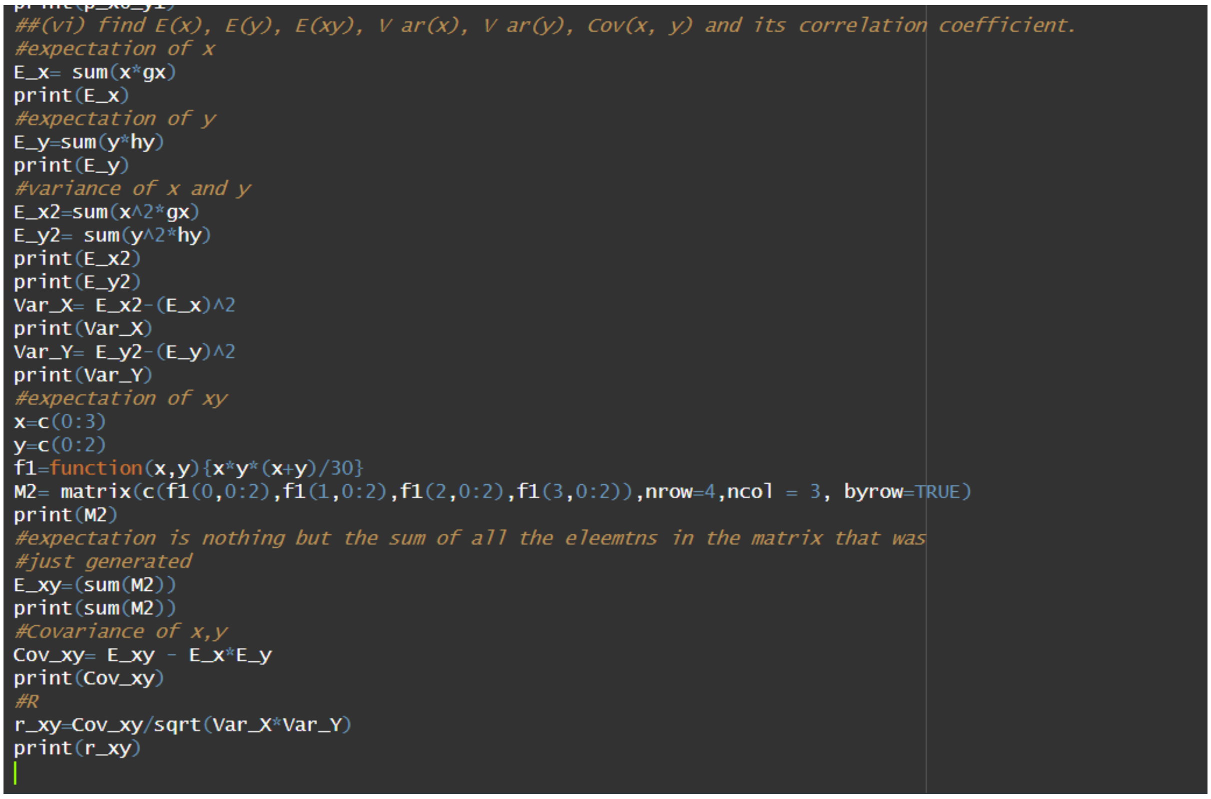


```
##(v) find the conditional probability at x = 0 given y = 1.
p_x0_y1=M1[1,2]/hy[2]
print(p_x0_y1)
##(vi) find E(x), E(y), E(xy), V ar(x), V ar(y), Cov(x, y) and its c
```

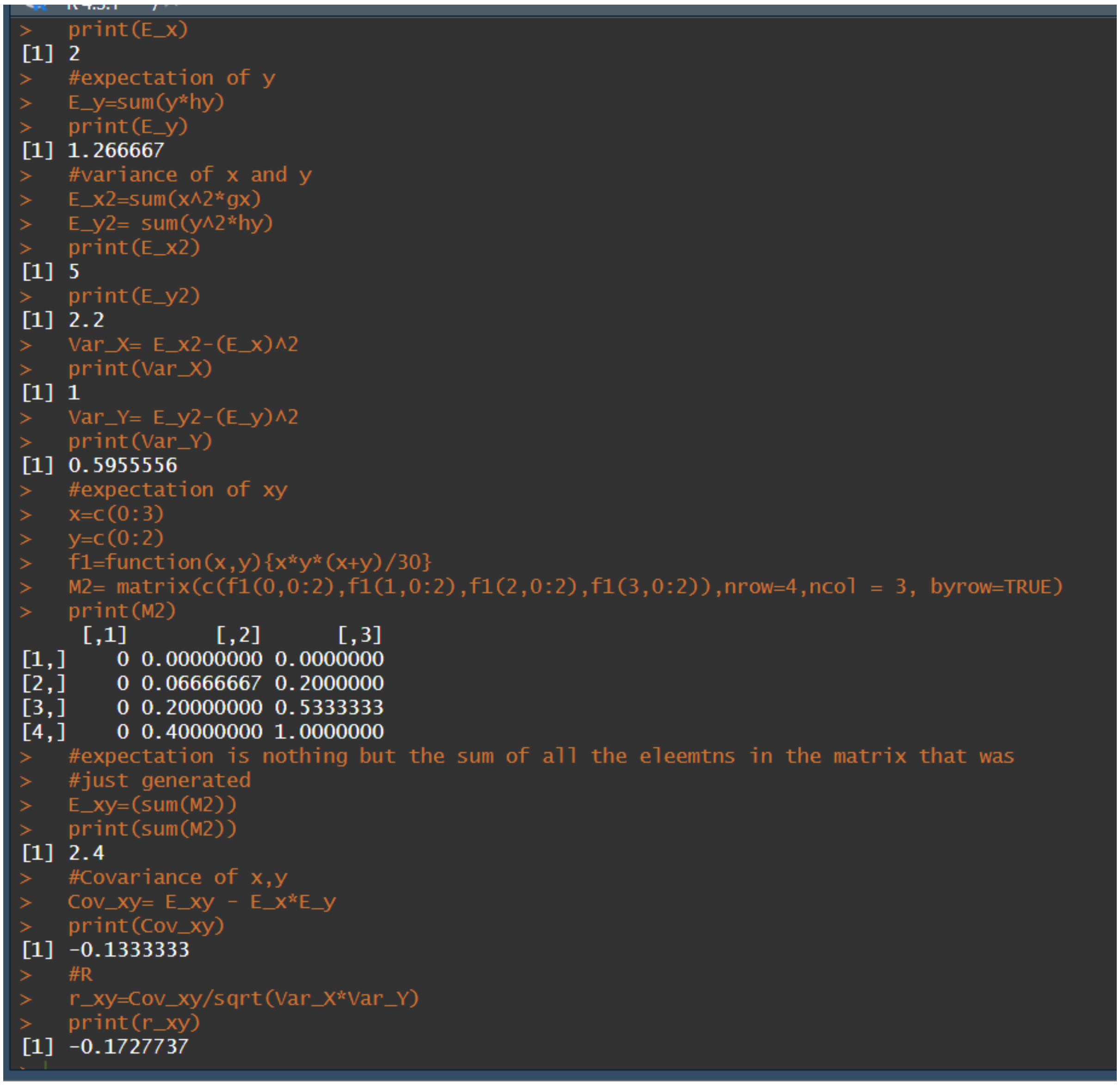


```
> p_x0_y1=M1[1,2]/hy[2]
>   print(p_x0_y1)
[1] 0.1
>
```

(vi) find E(x), E(y), E(xy), V ar(x), V ar(y), Cov(x, y) and its correlation coefficient.



```
##(vi) find E(x), E(y), E(xy), V ar(x), V ar(y), Cov(x, y) and its correlation coefficient.
#expectation of x
E_x= sum(x*gx)
print(E_x)
#expectation of y
E_y=sum(y*hy)
print(E_y)
#variance of x and y
E_x2=sum(x^2*gx)
E_y2= sum(y^2*hy)
print(E_x2)
print(E_y2)
Var_X= E_x2-(E_x)^2
print(Var_X)
Var_Y= E_y2-(E_y)^2
print(Var_Y)
#expectation of xy
x=c(0:3)
y=c(0:2)
f1=function(x,y){x*y*(x+y)/30}
M2= matrix(c(f1(0,0:2),f1(1,0:2),f1(2,0:2),f1(3,0:2)),nrow=4,ncol = 3, byrow=TRUE)
print(M2)
#expectation is nothing but the sum of all the eleemtns in the matrix that was
#just generated
E_xy=(sum(M2))
print(sum(M2))
#Covariance of x,y
Cov_xy= E_xy - E_x*E_y
print(Cov_xy)
#R
r_xy=Cov_xy/sqrt(Var_X*Var_Y)
print(r_xy)
```



```
>   print(E_x)
[1] 2
>   #expectation of y
>   E_y=sum(y*hy)
>   print(E_y)
[1] 1.266667
>   #variance of x and y
>   E_x2=sum(x^2*gx)
>   E_y2= sum(y^2*hy)
>   print(E_x2)
[1] 5
>   print(E_y2)
[1] 2.2
>   Var_X= E_x2-(E_x)^2
>   print(Var_X)
[1] 1
>   Var_Y= E_y2-(E_y)^2
>   print(Var_Y)
[1] 0.5955556
>   #expectation of xy
>   x=c(0:3)
>   y=c(0:2)
>   f1=function(x,y){x*y*(x+y)/30}
>   M2= matrix(c(f1(0,0:2),f1(1,0:2),f1(2,0:2),f1(3,0:2)),nrow=4,ncol = 3, byrow=TRUE)
>   print(M2)
     [,1]       [,2]      [,3]
[1,]    0 0.00000000 0.0000000
[2,]    0 0.06666667 0.2000000
[3,]    0 0.20000000 0.5333333
[4,]    0 0.40000000 1.0000000
>   #expectation is nothing but the sum of all the eleemtns in the matrix that was
>   #just generated
>   E_xy=(sum(M2))
>   print(sum(M2))
[1] 2.4
>   #Covariance of x,y
>   Cov_xy= E_xy - E_x*E_y
>   print(Cov_xy)
[1] -0.1333333
>   #R
>   r_xy=Cov_xy/sqrt(Var_X*Var_Y)
>   print(r_xy)
[1] -0.1727737
```

