PROBABILITY AND STATISTICS LAB REPORT

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

Submitted By:

Name – Vanshikaa

Roll number -102103580

Batch - 3COE21

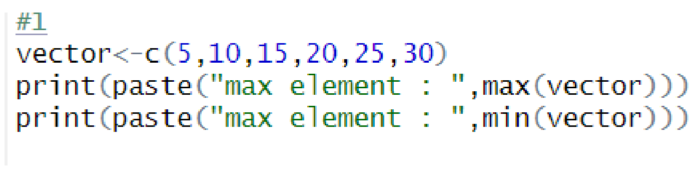
Submitted To:

Dr. Rajanish Rai

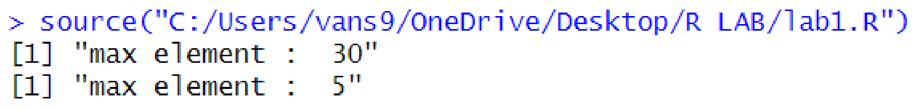
July 2023 – December 2023

LAB 1

Q1

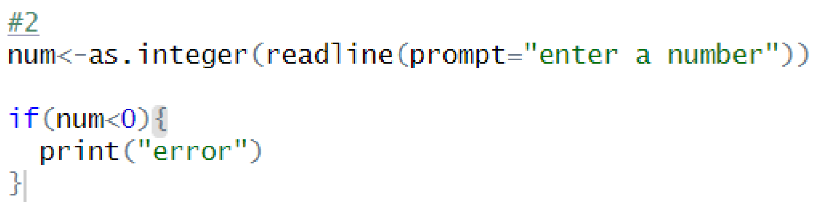


```r
#1
vector<-c(5,10,15,20,25,30)
print(paste("max element : ",max(vector)))
print(paste("max element : ",min(vector)))
```

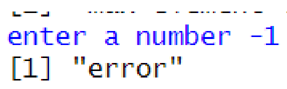


```
> source("C:/Users/vans9/OneDrive/Desktop/R LAB/lab1.R")
[1] "max element :  30"
[1] "max element :  5"
```

Q2

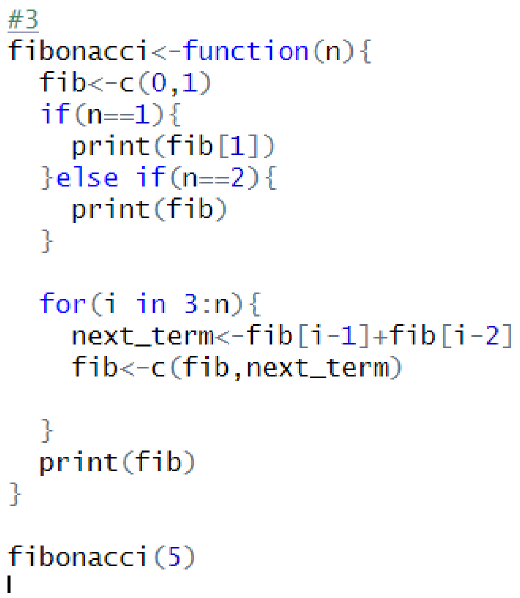


```r
#2
num<-as.integer(readline(prompt="enter a number"))

if(num<0){
  print("error")
}
```

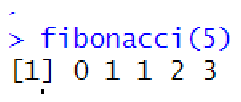


```
enter a number -1
[1] "error"
```

Q3

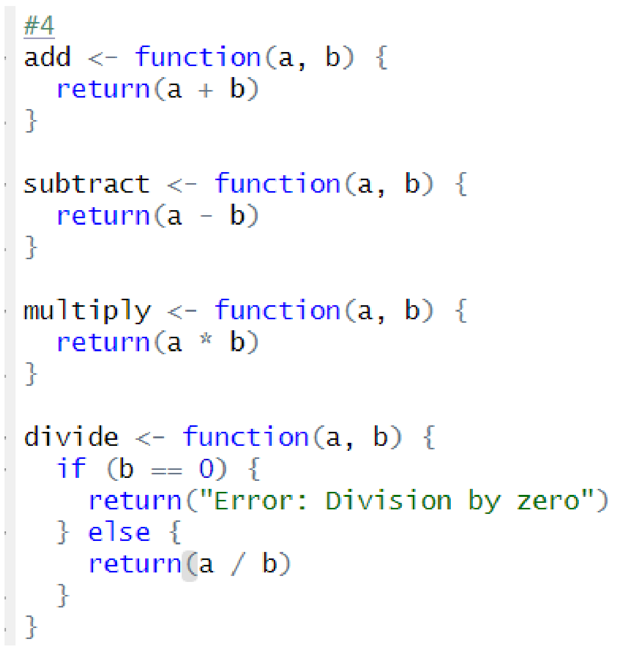


```r
#3
fibonacci<-function(n){
  fib<-c(0,1)
  if(n==1){
    print(fib[1])
  }else if(n==2){
    print(fib)
  }

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

  }
  print(fib)
}

fibonacci(5)
```