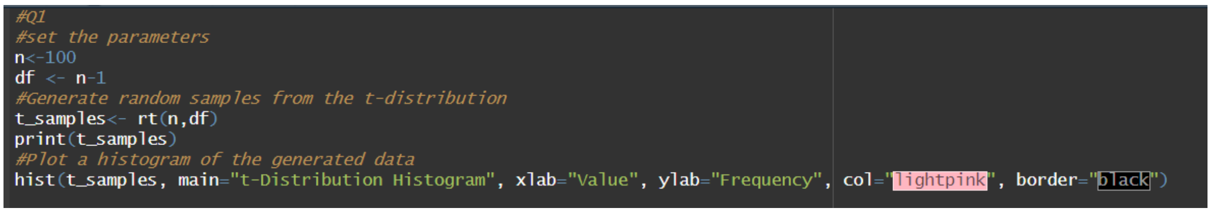
Probability and Statistics (UCS410)

Experiment 7: Chi-square, t-distribution, F-distribution

1) Use the rt(n, df) function in r to investigate the t-distribution for n = 100 and df = n − 1 and plot

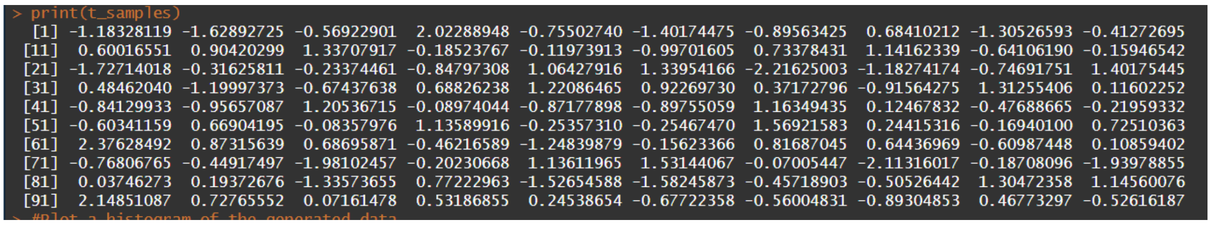
the histogram for the same.

CODE:



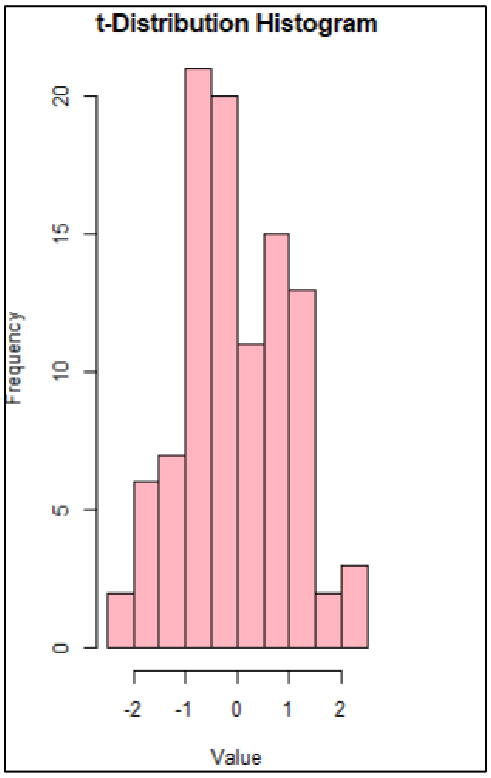
```
#Q1
#set the parameters
n<-100
df <- n-1
#Generate random samples from the t-distribution
t_samples<- rt(n,df)
print(t_samples)
#Plot a histogram of the generated data
hist(t_samples, main="t-Distribution Histogram", xlab="Value", ylab="Frequency", col="lightpink", border="black")
```

OUTPUT:



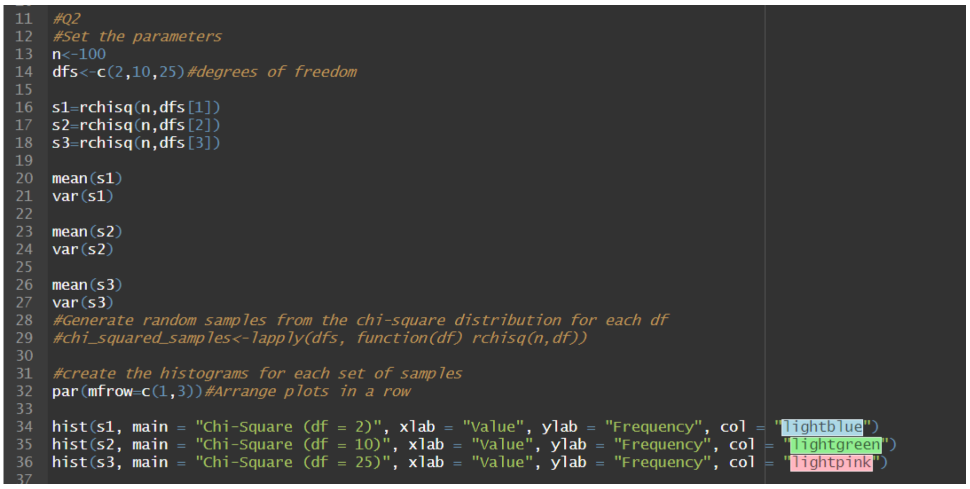
```
> print(t_samples)
  [1] -1.18328119 -1.62892725 -0.56922901  2.02288948 -0.75502740 -1.40174475 -0.89563425  0.68410212 -1.30526593 -0.41272695
 [11]  0.60016551  0.90420299  1.33707917 -0.18523767 -0.11973913 -0.99701605  0.73378431  1.14162339 -0.64106190 -0.15946542
 [21] -1.72714018 -0.31625811 -0.23374461 -0.84797308  1.06427916  1.33954166 -2.21625003 -1.18274174 -0.74691751  1.40175445
 [31]  0.48462040 -1.19997373 -0.67437638  0.68826238  1.22086465  0.92269730  0.37172796 -0.91564275  1.31255406  0.11602252
 [41] -0.84129933 -0.95657087  1.20536715 -0.08974044 -0.87177898 -0.89755059  1.16349435  0.12467832 -0.47688665 -0.21959332
 [51] -0.60341159  0.66904195 -0.08357976  1.13589916 -0.25357310 -0.25467470  1.56921583  0.24415316 -0.16940100  0.72510363
 [61]  2.37628492  0.87315639  0.68695871 -0.46216589 -1.24839879 -0.15623366  0.81687045  0.64436969 -0.60987448  0.10859402
 [71] -0.76806765 -0.44917497 -1.98102457 -0.20230668  1.13611965  1.53144067 -0.07005447 -2.11316017 -0.18708096 -1.93978855
 [81]  0.03746273  0.19372676 -1.33573655  0.77222963 -1.52654588 -1.58245873 -0.45718903 -0.50526442  1.30472358  1.14560076
 [91]  2.14851087  0.72765552  0.07161478  0.53186855  0.24538654 -0.67722358 -0.56004831 -0.89304853  0.46773297 -0.52616187
> #Plot a histogram of the generated data
```

PLOT:



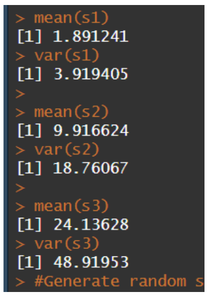
(2) Use the rchisq(n, df) function in r to investigate the chi-square distribution with n = 100 and df = 2, 10, 25.

CODE:



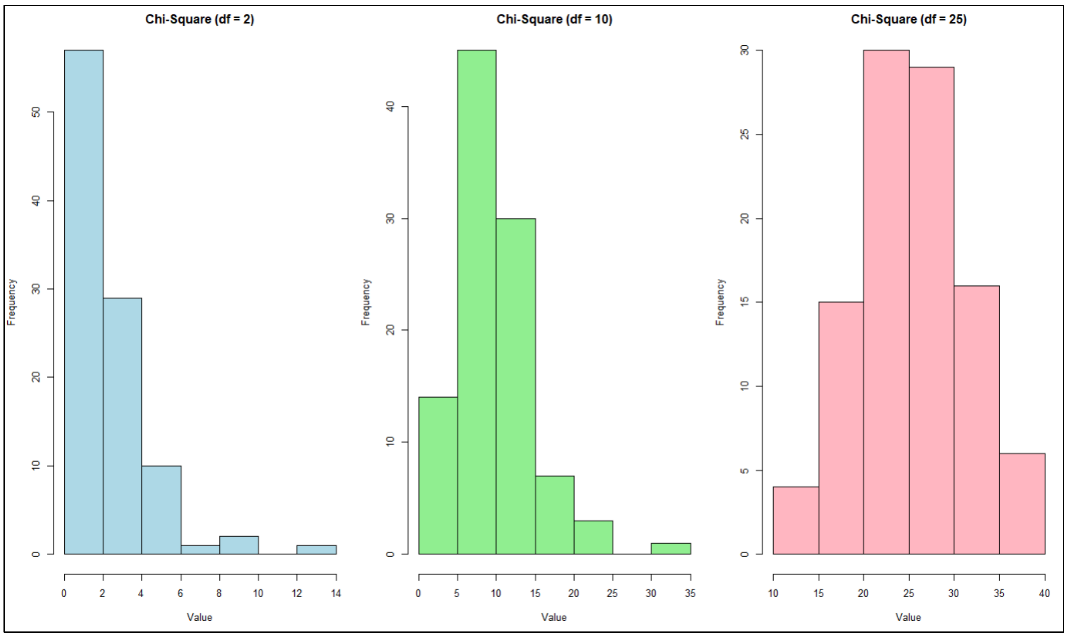
```
#Q2
#Set the parameters
n<-100
dfs<-c(2,10,25)#degrees of freedom

s1=rchisq(n,dfs[1])
s2=rchisq(n,dfs[2])
s3=rchisq(n,dfs[3])

mean(s1)
var(s1)

mean(s2)
var(s2)

mean(s3)
var(s3)
#Generate random samples from the chi-square distribution for each df
#chi_squared_samples<-lapply(dfs, function(df) rchisq(n,df))

#create the histograms for each set of samples
par(mfrow=c(1,3))#Arrange plots in a row

hist(s1, main = "Chi-Square (df = 2)", xlab = "Value", ylab = "Frequency", col = "lightblue")
hist(s2, main = "Chi-Square (df = 10)", xlab = "Value", ylab = "Frequency", col = "lightgreen")
hist(s3, main = "Chi-Square (df = 25)", xlab = "Value", ylab = "Frequency", col = "lightpink")
```

OUTPUT:



```
> mean(s1)
[1] 1.891241
> var(s1)
[1] 3.919405
>
> mean(s2)
[1] 9.916624
> var(s2)
[1] 18.76067
>
> mean(s3)
[1] 24.13628
> var(s3)
[1] 48.91953
> #Generate random s
```

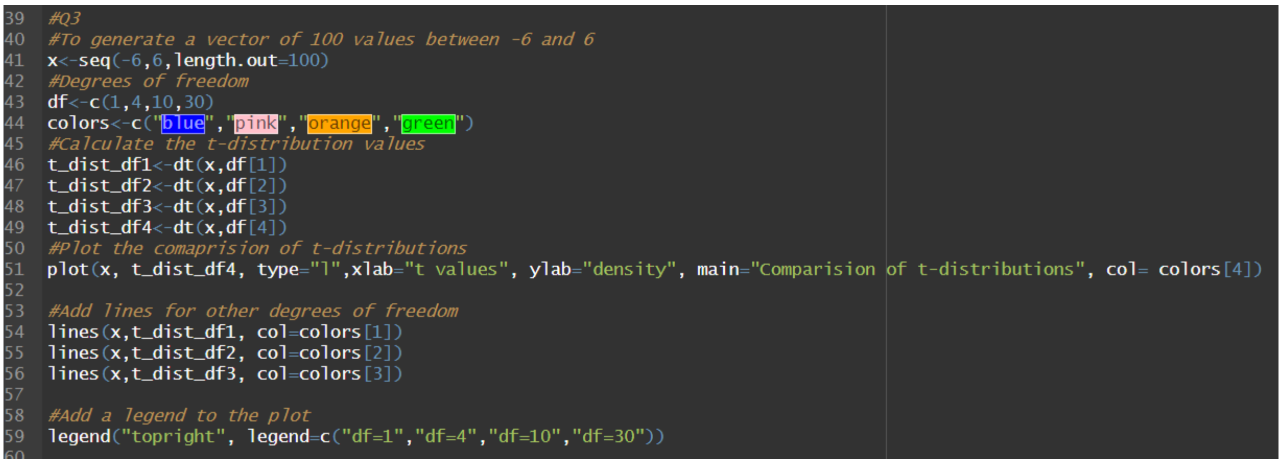
PLOTS:



(3) Generate a vector of 100 values between -6 and 6. Use the dt() function in r to find the values of a t-distribution given a random variable x and degrees of freedom 1,4,10,30. Using these values

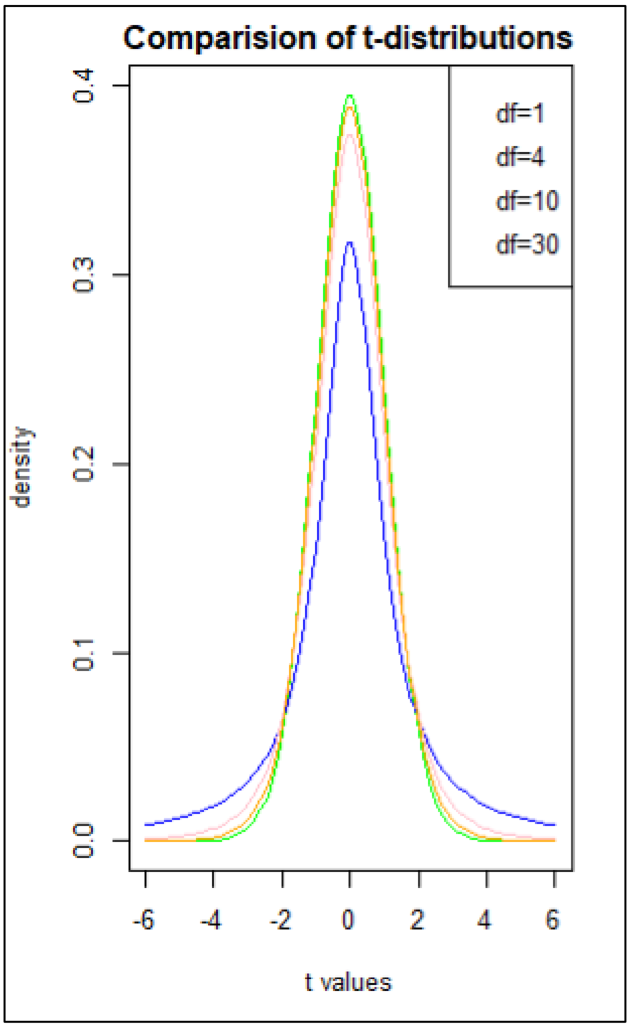
plot the density function for students t-distribution with degrees of freedom 30. Also shows a comparison of probability density functions having different degrees of freedom (1,4,10,30).

CODE:



```r
#Q3
#To generate a vector of 100 values between -6 and 6
x<-seq(-6,6,length.out=100)
#Degrees of freedom
df<-c(1,4,10,30)
colors<-c("blue","pink","orange","green")
#Calculate the t-distribution values
t_dist_df1<-dt(x,df[1])
t_dist_df2<-dt(x,df[2])
t_dist_df3<-dt(x,df[3])
t_dist_df4<-dt(x,df[4])
#Plot the comaprision of t-distributions
plot(x, t_dist_df4, type="l",xlab="t values", ylab="density", main="Comparision of t-distributions", col= colors[4])

#Add lines for other degrees of freedom
lines(x,t_dist_df1, col=colors[1])
lines(x,t_dist_df2, col=colors[2])
lines(x,t_dist_df3, col=colors[3])

#Add a legend to the plot
legend("topright", legend=c("df=1","df=4","df=10","df=30"))
```

OUTPUT/PLOT:



(4) Write a r-code

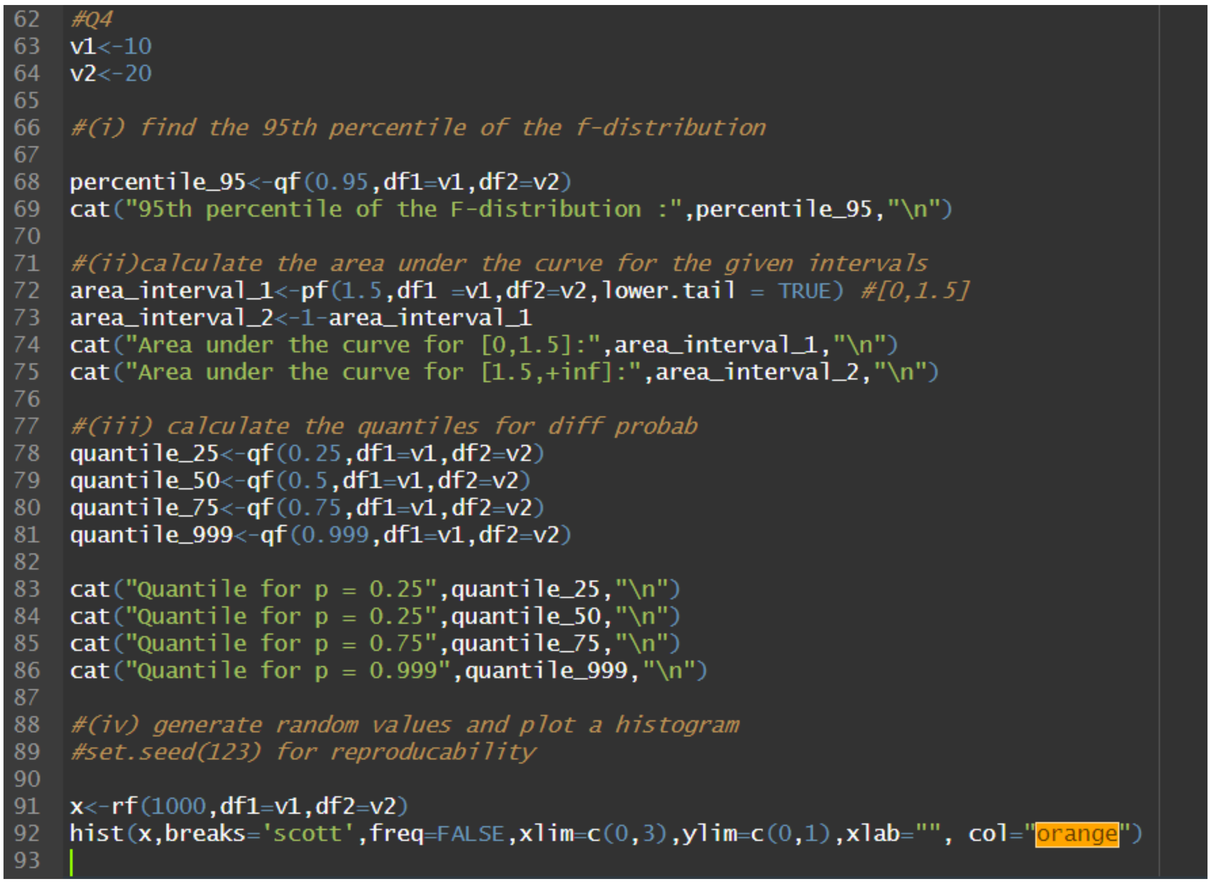
(i) To find the 95th percentile of the F-distribution with (10, 20) degrees of freedom.

(ii) To calculate the area under the curve for the interval [0, 1.5] and the interval [1.5, +∞) of a F-curve with v1 = 10 and v2 = 20 (USE pf()).