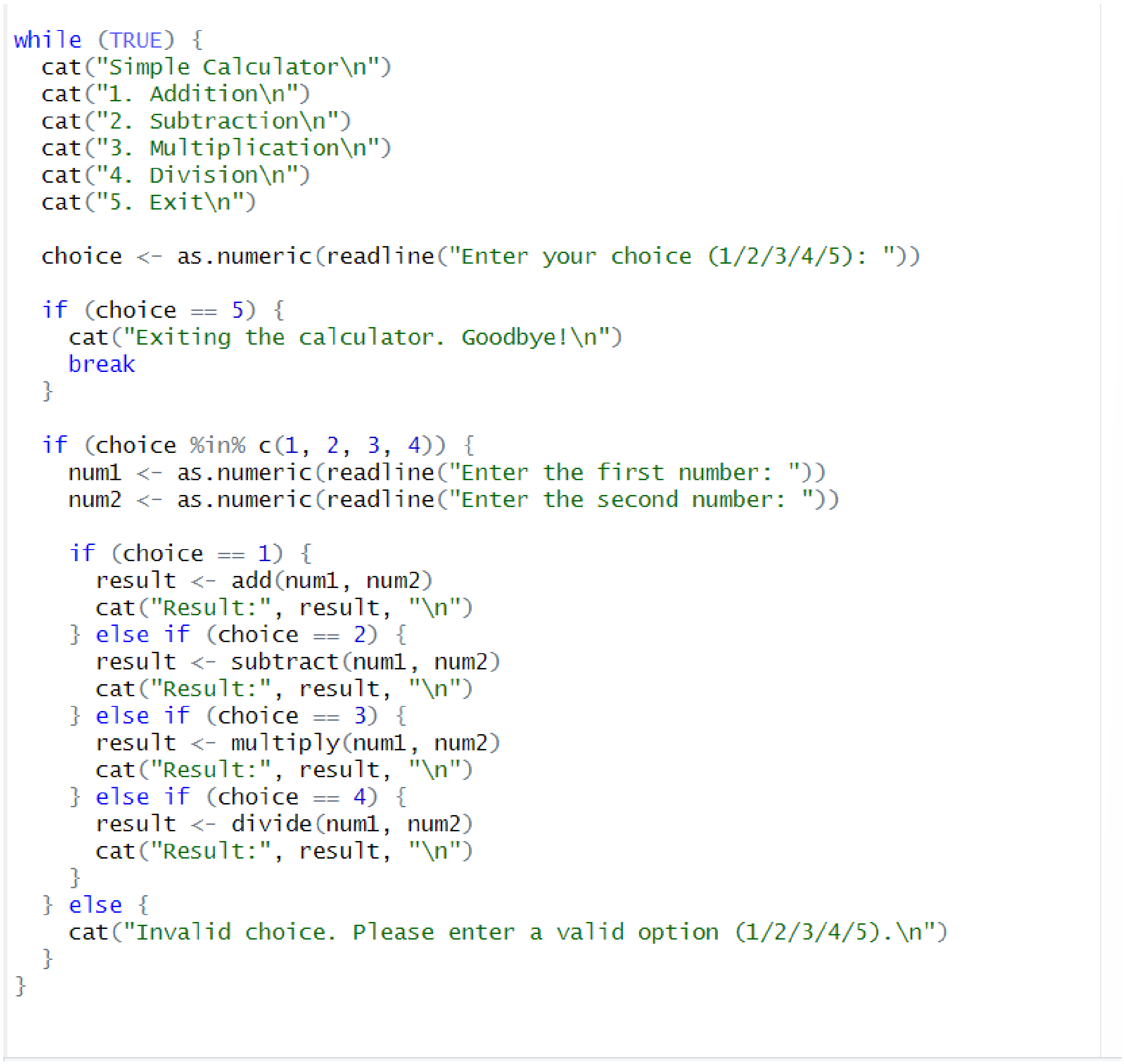


```
> fibonacci(5)
[1] 0 1 1 2 3
```

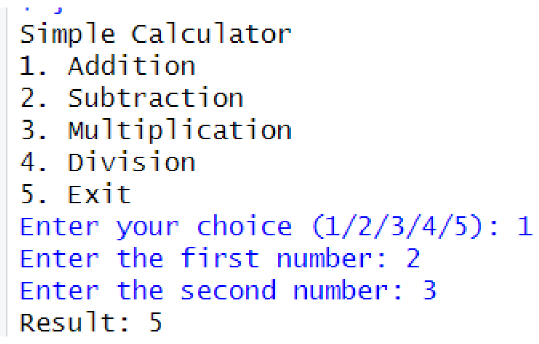
Q4



```r
#4
add <- function(a, b) {
  return(a + b)
}

subtract <- function(a, b) {
  return(a - b)
}

multiply <- function(a, b) {
  return(a * b)
}

divide <- function(a, b) {
  if (b == 0) {
    return("Error: Division by zero")
  } else {
    return(a / b)
  }
}
```

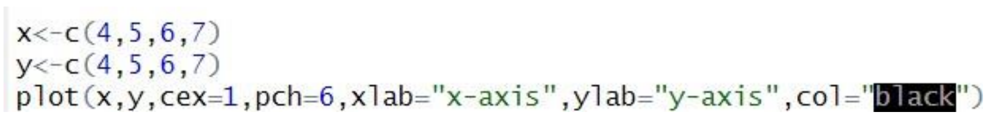


```
while (TRUE) {
  cat("Simple Calculator\n")
  cat("1. Addition\n")
  cat("2. Subtraction\n")
  cat("3. Multiplication\n")
  cat("4. Division\n")
  cat("5. Exit\n")

  choice <- as.numeric(readline("Enter your choice (1/2/3/4/5): "))

  if (choice == 5) {
    cat("Exiting the calculator. Goodbye!\n")
    break
  }

  if (choice %in% c(1, 2, 3, 4)) {
    num1 <- as.numeric(readline("Enter the first number: "))
    num2 <- as.numeric(readline("Enter the second number: "))

    if (choice == 1) {
      result <- add(num1, num2)
      cat("Result:", result, "\n")
    } else if (choice == 2) {
      result <- subtract(num1, num2)
      cat("Result:", result, "\n")
    } else if (choice == 3) {
      result <- multiply(num1, num2)
      cat("Result:", result, "\n")
    } else if (choice == 4) {
      result <- divide(num1, num2)
      cat("Result:", result, "\n")
    }
  } else {
    cat("Invalid choice. Please enter a valid option (1/2/3/4/5).\n")
  }
}
```

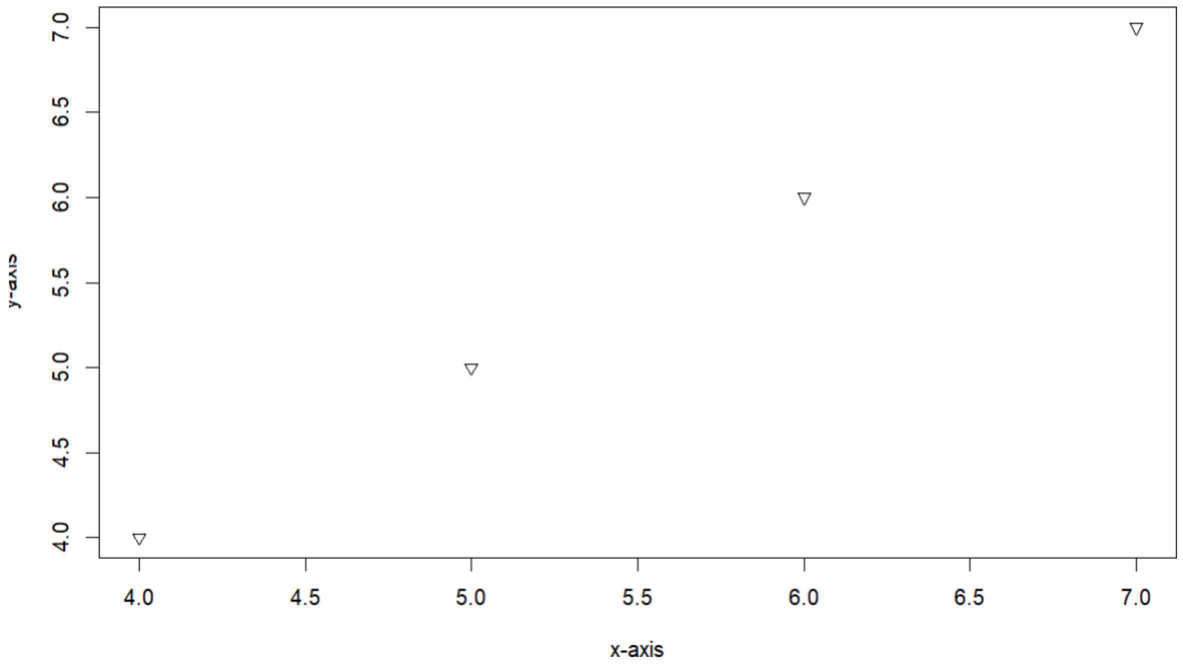


```
Simple Calculator
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Enter your choice (1/2/3/4/5): 1
Enter the first number: 2
Enter the second number: 3
Result: 5
```