(iii) To calculate the quantile for a given area (= probability) under the curve for a F-curve with v1 = 10 and v2 = 20 that corresponds to q = 0.25, 0.5, 0.75 and 0.999. (use the qf())

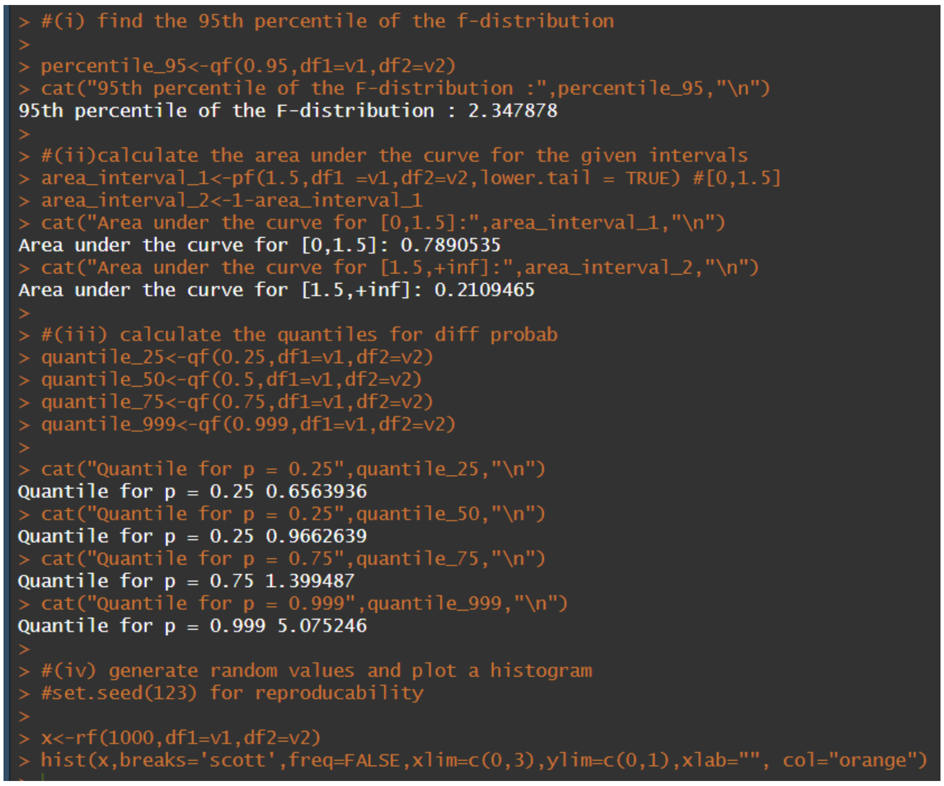
(iv) To generate 1000 random values from the F-distribution with v1 = 10 and v2 = 20 (use rf())and plot a histogram.

CODE:



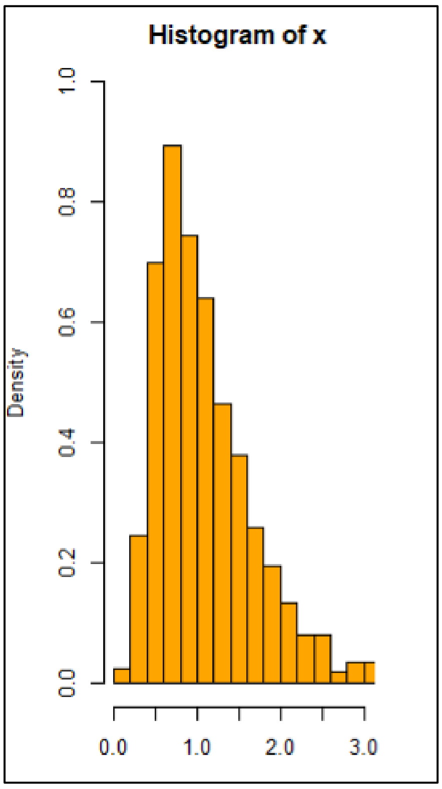
```
#Q4
v1<-10
v2<-20

#(i) find the 95th percentile of the f-distribution

percentile_95<-qf(0.95,df1=v1,df2=v2)
cat("95th percentile of the F-distribution :",percentile_95,"\n")

#(ii)calculate the area under the curve for the given intervals
area_interval_1<-pf(1.5,df1 =v1,df2=v2,lower.tail = TRUE) #[0,1.5]
area_interval_2<-1-area_interval_1
cat("Area under the curve for [0,1.5]:",area_interval_1,"\n")
cat("Area under the curve for [1.5,+inf]:",area_interval_2,"\n")

#(iii) calculate the quantiles for diff probab
quantile_25<-qf(0.25,df1=v1,df2=v2)
quantile_50<-qf(0.5,df1=v1,df2=v2)
quantile_75<-qf(0.75,df1=v1,df2=v2)
quantile_999<-qf(0.999,df1=v1,df2=v2)

cat("Quantile for p = 0.25",quantile_25,"\n")
cat("Quantile for p = 0.25",quantile_50,"\n")
cat("Quantile for p = 0.75",quantile_75,"\n")
cat("Quantile for p = 0.999",quantile_999,"\n")

#(iv) generate random values and plot a histogram
#set.seed(123) for reproducability

x<-rf(1000,df1=v1,df2=v2)
hist(x,breaks='scott',freq=FALSE,xlim=c(0,3),ylim=c(0,1),xlab="", col="orange")

```

OUTPUT:



```
> #(i) find the 95th percentile of the f-distribution
>
> percentile_95<-qf(0.95,df1=v1,df2=v2)
> cat("95th percentile of the F-distribution :",percentile_95,"\n")
95th percentile of the F-distribution : 2.347878
>
> #(ii)calculate the area under the curve for the given intervals
> area_interval_1<-pf(1.5,df1 =v1,df2=v2,lower.tail = TRUE) #[0,1.5]
> area_interval_2<-1-area_interval_1
> cat("Area under the curve for [0,1.5]:",area_interval_1,"\n")
Area under the curve for [0,1.5]: 0.7890535
> cat("Area under the curve for [1.5,+inf]:",area_interval_2,"\n")
Area under the curve for [1.5,+inf]: 0.2109465
>
> #(iii) calculate the quantiles for diff probab
> quantile_25<-qf(0.25,df1=v1,df2=v2)
> quantile_50<-qf(0.5,df1=v1,df2=v2)
> quantile_75<-qf(0.75,df1=v1,df2=v2)
> quantile_999<-qf(0.999,df1=v1,df2=v2)
>
> cat("Quantile for p = 0.25",quantile_25,"\n")
Quantile for p = 0.25 0.6563936
> cat("Quantile for p = 0.25",quantile_50,"\n")
Quantile for p = 0.25 0.9662639
> cat("Quantile for p = 0.75",quantile_75,"\n")
Quantile for p = 0.75 1.399487
> cat("Quantile for p = 0.999",quantile_999,"\n")
Quantile for p = 0.999 5.075246
>
> #(iv) generate random values and plot a histogram
> #set.seed(123) for reproducability
>
> x<-rf(1000,df1=v1,df2=v2)
> hist(x,breaks='scott',freq=FALSE,xlim=c(0,3),ylim=c(0,1),xlab="", col="orange")
```

PLOT:



Probability and Statistics (UCS410)

Experiment 8

A pipe manufacturing organization produces different kinds of pipes. We are given the monthly data of the wall thickness of certain types of pipes (data is available on LMS Clt-data.csv).

The organization has an analysis to perform and one of the basic assumptions of that analysis is that the data should be normally distributed.

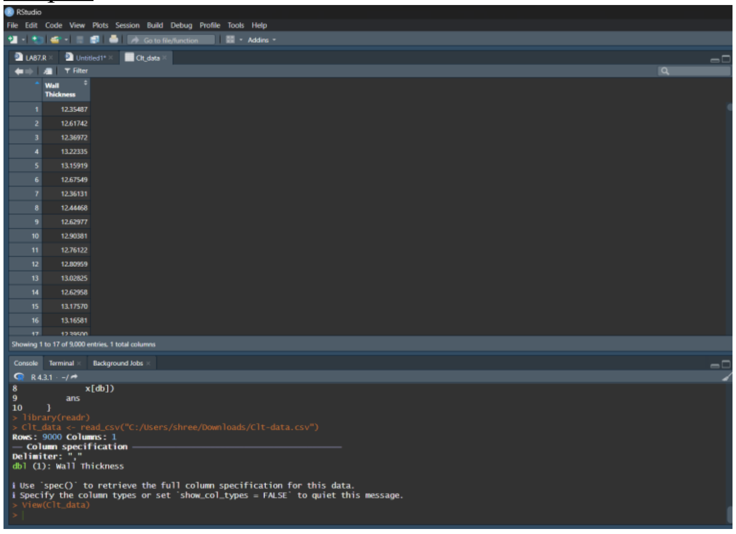
You have the following tasks to do:

(a) Import the csv data file in R.

Code:

```r
# (a) Import the csv data file in R
library(readr)
Clt_data <- read_csv("C:/Users/shree/Downloads/Clt-data.csv")
View(Clt_data)
```

Output:



	Wall Thickness
1	12.35487
2	12.61742
3	12.36972
4	13.22335
5	13.15919
6	12.67549
7	12.36131
8	12.44468
9	12.62977
10	12.90381
11	12.76122
12	12.80959
13	13.02825
14	12.62958
15	13.17570
16	13.16581
17	12.39500

Showing 1 to 17 of 9,000 entries, 1 total columns

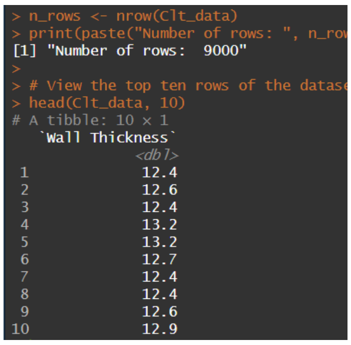
```
8             x[db])
9         ans
10     }
> library(readr)
> Clt_data <- read_csv("C:/Users/shree/Downloads/Clt-data.csv")
Rows: 9000 Columns: 1
── Column specification ──────────────────────────────
Delimiter: ","
dbl (1): Wall Thickness

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
> View(Clt_data)
>
```

(b) Validate data for correctness by counting number of rows and viewing the top ten rows of the dataset.

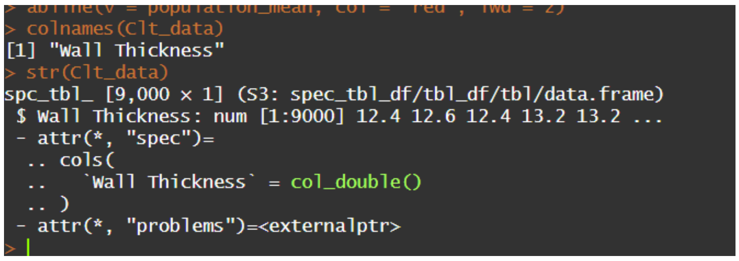
Code:

```r
# (b) Validate data for correctness
# Count number of rows
n_rows <- nrow(Clt_data)
print(paste("Number of rows: ", n_rows))
# View the top ten rows of the dataset
head(Clt_data, 10)
# Data about the column names
colnames(Clt_data)
str(Clt_data)
```

Output:



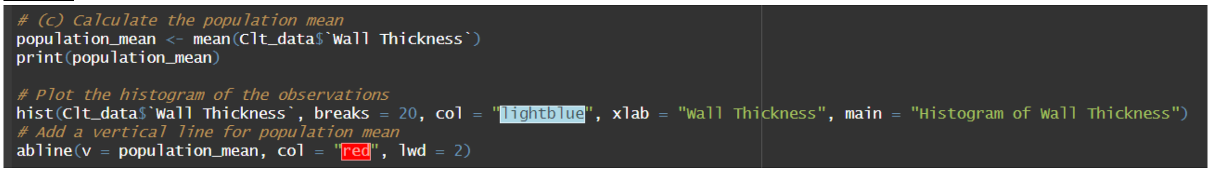
```
> n_rows <- nrow(Clt_data)
> print(paste("Number of rows: ", n_row
[1] "Number of rows:  9000"
>
> # View the top ten rows of the datase
> head(Clt_data, 10)
# A tibble: 10 × 1
  `Wall Thickness`
             <dbl>
 1            12.4
 2            12.6
 3            12.4
 4            13.2
 5            13.2
 6            12.7
 7            12.4
 8            12.4
 9            12.6
10            12.9
```



```
> abline(v = population_mean, col = "red", lwd = 2)
> colnames(Clt_data)
[1] "Wall Thickness"
> str(Clt_data)
spc_tbl_ [9,000 × 1] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ Wall Thickness: num [1:9000] 12.4 12.6 12.4 13.2 13.2 ...
 - attr(*, "spec")=
  .. cols(
  ..   `Wall Thickness` = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
> 
```

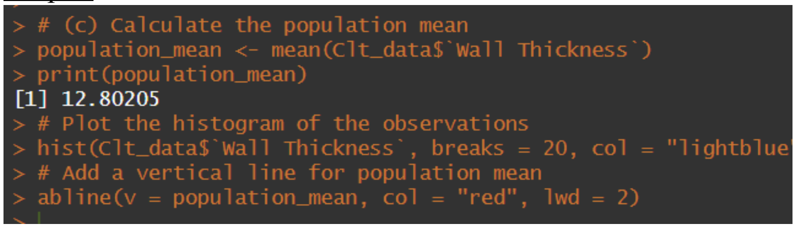
(c) Calculate the population mean and plot the observations by making a histogram.

Code:



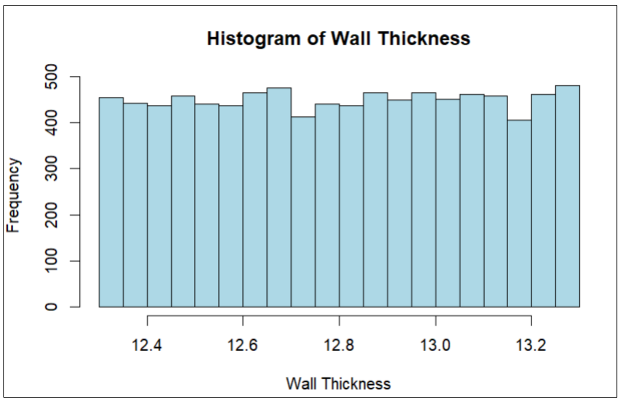
```
# (c) Calculate the population mean
population_mean <- mean(Clt_data$`Wall Thickness`)
print(population_mean)

# Plot the histogram of the observations
hist(Clt_data$`Wall Thickness`, breaks = 20, col = "lightblue", xlab = "Wall Thickness", main = "Histogram of Wall Thickness")
# Add a vertical line for population mean
abline(v = population_mean, col = "red", lwd = 2)
```

Output:



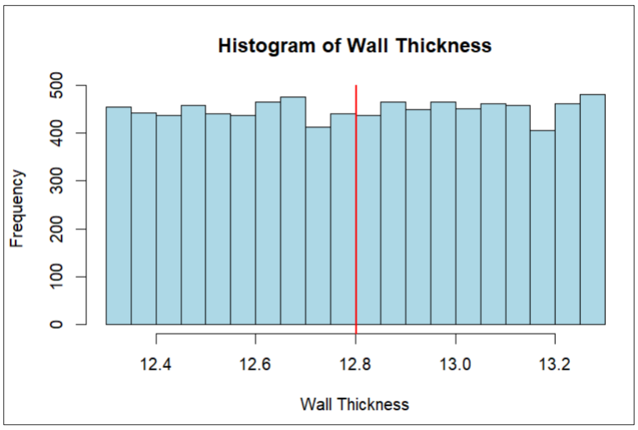
```
> # (c) Calculate the population mean
> population_mean <- mean(Clt_data$`Wall Thickness`)
> print(population_mean)
[1] 12.80205
> # Plot the histogram of the observations
> hist(Clt_data$`Wall Thickness`, breaks = 20, col = "lightblue"
> # Add a vertical line for population mean
> abline(v = population_mean, col = "red", lwd = 2)
> 
```

The Histogram:



(d) Mark the mean computed in last step by using the function abline.

Histogram with abline:



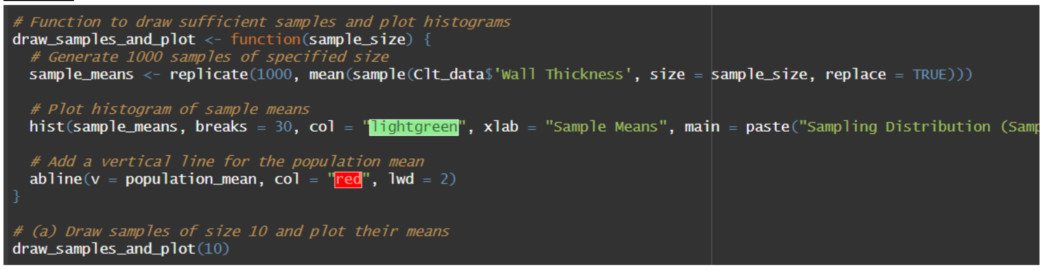
See the red vertical line in the histogram? That's the population mean. Comment on whether the data is normally distributed or not?

Ans: Although the **abline** is right in the middle of the histogram still it does not confirm it is a normal distribution. After studying the histogram, we can clearly say that the histogram does NOT resemble a BELL-SHAPED CURVE so we can say that the data is **NOT NORMALLY DISTRIBUTED**.

Now perform the following tasks:

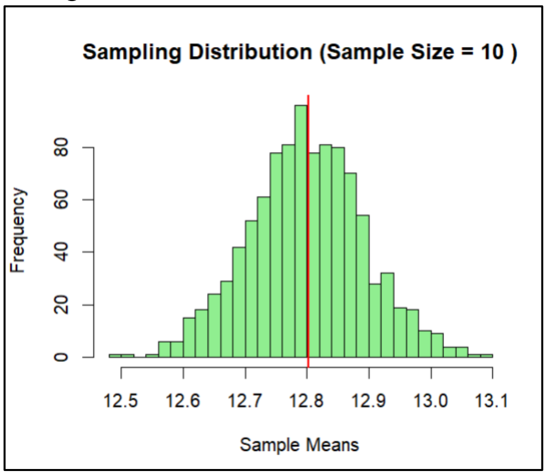
(a) Draw sufficient samples of size 10, calculate their means, and plot them in R by making histogram. Do you get a normal distribution.

Code:



```
# Function to draw sufficient samples and plot histograms
draw_samples_and_plot <- function(sample_size) {
  # Generate 1000 samples of specified size
  sample_means <- replicate(1000, mean(sample(Clt_data$'Wall Thickness', size = sample_size, replace = TRUE)))

  # Plot histogram of sample means
  hist(sample_means, breaks = 30, col = "lightgreen", xlab = "Sample Means", main = paste("Sampling Distribution (Samp

  # Add a vertical line for the population mean
  abline(v = population_mean, col = "red", lwd = 2)
}

# (a) Draw samples of size 10 and plot their means
draw_samples_and_plot(10)
```

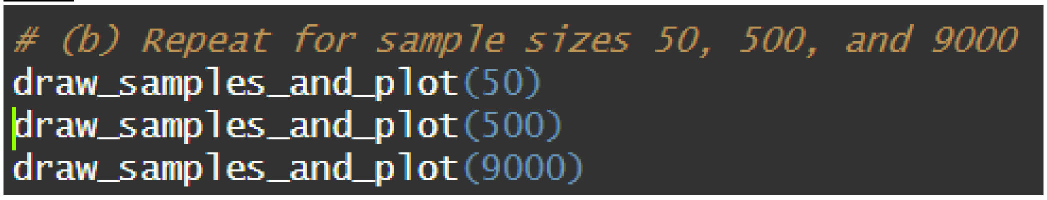
Histogram:



The mean is coming around 12.8. The histogram clearly represents a Bell-Shaped curve. So we can conclude that the sample is **NORMALLY DISTRIBUTED**.

(b) Now repeat the same with sample size 50, 500 and 9000. Can you comment on what you observe.