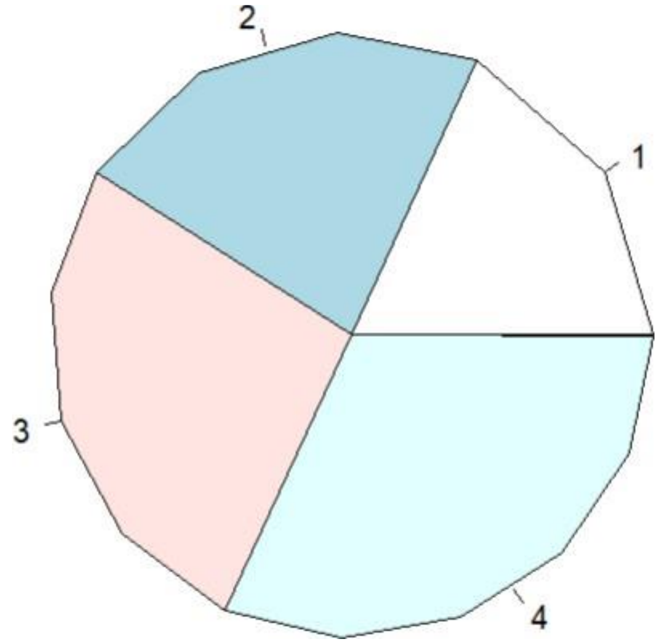
Q5

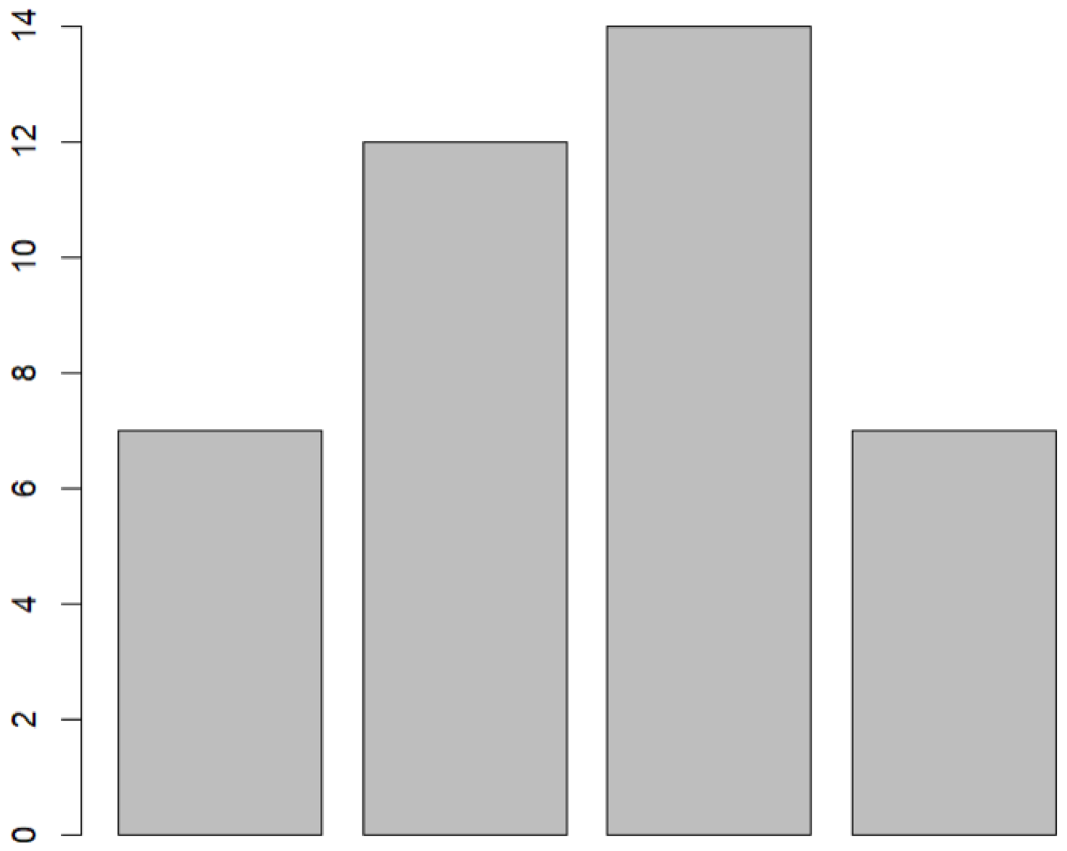


```
x<-c(4,5,6,7)
y<-c(4,5,6,7)
plot(x,y,cex=1,pch=6,xlab="x-axis",ylab="y-axis",col="black")
```



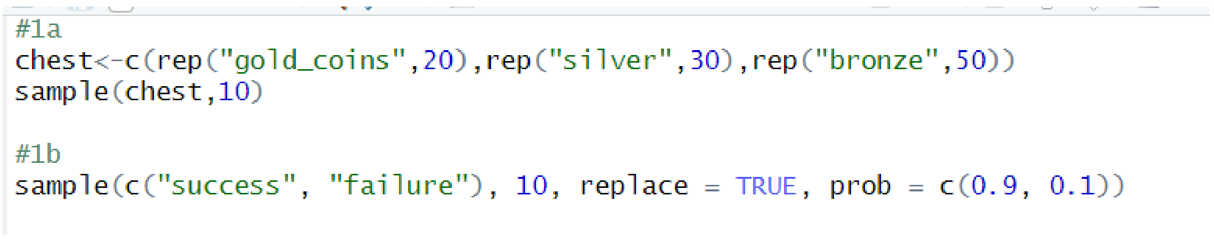
```
x<-c(4,5,6,7)
pie(x,edges = 20,radius = 0.8,clockwise = FALSE)

x<-c(7,12,14,7)
barplot(x)
```


1
2
3
4

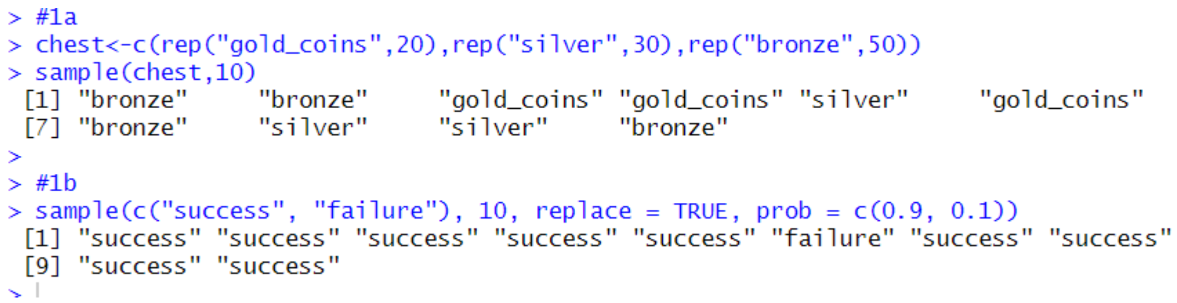

0
2
4
6
8
10
12
14

LAB 2

Q1

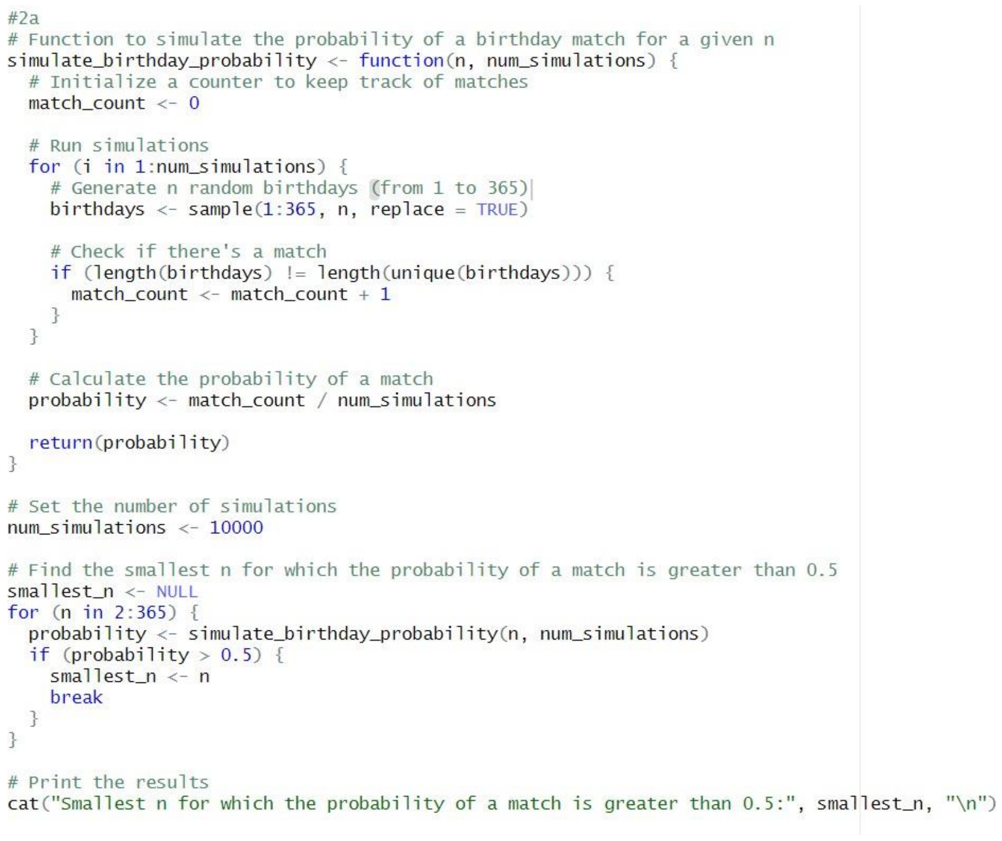


```r
#1a
chest<-c(rep("gold_coins",20),rep("silver",30),rep("bronze",50))
sample(chest,10)

#1b
sample(c("success", "failure"), 10, replace = TRUE, prob = c(0.9, 0.1))
```

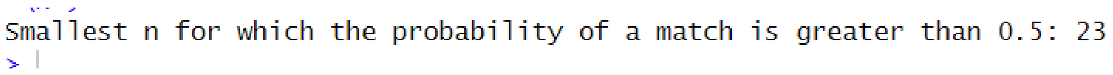


```
> #1a
> chest<-c(rep("gold_coins",20),rep("silver",30),rep("bronze",50))
> sample(chest,10)
 [1] "bronze"     "bronze"     "gold_coins" "gold_coins" "silver"     "gold_coins"
 [7] "bronze"     "silver"     "silver"     "bronze"    
> 
> #1b
> sample(c("success", "failure"), 10, replace = TRUE, prob = c(0.9, 0.1))
 [1] "success" "success" "success" "success" "success" "failure" "success" "success"
 [9] "success" "success"
> 
```