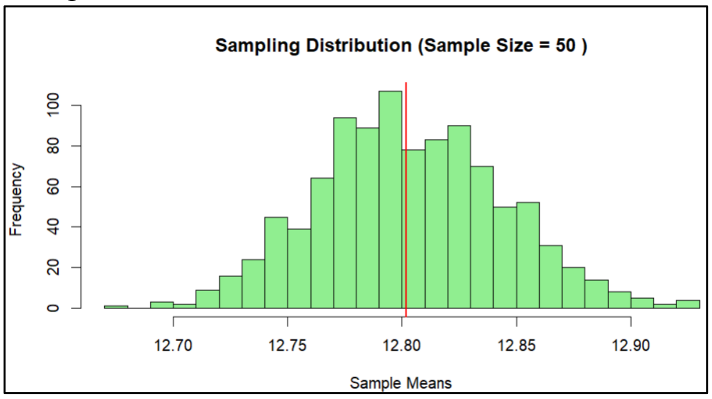
Code:



```
# (b) Repeat for sample sizes 50, 500, and 9000
draw_samples_and_plot(50)
draw_samples_and_plot(500)
draw_samples_and_plot(9000)
```

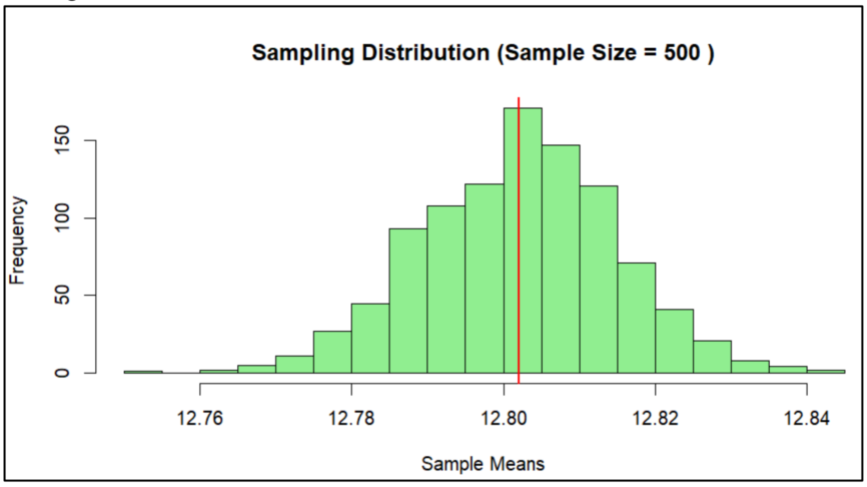
50:

Histogram:



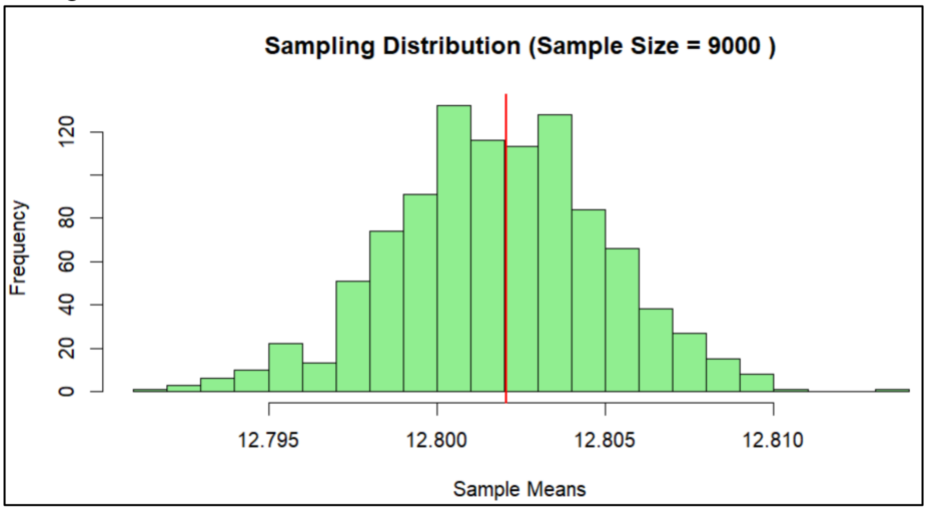
500:

Histogram:



9000:

Histogram:



Here, we get a good bell-shaped curve and the sampling distribution approaches normal distribution as the sample sizes increase. Therefore, we can recommend the organization to use sampling distributions of mean for further analysis.

ALL CODES:

EXPERIMENT 1

```
#(1)
v1<-c(5,10,15,20,25,30)
print(paste("Maximum Number is: ", max(v1)))
print(paste("Minimum Number is: ", min(v1)))

#(2)
factorial_result <- 1
n <- as.integer(readline(prompt = "Enter integer: "))

if (n <= 0) {
  print('Error')
} else {
  for (i in 1:n) {
    factorial_result <- factorial_result * i
  }
  print(paste("Factorial of", n, "is", factorial_result))
}

#(3)
n <- as.integer(readline("Enter the value of n: "))
if (n <= 2) {
  print('Error')
}else {
  fib <- numeric(n)
  fib[1] <- 0
  fib[2] <- 1

  for (i in 3:n) {
    fib[i] <- fib[i - 1] + fib[i - 2]
  }
  cat("Fibonacci sequence of", n, "terms:", fib)
}

#(4)
num1 <- as.numeric(readline("Enter the first number: "))
num2 <- as.numeric(readline("Enter the second number: "))

cat("Select operation:\n1. Add\n2. Subtract\n3. Multiply\n4. Divide\n")
choice <- as.integer(readline("Enter choice (1/2/3/4): "))

result <- switch(choice,
                 "1" = num1 + num2,
                 "2" = num1 - num2,
                 "3" = num1 * num2,
                 "4" = {
```

```
           if (num2 == 0) {
             stop("Error: Division by zero is not allowed.")
           }
           num1 / num2
         },
         stop("Error: Invalid choice."))
cat("Result:", result)

#(5)
# Load necessary library for plotting
install.packages("plotrix")
library(plotrix)

cities <- c("Kolkata", "Mumbai", "Delhi", "Chennai", "Patiala")
values <- c(5, 8, 30, 22, 55)

print(pie(values, labels = cities, main = "City Distribution"))

#BAR GRAPH
bar_colors <- c("red", "green", "blue", "orange", "purple")
barplot(values, names.arg = cities, main = "City Distribution")

#HISTOGRAM
data <- rnorm(100)
hist(data, main = "Histogram of Random Data", xlab = "Value", ylab = "Frequency", col = "blue")

#SCATTER PLOT
x <- rnorm(50)
y <- 2 * x + rnorm(50)

plot(x, y, main = "Scatter Plot", xlab = "X", ylab = "Y", col = "red", pch = 19)

#LINE PLOT
x <- seq(0, 2 * pi, length.out = 100)
y <- sin(x)

plot(x, y, type = "l", main = "Sine Function", xlab = "X", ylab = "Y", col = "green")

#BOX PLOT
data <- matrix(rnorm(200), ncol = 4)#random data
boxplot(data, main = "Box Plot of Random Data", col = c("red", "blue", "green", "purple"))
```

EXPERIMENT 2

```
##(1)
coins <- c(rep("gold", 20), rep("silver", 30), rep("bronze", 50))

sampleSpace <- sample(coins, size = 10, replace = FALSE)

print(sampleSpace)
```

```
##(2)
outcomes <- c("Success", "Failure")
probab <- c(0.9, 0.1)

# Generate a sample space for 10 surgical procedures
sample_space <- sample(outcomes, size = 10, replace = TRUE, prob = probab)

# Display
cat("Sample space for next 10 Procedures:\n", sample_space)

##(3)
###TAKING USER INPUT OF N
n <- as.integer(readline("Number of people in the room: "))

# Calculate probability
prob_no_shared <- 1
for (i in 1:n) {
  prob_no_shared <- prob_no_shared * (365 - i + 1) / 365
}

prob_shared <- 1 - prob_no_shared

cat("Probability that at least two people share a birthday in a room with", n, "people:", prob_shared,
"\n")

##USING AN ARRAY OF VALUES FOR N
num_simulations <- 10000

for (n in c(5, 10, 15, 20, 25)) {
  shared_birthday_count <- 0

  for (sim in 1:num_simulations) {
    birthdays <- sample(1:365, size = n, replace = TRUE)
    if (length(birthdays) != length(unique(birthdays))) {
      shared_birthday_count <- shared_birthday_count + 1
    }
  }

  prob_shared <- shared_birthday_count / num_simulations
  cat("Estimated probability of shared birthday with", n, "people:", prob_shared, "\n")
}

##SMALLEST VALUE OF n FOR WHICH THE PROBABILITY IS GREATER THAN 0.5
num_simulations <- 10000
n <- 1
while (TRUE) {
  shared_birthday_count <- 0

  for (sim in 1:num_simulations) {
    birthdays <- sample(1:365, size = n, replace = TRUE)
    if (length(birthdays) != length(unique(birthdays))) {
      shared_birthday_count <- shared_birthday_count + 1
    }
  }

  prob_shared <- shared_birthday_count / num_simulations
  if (prob_shared > 0.5) {
    break
  }

  n <- n + 1
}

cat("Smallest value of n for which the probability is greater than 0.5:", n, "\n")


###(3)
conditional_probability <- function(prob_a, prob_b_given_a, prob_b) {
  prob_a_given_b <- (prob_b_given_a * prob_a) / prob_b
  return(prob_a_given_b)
}

# Given probabilities
prob_cloudy <- 0.4
prob_rain <- 0.2
prob_clouds_given_rain <- 0.85

# Compute the probability of rain given that it's cloudy
prob_rain_given_cloudy <- conditional_probability(prob_rain, prob_clouds_given_rain, prob_cloudy)

cat("Probability of rain given that it's cloudy:", prob_rain_given_cloudy, "\n")


###(4)
#Loading the dataset
data(iris)

# (a) Print first few rows of the dataset
head(iris)

# (b) Find the structure of the dataset
str(iris)

# (c) Find the range of sepal length
range_sepal_length <- range(iris$Sepal.Length)
cat("Range of sepal length:", range_sepal_length, "\n")

# (d) Find the mean of sepal length
mean_sepal_length <- mean(iris$Sepal.Length)
cat("Mean of sepal length:", mean_sepal_length, "\n")
```

```r
# (e) Find the median of sepal length
median_sepal_length <- median(iris$Sepal.Length)
cat("Median of sepal length:", median_sepal_length, "\n")

# (f) Find the first and third quartiles and the interquartile range
quartiles_sepal_length <- quantile(iris$Sepal.Length, probs = c(0.25, 0.75))
iqr_sepal_length <- quartiles_sepal_length[2] - quartiles_sepal_length[1]
cat("First Quartile:", quartiles_sepal_length[1], "\n")
cat("Third Quartile:", quartiles_sepal_length[2], "\n")
cat("Interquartile Range:", iqr_sepal_length, "\n")

# (g) Find the standard deviation and variance of sepal length
sd_sepal_length <- sd(iris$Sepal.Length)
var_sepal_length <- var(iris$Sepal.Length)
cat("Standard Deviation of sepal length:", sd_sepal_length, "\n")
cat("Variance of sepal length:", var_sepal_length, "\n")

# (h) Perform the above exercises for other attributes (sepal.width, petal.length, petal.width)
##SEPAL.WIDTH
#Range
range_sepal_width <- range(iris$Sepal.Width)
cat("Range of sepal width:", range_sepal_width, "\n")
#Mean
mean_sepal_width <- mean(iris$Sepal.Width)
cat("Mean of sepal width:", mean_sepal_width, "\n")
#Median
median_sepal_width <- median(iris$Sepal.Width)
cat("Median of sepal width:", median_sepal_width, "\n")
#Quartiles
quartiles_sepal_width <- quantile(iris$Sepal.Width, probs = c(0.25, 0.75))
iqr_sepal_width <- quartiles_sepal_width[2] - quartiles_sepal_width[1]
cat("First Quartile of sepal width:", quartiles_sepal_width[1], "\n")
cat("Third Quartile of sepal width:", quartiles_sepal_width[2], "\n")
cat("Interquartile Range of sepal width:", iqr_sepal_width, "\n")
##Standard Deviation and Variance
sd_sepal_width <- sd(iris$Sepal.Width)
var_sepal_width <- var(iris$Sepal.Width)
cat("Standard Deviation of sepal width:", sd_sepal_width, "\n")
cat("Variance of sepal width:", var_sepal_width, "\n")

##PETAL.WIDTH
#Range
range_petal_width <- range(iris$Petal.Width)
cat("Range of petal width:", range_petal_width, "\n")
#Mean
mean_petal_width <- mean(iris$Petal.Width)
cat("Mean of petal width:", mean_petal_width, "\n")
#Median
median_petal_width <- median(iris$Petal.Width)
cat("Median of petal width:", median_petal_width, "\n")
#Quartiles
```

```
quartiles_petal_width <- quantile(iris$Petal.Width, probs = c(0.25, 0.75))
iqr_petal_width <- quartiles_petal_width[2] - quartiles_petal_width[1]
cat("First Quartile of petal width:", quartiles_petal_width[1], "\n")
cat("Third Quartile of petal width:", quartiles_petal_width[2], "\n")
cat("Interquartile Range of petal width:", iqr_petal_width, "\n")
##Standard Deviation and Variance
sd_petal_width <- sd(iris$Petal.Width)
var_petal_width <- var(iris$Petal.Width)
cat("Standard Deviation of sepal width:", sd_petal_width, "\n")
cat("Variance of sepal width:", var_petal_width, "\n")

##PETAL.LENGTH
#Range
range_petal_length <- range(iris$Petal.Length)
cat("Range of petal length:", range_petal_length, "\n")
#Mean
mean_petal_length <- mean(iris$Petal.Length)
cat("Mean of petal length:", mean_petal_length, "\n")
#Median
median_petal_length <- median(iris$Petal.Length)
cat("Median of petal length:", median_petal_length, "\n")
#Quartiles
quartiles_petal_length <- quantile(iris$Petal.Length, probs = c(0.25, 0.75))
iqr_petal_length <- quartiles_petal_length[2] - quartiles_petal_length[1]
cat("First Quartile of petal length:", quartiles_petal_length[1], "\n")
cat("Third Quartile of petal length:", quartiles_petal_length[2], "\n")
cat("Interquartile Range of petal length:", iqr_petal_length, "\n")
##Standard Deviation and Variance
sd_petal_length <- sd(iris$Petal.Length)
var_petal_length <- var(iris$Petal.Length)
cat("Standard Deviation of petal length:", sd_petal_length, "\n")
cat("Variance of petal length:", var_petal_length, "\n")


# (i) Use the built-in function summary on the dataset Iris
summary(iris)

#R does not have a standard in-built function to calculate mode.
#So we create a user function to calculate mode of a data set in R.
#This function n takes the vector as input and gives the mode value as output.
calculate_mode <- function(data) {
  table_data <- table(data)
  mode <- as.numeric(names(table_data[table_data == max(table_data)]))
  return(mode)
}

# Test the function
dataset <- c(1,0,2,1,0,3,4,4,7)
mode_value <- calculate_mode(dataset)
cat("Mode of the dataset:", mode_value, "\n")
```

EXPERIMENT 3

```
#Q1.
# Number of dice rolls
n <- 12
# Probability of getting a 6 on one roll
p_success <- 1/6

# Calculate the cumulative probabilities using binomial distribution
cum_prob_6 <- pbinom(6, size = n, prob = p_success)
cum_prob_9 <- pbinom(9, size = n, prob = p_success)

# Calculate the probability of getting 7, 8, or 9 sixes
prob_7_to_9 <- cum_prob_9 - cum_prob_6

cat("Probability of getting 7, 8, or 9 sixes:", prob_7_to_9, "\n")

###USING DBINOM
# Number of dice rolls
n <- 12
# Probability of getting a 6 on one roll
p_success <- 1/6

# Calculate the probabilities using binomial distribution (PDF)
prob_7 <- dbinom(7, size = n, prob = p_success)
prob_8 <- dbinom(8, size = n, prob = p_success)
prob_9 <- dbinom(9, size = n, prob = p_success)

# Calculate the probability of getting 7, 8, or 9 sixes
prob_7_to_9 <- prob_7+prob_8+prob_9

cat("Probability of getting 7, 8, or 9 sixes:", prob_7_to_9, "\n")

#Q2
# Mean and standard deviation
mean_score <- 72
standard_deviation <- 15.2

# Score threshold
threshold <- 84

# cumulative distribution function (CDF)
probability_above_threshold <- 1 - pnorm(threshold, mean = mean_score, sd = standard_deviation)

# probability to percentage
percentage_above_threshold <- probability_above_threshold * 100

cat("Percentage of students scoring 84 or more:", percentage_above_threshold, "%\n")

#Q3.
# library for Poisson distribution calculations
```

```
library(stats)

# Parameters for the Poisson distribution
lambda_x <- 5  # Average number of cars from 10AM to 11AM
lambda_y <- 50 # Average number of customers from 8AM to 6PM

# Probability that no car arrives during 10AM to 11AM
prob_x <- dpois(0, lambda = lambda_x)

# Probability of having between 48 and 50 customers (inclusive) from 8AM to 6PM
prob_y <- sum(dpois(48:50, lambda = lambda_y))

cat("Probability that no car arrives during 10AM to 11AM:", prob_x, "\n")
cat("Probability of having between 48 and 50 customers from 8AM to 6PM:", prob_y, "\n")

#Q4
# Parameters for hypergeometric distribution
total_processors <- 250
defective_processors <- 17
sample_size <- 5

# p(x=3)
prob_3 <- dhyper(3, m = defective_processors, n = total_processors - defective_processors, k =
sample_size)

cat("Probability of exactly 3 defective processors in the sample:", prob_3, "\n")

#Q5
# Given probability
p_success <- 0.447

# Sample size
n <- 31

# (a) X is distributed as a binomial distribution
# (b) Sketch the probability mass function (PMF)
xx <- seq(0,31,1)
pmf_values <- numeric()
cdf_values <- numeric()

for(i in 1:length(xx))
{
 pmf_values[i] = dbinom(xx[i],n,p_success)
}

plot(xx,pmf_values)
# (c) Sketch the cumulative distribution function (CDF)

 for(i in 1:length(xx))
 {
  cdf_values[i] = pbinom(xx[i],n,p_success)
  }
# (d) Mean, variance, and standard deviation
mean_x <- n * p_success
variance_x <- n * p_success * (1 - p_success)
std_dev_x <- sqrt(variance_x)

# Print the results
cat("Mean of X:", mean_x, "\n")
cat("Variance of X:", variance_x, "\n")
cat("Standard Deviation of X:", std_dev_x, "\n")

# Plot PMF and CDF
plot(xx, pmf_values, xlab = "Number of Students (X)", ylab = "Probability", main = "Probability
Mass Function (PMF) of X")
plot(xx, cdf_values, xlab = "Number of Students (X)", ylab = "Cumulative Probability", main =
"Cumulative Distribution Function (CDF) of X")
```

EXPERIMENT 4

```
#Q1:

#Using weighted.mean()
x <- c(0, 1, 2, 3, 4)
p_x <- c(0.41, 0.37, 0.16, 0.05, 0.01)
mean_imperf <- weighted.mean(x, p_x)
cat("The average number of imperfections per 10 meters of fabric is:", mean_imperf)

#Using sum()
x <- c(0, 1, 2, 3, 4)
p_x <- c(0.41, 0.37, 0.16, 0.05, 0.01)
mean_imperf <- sum(x * p_x)
cat("The average number of imperfections per 10 meters of fabric is:", mean_imperf)



#Q2.
pdf <- function(t){
  t*0.1*exp(-0.1*t)
}

expected_value <- integrate(pdf, lower = 0, upper = Inf)$value
cat("The expected value of T is:", expected_value)

#Q3.
#Y = (12X+(3-X)2 - (6*3)) = (10X-12)
x<-c(0,1,2,3)
probab<-c(0.1,0.2,0.2,0.5)
print(weighted.mean(x,probab))
expval<-(10*weighted.mean(x,probab))-12
print(expval)
```

```
cat("The expected value of Y (net revenue) is:", expval)

#or

x<-c(0,1,2,3)
probabx<-c(0.1,0.2,0.2,0.5)
y<-10*x-12
probaby<-probabx
expval<-sum(y*probaby)
cat("The expected value of Y (net revenue) is:", expval)


#Q4.

f1<-function(x){
  x*0.5*exp(-abs(x))
}

f2<-function(x){
  x^2*0.5*exp(-abs(x))
}

moment1<-integrate(f1,1,10)
moment2<-integrate(f2,1,10)

print(moment1$value)
print(moment2$value)

meanval<-moment1$value
print(meanval)

f3<-function(m1,m2){
  return (m2-(m1^2))
}
print(meanval)

varval<-f3(moment1$value,moment2$value)
print(varval)



#Q5.
yf<-function(y){
  (3/4)*(1/4)^(sqrt(y)-1)
}
x<-as.integer(readline(prompt="Enter the value of x"))
y=x^2
proby<-yf(y)
print(proby)
```

```
x<-c(1,2,3,4,5)
y<-x^2
proby<-yf(y)
print(proby)

expval<-sum(y*proby)
print(expval)

m<-expval
y1<-(y-m)^2
proby1<-yf(y1)
print(proby1)
varval<-sum(y1*proby1)
print(varval)
```