Q2



```r
#2a
# Function to simulate the probability of a birthday match for a given n
simulate_birthday_probability <- function(n, num_simulations) {
  # Initialize a counter to keep track of matches
  match_count <- 0
  
  # Run simulations
  for (i in 1:num_simulations) {
    # Generate n random birthdays (from 1 to 365)
    birthdays <- sample(1:365, n, replace = TRUE)
    
    # Check if there's a match
    if (length(birthdays) != length(unique(birthdays))) {
      match_count <- match_count + 1
    }
  }
  
  # Calculate the probability of a match
  probability <- match_count / num_simulations
  
  return(probability)
}

# Set the number of simulations
num_simulations <- 10000

# Find the smallest n for which the probability of a match is greater than 0.5
smallest_n <- NULL
for (n in 2:365) {
  probability <- simulate_birthday_probability(n, num_simulations)
  if (probability > 0.5) {
    smallest_n <- n
    break
  }
}

# Print the results
cat("Smallest n for which the probability of a match is greater than 0.5:", smallest_n, "\n")
```



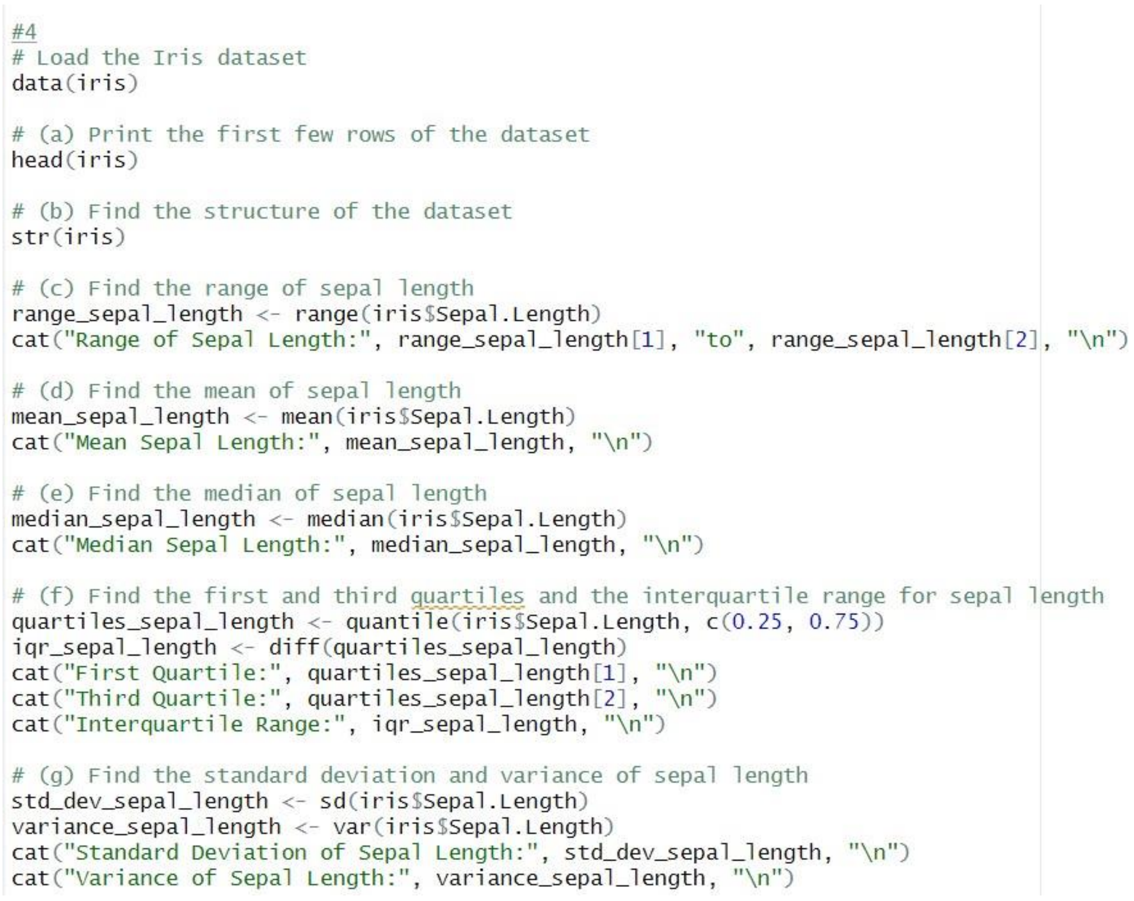
```
Smallest n for which the probability of a match is greater than 0.5: 23
> 
```

Q3

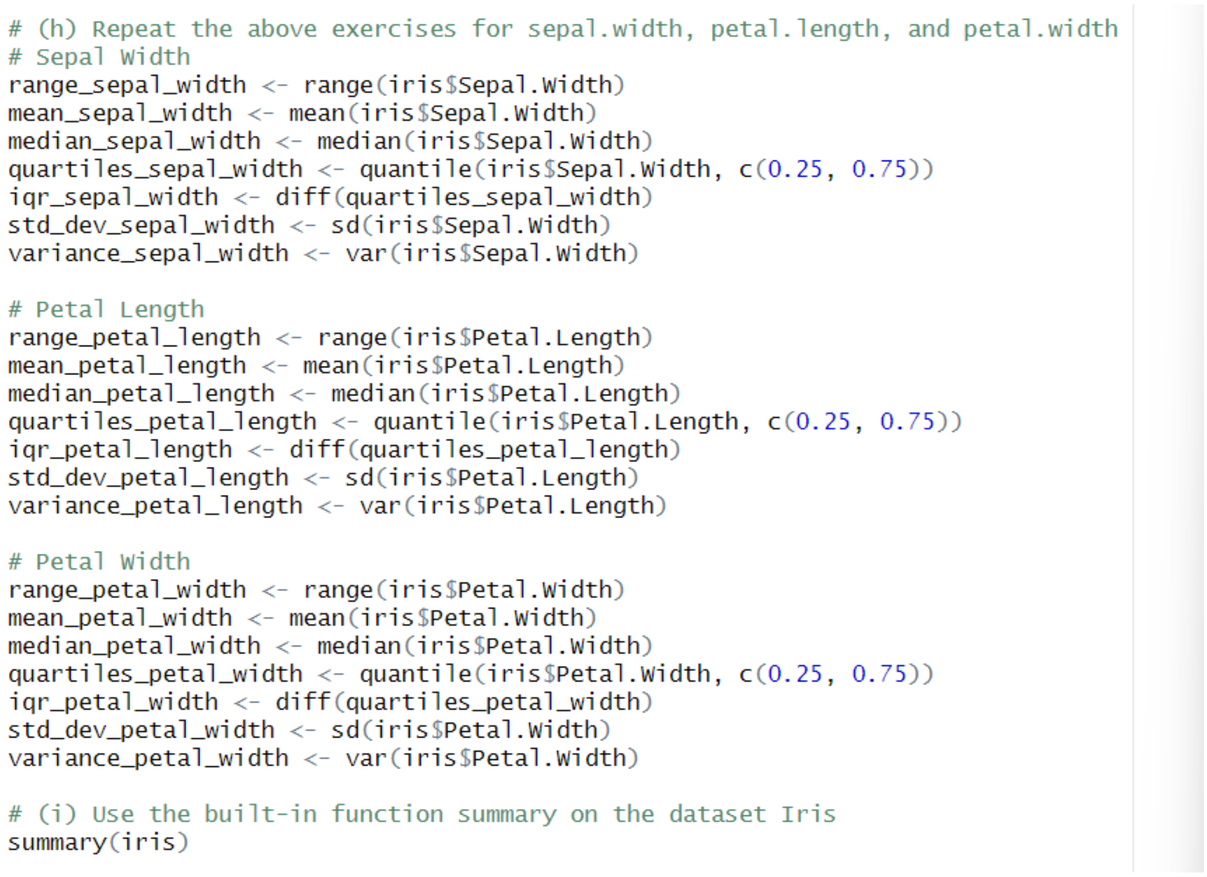
```
conditional_prob<-function(P_cloud,P_rain,P_cloud_rain){


  P_rain_cloud<-P_cloud_rain*P_rain/P_cloud

  return (P_rain_cloud)

}
P_cloud<-0.4
P_rain<-0.2
P_cloud_rain<-0.85

ans<-conditional_prob(P_cloud,P_rain,P_cloud_rain )
print(ans)
```

```
> conditional_prob<-function(P_cloud,P_rain,P_cloud_rain){
+ 
+ 
+   P_rain_cloud<-P_cloud_rain*P_rain/P_cloud
+ 
+   return (P_rain_cloud)
+ 
+ }
> P_cloud<-0.4
> P_rain<-0.2
> P_cloud_rain<-0.85
> 
> ans<-conditional_prob(P_cloud,P_rain,P_cloud_rain )
> print(ans)
[1] 0.425
```

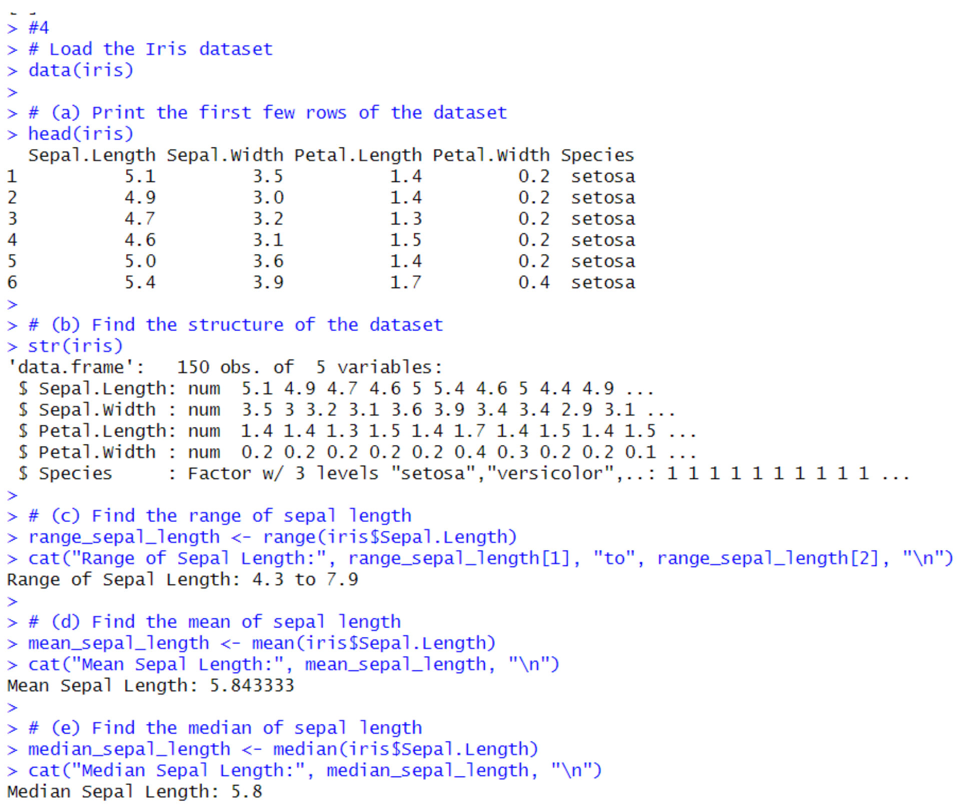
Q4



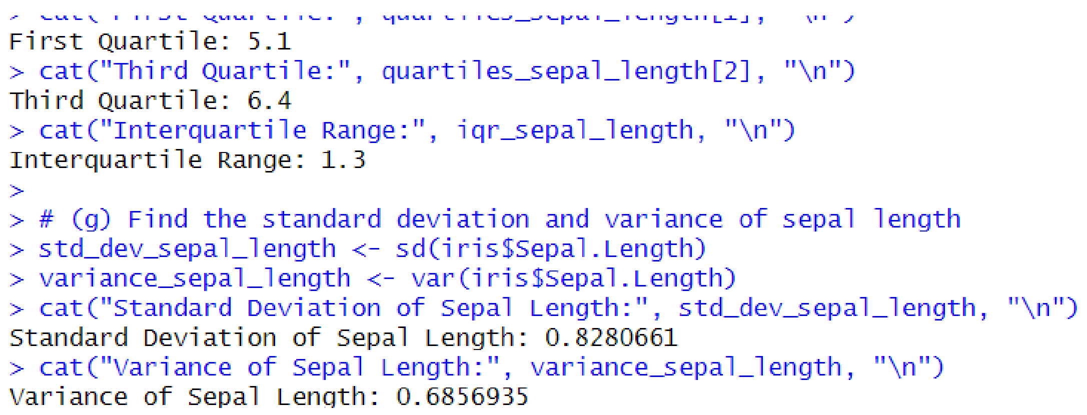
```r
#4
# Load the Iris dataset
data(iris)

# (a) Print the 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[1], "to", range_sepal_length[2], "\n")

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

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

# (f) Find the first and third quartiles and the interquartile range for sepal length
quartiles_sepal_length <- quantile(iris$Sepal.Length, c(0.25, 0.75))
iqr_sepal_length <- diff(quartiles_sepal_length)
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
std_dev_sepal_length <- sd(iris$Sepal.Length)
variance_sepal_length <- var(iris$Sepal.Length)
cat("Standard Deviation of Sepal Length:", std_dev_sepal_length, "\n")
cat("Variance of Sepal Length:", variance_sepal_length, "\n")
```



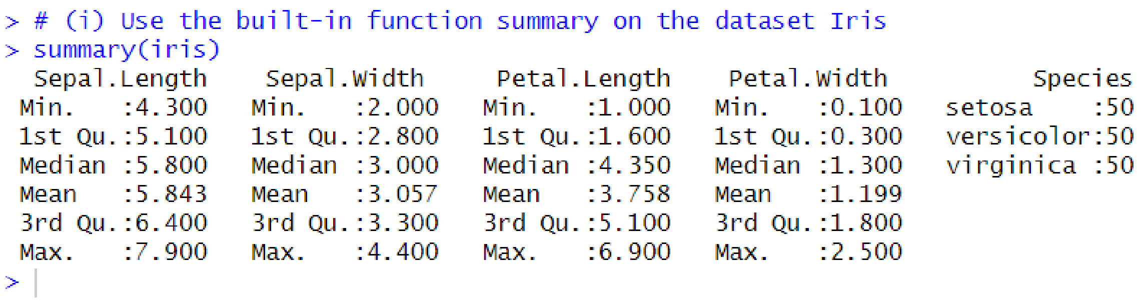
```r
# (h) Repeat the above exercises for sepal.width, petal.length, and petal.width
# Sepal Width
range_sepal_width <- range(iris$Sepal.Width)
mean_sepal_width <- mean(iris$Sepal.Width)
median_sepal_width <- median(iris$Sepal.Width)
quartiles_sepal_width <- quantile(iris$Sepal.Width, c(0.25, 0.75))
iqr_sepal_width <- diff(quartiles_sepal_width)
std_dev_sepal_width <- sd(iris$Sepal.Width)
variance_sepal_width <- var(iris$Sepal.Width)

# Petal Length
range_petal_length <- range(iris$Petal.Length)
mean_petal_length <- mean(iris$Petal.Length)
median_petal_length <- median(iris$Petal.Length)
quartiles_petal_length <- quantile(iris$Petal.Length, c(0.25, 0.75))
iqr_petal_length <- diff(quartiles_petal_length)
std_dev_petal_length <- sd(iris$Petal.Length)
variance_petal_length <- var(iris$Petal.Length)

# Petal Width
range_petal_width <- range(iris$Petal.Width)
mean_petal_width <- mean(iris$Petal.Width)
median_petal_width <- median(iris$Petal.Width)
quartiles_petal_width <- quantile(iris$Petal.Width, c(0.25, 0.75))
iqr_petal_width <- diff(quartiles_petal_width)
std_dev_petal_width <- sd(iris$Petal.Width)
variance_petal_width <- var(iris$Petal.Width)

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