EXPERIMENT 5

```
#q1
punif(45, min = 0, max = 69, lower.tail = FALSE)

1-punif(45, min-0, max=60)
punif(15, min = 0, max = 60)


#(b) Waiting time between 20 and 30 min
#F(30)-F(20)
#P(x<=30)-P(x<=20)
punif(30, min=0, max=60) - punif(20, min = 0, max = 60)

#2
#(a)
dexp(3, rate = 1/2)

#(b)
x<- seq(0,5, by=0.02)
px<-dexp(x,rate=1/2)
plot(x,px,xlab="x", ylab="f(x)", main="pdf of exponential distribution at lambda=1/2")

#(c)
#F(3)
#P(X<=3)
c2= pexp(3,rate=0.5)
print(c2)

#(d)
Fx<-pexp(x,rate=1/2)
plot(x,Fx,xlab="x", ylab="F(x)", main="cdf of exponential distribution at lambda=1/2")

#(e)
n<-1000
```

```
x_sim<-rexp(n,rate=1/2)
#rexp is used to generate random sample of exp dist
plot(density(x_sim), xlab="simulated x", ylab="density", main="simulated data for exp dist at
lambda=1/2")
hist(x_sim, probability=TRUE, xlab="simulated x", ylab="density", main="simulated data for exp
dist at lambda=1/2")


#Q3
#(a)
alpha<-2
beta<-1/3
a3_i<- dgamma(3,shape=alpha, scale=beta)
print(a3_i)
a3_ii<-pgamma(1,shape=alpha, scale=beta, lower.tail=FALSE)
print(a3_ii)

#(b)
#the pth quantile is type smallest value of gamma random variable x such that P(x<=x)>=p
#we need to find smallest value of c such that P(x<=c)>=0.7, so p=0.7
#here we use quantile function gamma
prob<-0.7
b3<-qgamma(0.7, shape=alpha, scale=beta)
print(b3)
```

EXPERIMENT 6

```
library('pracma')
install.packages('pracma')

#q1
 ##(i) to check JPDF or not
 f=function(x,y){2*(2*x+3*y)/5}
 I=integral2(f,xmin=0,xmax=1,ymin=0,ymax=1)
 print(I$Q)

 ##(ii) to find marginal distribution
 gx_1= function(y){f(1,y)}
 gx1= integral(gx_1,0,1)
 print(gx1)

 ##(iii) find marginal of y at 0 for h(y)
 hy_0= function(x){f(x,0)}
 hy0= integral(hy_0,0,1)
 print(hy0)

 ##(iv) find the expected walue of g(x,y)=xy
 f_xy=function(x,y){x*y*f(x,y)}
 E_xy= integral2(f_xy,0,1,0,1)
 print(E_xy$Q)
```

```
#Q2 JPMF is given

  ##(i)displaying the JPMF in a rectangluar form
  f=function(x,y){(x+y)/30}
  x=c(0:3)
  y=c(0:2)
  M1= matrix(c(f(0,0:2),f(1,0:2),f(2,0:2),f(3,0:2)), nrow=4,ncol=3,byrow=TRUE)
  ##if we do by column then we have to make bycol=TRUE and the matrix would be written as
f(0:3,0),f(0:3,1)
  ##make sure you correlate with the function and the pmf that you make on paper and try to replicate
that table
  ##in this code matrix that you are generating
  print(M1)
  ##(ii) checking Joint Mass Function
  sum(M1)
  ##(iii) finding the marginal distribution g(x) at x=0,1,2,3
  gx=apply(M1,1,sum)
  cat("The marginal probabilities are")
  print((gx))
  print(sum(gx))
  ##(iv) finding the marginal distribution h(y) at y=0,1,2
  hy=apply(M1,2,sum)
  cat("The marginal probabilities are")
  print((hy))
  print(sum(hy))
  ##(v) find the conditional probability at x = 0 given y = 1.
  p_x0_y1=M1[1,2]/hy[2]
  print(p_x0_y1)
  ##(vi) find E(x), E(y), E(xy), V ar(x), V ar(y), Cov(x, y) and its correlation coefficient.
  #expectation of x
  E_x= sum(x*gx)
  print(E_x)
  #expectation of y
  E_y=sum(y*hy)
  print(E_y)
  #variance of x and y
  E_x2=sum(x^2*gx)
  E_y2= sum(y^2*hy)
  print(E_x2)
  print(E_y2)
  Var_X= E_x2-(E_x)^2
  print(Var_X)
  Var_Y= E_y2-(E_y)^2
  print(Var_Y)
  #expectation of xy
  x=c(0:3)
  y=c(0:2)
  f1=function(x,y){x*y*(x+y)/30}
  M2= matrix(c(f1(0,0:2),f1(1,0:2),f1(2,0:2),f1(3,0:2)),nrow=4,ncol = 3, byrow=TRUE)
  print(M2)
```

```
#expectation is nothing but the sum of all the eleemtns in the matrix that was
#just generated
E_xy=(sum(M2))
print(sum(M2))
#Covariance of x,y
Cov_xy= E_xy - E_x*E_y
print(Cov_xy)
#R
r_xy=Cov_xy/sqrt(Var_X*Var_Y)
print(r_xy)
```

EXPERIMENT 7

```
#Q1
#set the parameters
n<-100
df <- n-1
#Generate random samples from the t-distribution
t_samples<- rt(n,df)
print(t_samples)
#Plot a histogram of the generated data
hist(t_samples, main="t-Distribution Histogram", xlab="Value", ylab="Frequency", col="lightpink",
border="black")

#Q2
#Set the parameters
n<-100
dfs<-c(2,10,25)#degrees of freedom

s1=rchisq(n,dfs[1])
s2=rchisq(n,dfs[2])
s3=rchisq(n,dfs[3])

mean(s1)
var(s1)

mean(s2)
var(s2)

mean(s3)
var(s3)
#Generate random samples from the chi-square distribution for each df
#chi_squared_samples<-lapply(dfs, function(df) rchisq(n,df))

#create the histograms for each set of samples
par(mfrow=c(1,3))#Arrange plots in a row

hist(s1, main = "Chi-Square (df = 2)", xlab = "Value", ylab = "Frequency", col = "lightblue")
hist(s2, main = "Chi-Square (df = 10)", xlab = "Value", ylab = "Frequency", col = "lightgreen")
hist(s3, main = "Chi-Square (df = 25)", xlab = "Value", ylab = "Frequency", col = "lightpink")


#Q3
#To generate a vector of 100 values between -6 and 6
x<-seq(-6,6,length.out=100)
#Degrees of freedom
df<-c(1,4,10,30)
colors<-c("blue","pink","orange","green")
#Calculate the t-distribution values
t_dist_df1<-dt(x,df[1])
t_dist_df2<-dt(x,df[2])
t_dist_df3<-dt(x,df[3])
t_dist_df4<-dt(x,df[4])
#Plot the comaprision of t-distributions
plot(x, t_dist_df4, type="l",xlab="t values", ylab="density", main="Comparision of t-distributions",
col= colors[4])

#Add lines for other degrees of freedom
lines(x,t_dist_df1, col=colors[1])
lines(x,t_dist_df2, col=colors[2])
lines(x,t_dist_df3, col=colors[3])

#Add a legend to the plot
legend("topright", legend=c("df=1","df=4","df=10","df=30"))


#Q4
v1<-10
v2<-20

#(i) find the 95th percentile of the f-distribution

percentile_95<-qf(0.95,df1=v1,df2=v2)
cat("95th percentile of the F-distribution :",percentile_95,"\n")

#(ii)calculate the area under the curve for the given intervals
area_interval_1<-pf(1.5,df1 =v1,df2=v2,lower.tail = TRUE) #[0,1.5]
area_interval_2<-1-area_interval_1
cat("Area under the curve for [0,1.5]:",area_interval_1,"\n")
cat("Area under the curve for [1.5,+inf]:",area_interval_2,"\n")

#(iii) calculate the quantiles for diff probab
quantile_25<-qf(0.25,df1=v1,df2=v2)
quantile_50<-qf(0.5,df1=v1,df2=v2)
quantile_75<-qf(0.75,df1=v1,df2=v2)
quantile_999<-qf(0.999,df1=v1,df2=v2)

cat("Quantile for p = 0.25",quantile_25,"\n")
cat("Quantile for p = 0.25",quantile_50,"\n")
cat("Quantile for p = 0.75",quantile_75,"\n")
cat("Quantile for p = 0.999",quantile_999,"\n")

#(iv) generate random values and plot a histogram
#set.seed(123) for reproducability

x<-rf(1000,df1=v1,df2=v2)
hist(x,breaks='scott',freq=FALSE,xlim=c(0,3),ylim=c(0,1),xlab="", col="orange")
```

EXPERIMENT 8

```
# (a) Import the csv data file in R
library(readr)
Clt_data <- read_csv("C:/Users/shree/Downloads/Clt-data.csv")
View(Clt_data)

# (b) Validate data for correctness
# Count number of rows
n_rows <- nrow(Clt_data)
print(paste("Number of rows: ", n_rows))
# View the top ten rows of the dataset
head(Clt_data, 10)
# Data about the column names
colnames(Clt_data)
str(Clt_data)


# (c) Calculate the population mean
population_mean <- mean(Clt_data$`Wall Thickness`)
print(population_mean)

# Plot the histogram of the observations
hist(Clt_data$`Wall Thickness`, breaks = 20, col = "lightblue", xlab = "Wall Thickness", main =
"Histogram of Wall Thickness")
# Add a vertical line for population mean
abline(v = population_mean, col = "red", lwd = 2)

# Function to draw sufficient samples and plot histograms
draw_samples_and_plot <- function(sample_size) {
  # Generate 1000 samples of specified size
  sample_means <- replicate(1000, mean(sample(Clt_data$'Wall Thickness', size = sample_size,
replace = TRUE)))

  # Plot histogram of sample means
  hist(sample_means, breaks = 30, col = "lightgreen", xlab = "Sample Means", main =
paste("Sampling Distribution (Sample Size =", sample_size, ")"))

  # Add a vertical line for the population mean
  abline(v = population_mean, col = "red", lwd = 2)
}
```

```
# (a) Draw samples of size 10 and plot their means
draw_samples_and_plot(10)

# (b) Repeat for sample sizes 50, 500, and 9000
draw_samples_and_plot(50)
draw_samples_and_plot(500)
draw_samples_and_plot(9000)
```