```
> #4
> # Load the Iris dataset
> data(iris)
>
> # (a) Print the first few rows of the dataset
> 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 the dataset
> 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 sepal length
> range_sepal_length <- range(iris$Sepal.Length)
> cat("Range of Sepal Length:", range_sepal_length[1], "to", range_sepal_length[2], "\n")
Range of Sepal Length: 4.3 to 7.9
>
> # (d) Find the mean of sepal length
> mean_sepal_length <- mean(iris$Sepal.Length)
> cat("Mean Sepal Length:", mean_sepal_length, "\n")
Mean Sepal Length: 5.843333
>
> # (e) Find the median of sepal length
> median_sepal_length <- median(iris$Sepal.Length)
> cat("Median Sepal Length:", median_sepal_length, "\n")
Median Sepal Length: 5.8
>
```

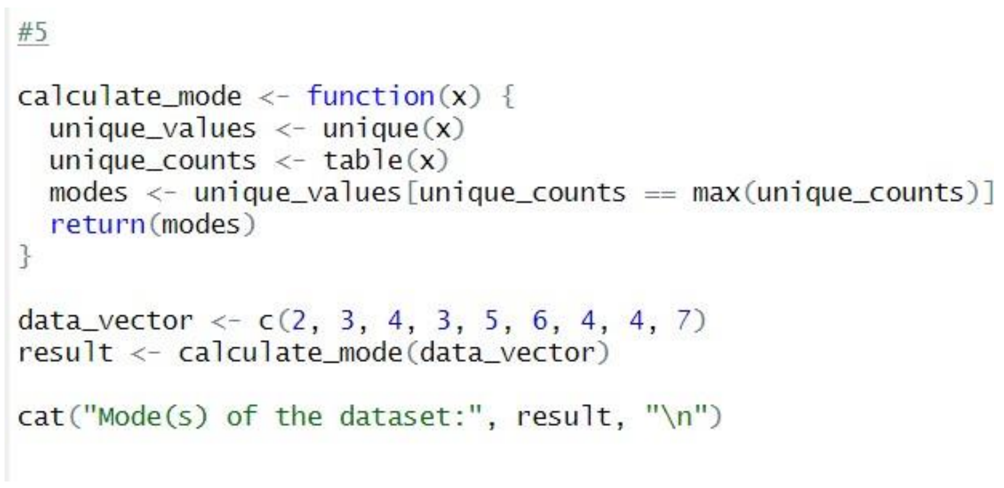


```
> cat("First Quartile:", quartiles_sepal_length[1], "\n")
First Quartile: 5.1
> cat("Third Quartile:", quartiles_sepal_length[2], "\n")
Third Quartile: 6.4
> cat("Interquartile Range:", iqr_sepal_length, "\n")
Interquartile Range: 1.3
>
> # (g) Find the standard deviation and variance of sepal length
> std_dev_sepal_length <- sd(iris$Sepal.Length)
> variance_sepal_length <- var(iris$Sepal.Length)
> cat("Standard Deviation of Sepal Length:", std_dev_sepal_length, "\n")
Standard Deviation of Sepal Length: 0.8280661
> cat("Variance of Sepal Length:", variance_sepal_length, "\n")
Variance of Sepal Length: 0.6856935
```

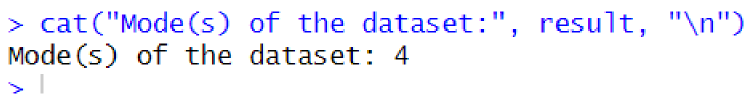


```
> # (i) Use the built-in function summary on the dataset Iris
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species  
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50  
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50  
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50  
 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                  
> 
```

Q5



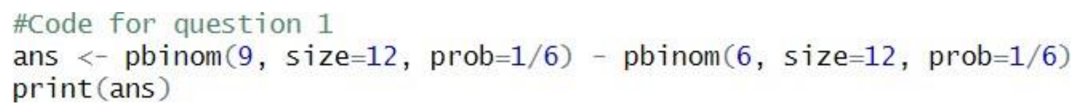
```r
#5

calculate_mode <- function(x) {
  unique_values <- unique(x)
  unique_counts <- table(x)
  modes <- unique_values[unique_counts == max(unique_counts)]
  return(modes)
}

data_vector <- c(2, 3, 4, 3, 5, 6, 4, 4, 7)
result <- calculate_mode(data_vector)

cat("Mode(s) of the dataset:", result, "\n")
```



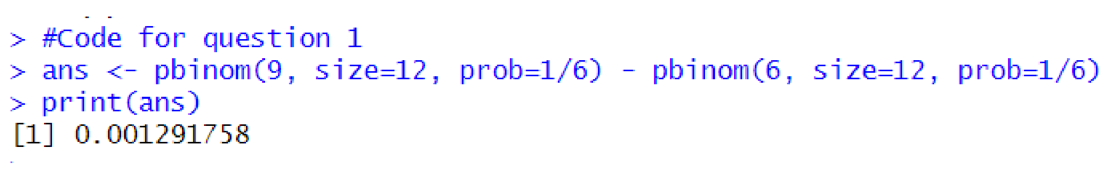
```
> cat("Mode(s) of the dataset:", result, "\n")
Mode(s) of the dataset: 4
> 
```

LAB 3

Q1

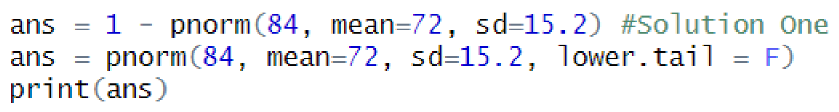


```
#Code for question 1
ans <- pbinom(9, size=12, prob=1/6) - pbinom(6, size=12, prob=1/6)
print(ans)
```

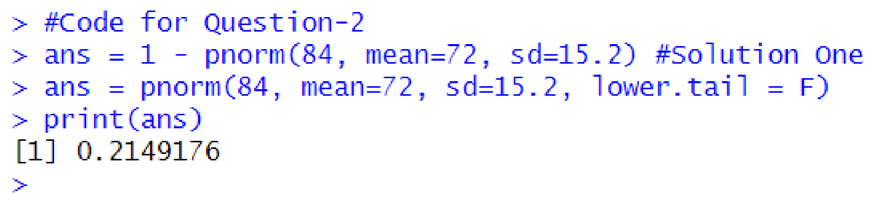


```
> #Code for question 1
> ans <- pbinom(9, size=12, prob=1/6) - pbinom(6, size=12, prob=1/6)
> print(ans)
[1] 0.001291758
```

Q2

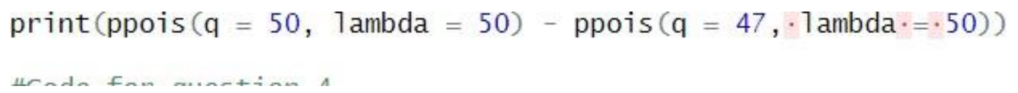


```
ans = 1 - pnorm(84, mean=72, sd=15.2) #Solution One
ans = pnorm(84, mean=72, sd=15.2, lower.tail = F)
print(ans)
```

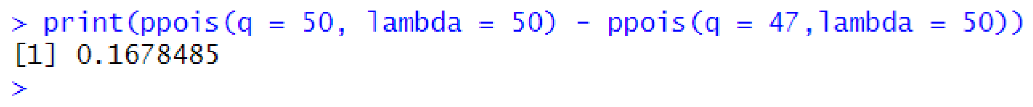


```
> #Code for Question-2
> ans = 1 - pnorm(84, mean=72, sd=15.2) #Solution One
> ans = pnorm(84, mean=72, sd=15.2, lower.tail = F)
> print(ans)
[1] 0.2149176
>
```

Q3

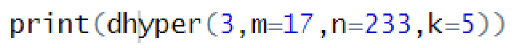


```
print(ppois(q = 50, lambda = 50) - ppois(q = 47, lambda = 50))
```

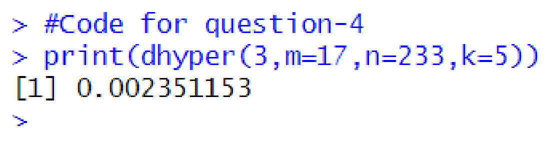


```
> print(ppois(q = 50, lambda = 50) - ppois(q = 47,lambda = 50))
[1] 0.1678485
>
```

Q4

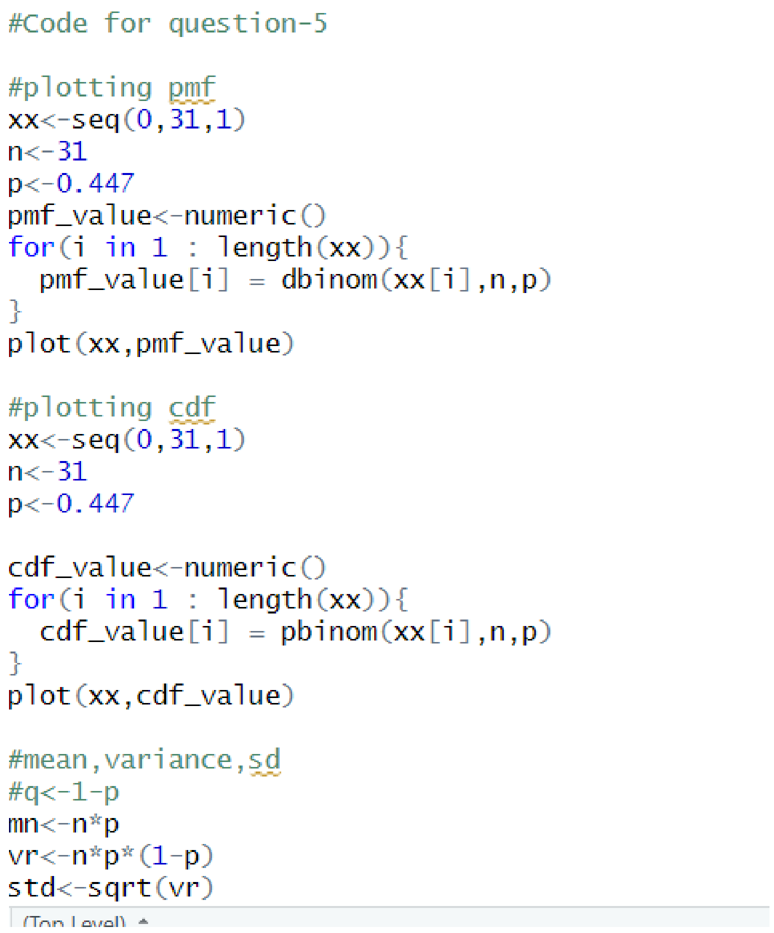


```
print(dhyper(3,m=17,n=233,k=5))
```

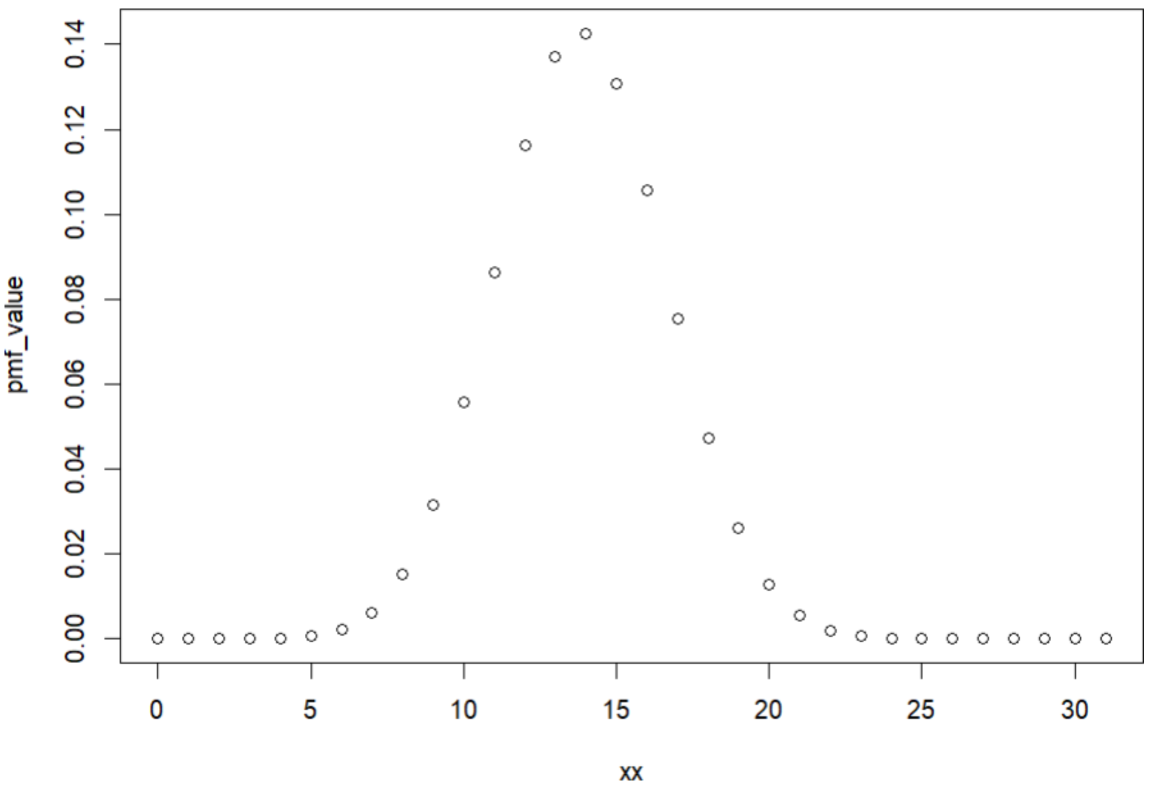


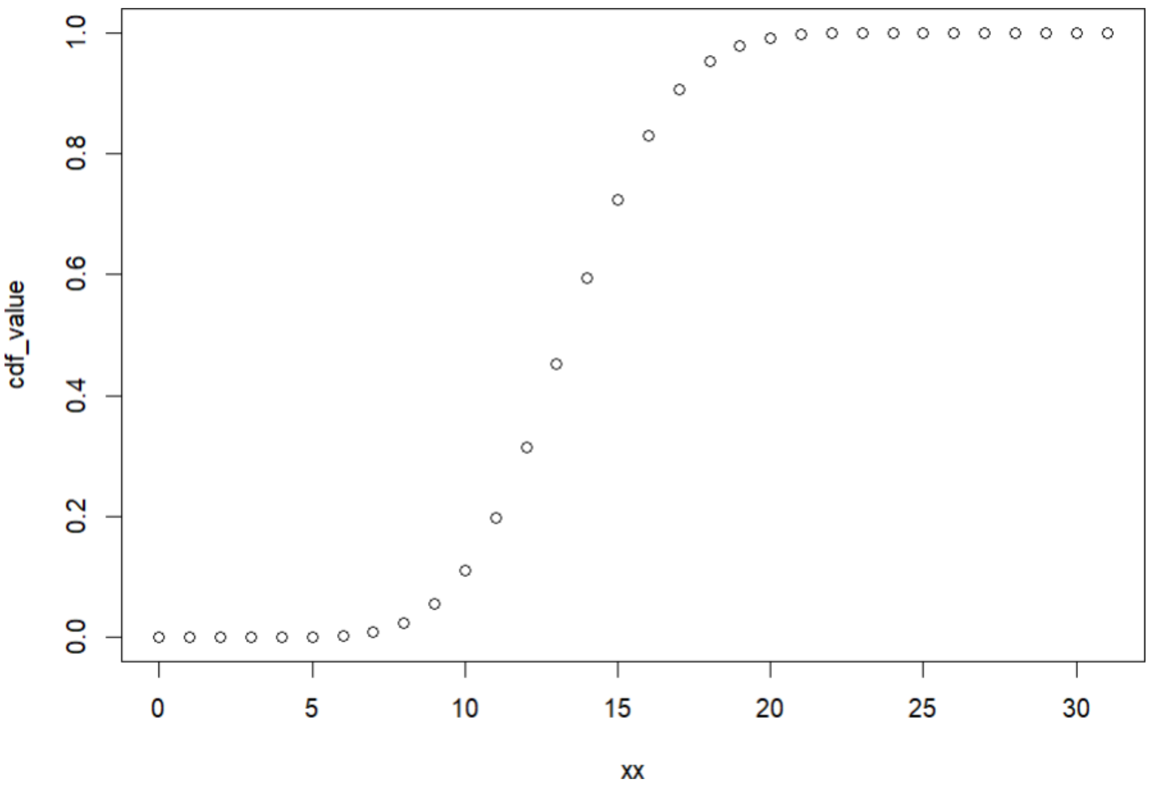
```
> #Code for question-4
> print(dhyper(3,m=17,n=233,k=5))
[1] 0.002351153
>
```

Q5



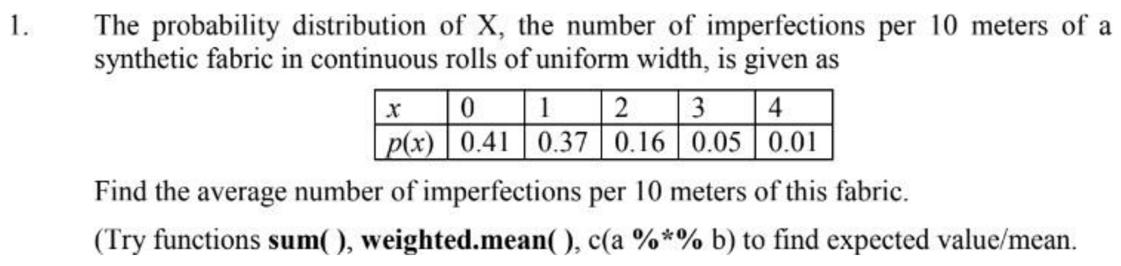
```
#Code for question-5

#plotting pmf
xx<-seq(0,31,1)
n<-31
p<-0.447
pmf_value<-numeric()
for(i in 1 : length(xx)){
  pmf_value[i] = dbinom(xx[i],n,p)
}
plot(xx,pmf_value)

#plotting cdf
xx<-seq(0,31,1)
n<-31
p<-0.447

cdf_value<-numeric()
for(i in 1 : length(xx)){
  cdf_value[i] = pbinom(xx[i],n,p)
}
plot(xx,cdf_value)

#mean,variance,sd
#q<-1-p
mn<-n*p
vr<-n*p*(1-p)
std<-sqrt(vr)
```


pmf_value
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0
5
10
15
20
25
30
xx


cdf_value
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
xx

LAB 4

Q1

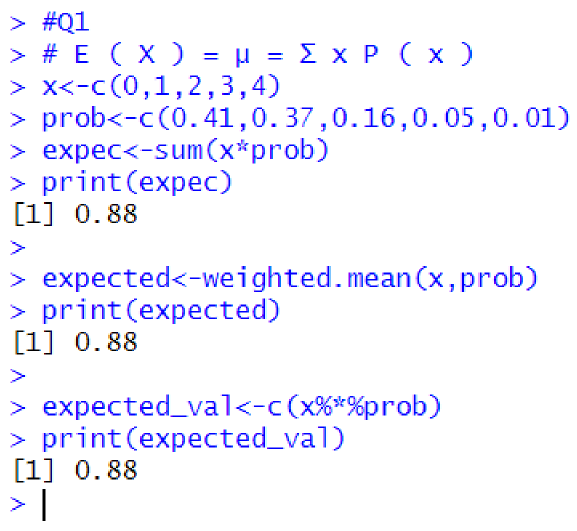


1. The probability distribution of X, the number of imperfections per 10 meters of a synthetic fabric in continuous rolls of uniform width, is given as

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

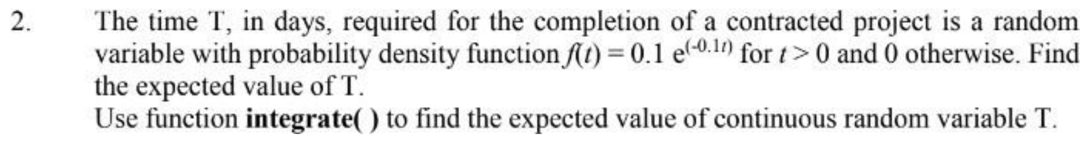
Find the average number of imperfections per 10 meters of this fabric.

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

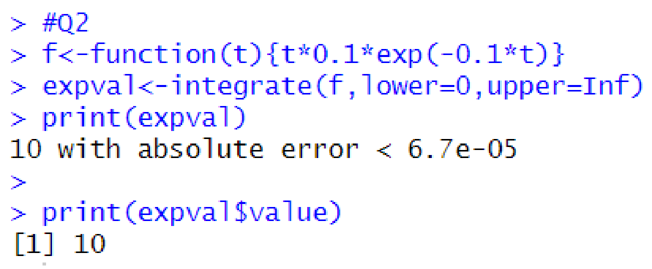


```
> #Q1
> # E ( X ) = μ = Σ x P ( x )
> x<-c(0,1,2,3,4)
> prob<-c(0.41,0.37,0.16,0.05,0.01)
> expec<-sum(x*prob)
> print(expec)
[1] 0.88
>
> expected<-weighted.mean(x,prob)
> print(expected)
[1] 0.88
>
> expected_val<-c(x%*%prob)
> print(expected_val)
[1] 0.88
> |
```

Q2



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.



```
> #Q2
> f<-function(t){t*0.1*exp(-0.1*t)}
> expval<-integrate(f,lower=0,upper=Inf)
> print(expval)
10 with absolute error < 6.7e-05
>
> print(expval$value)
[1] 10
```

Q3

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

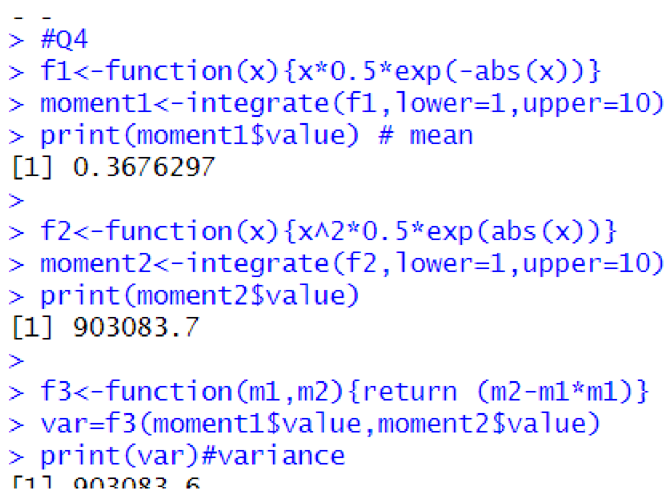
x	0	1	2	3
$p(x)$	0.1	0.2	0.2	0.5

Find the expected value of Y.

```
> #Q3
> x<-c(0,1,2,3)
> prob<-c(0.1,0.2,0.2,0.5)
> #y<-12*x+2*(3-x)-6*x
> y<-10*x-12
> expectedVal<-sum(y*prob)
> print(expectedVal)
[1] 9
```

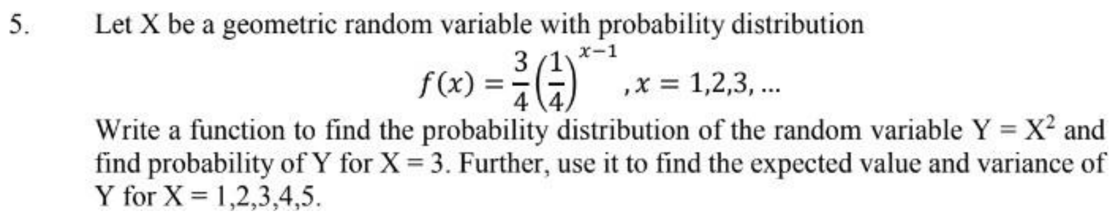
Q4

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.



```
> #Q4
> f1<-function(x){x*0.5*exp(-abs(x))}
> moment1<-integrate(f1,lower=1,upper=10)
> print(moment1$value) # mean
[1] 0.3676297
>
> f2<-function(x){x^2*0.5*exp(abs(x))}
> moment2<-integrate(f2,lower=1,upper=10)
> print(moment2$value)
[1] 903083.7
>
> f3<-function(m1,m2){return (m2-m1*m1)}
> var=f3(moment1$value,moment2$value)
> print(var)#variance
[1] 903083.6
```

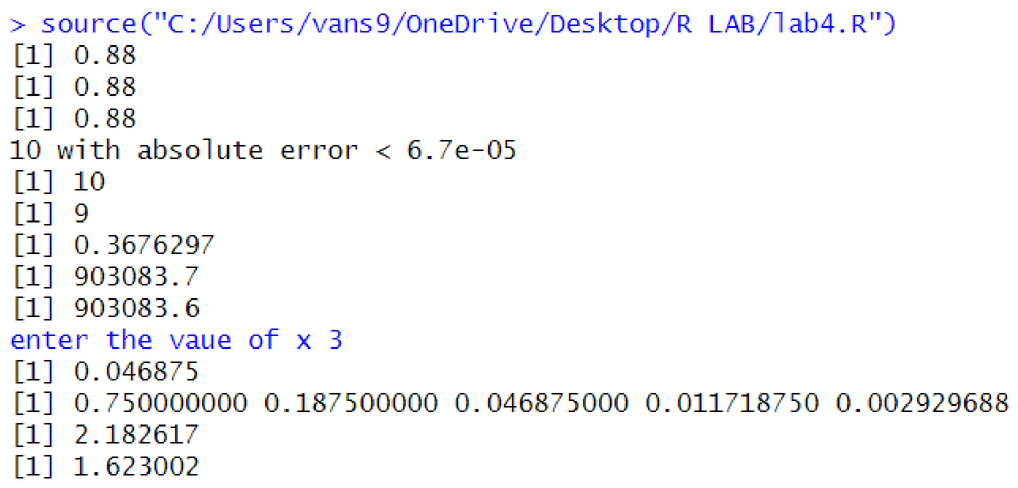
Q5



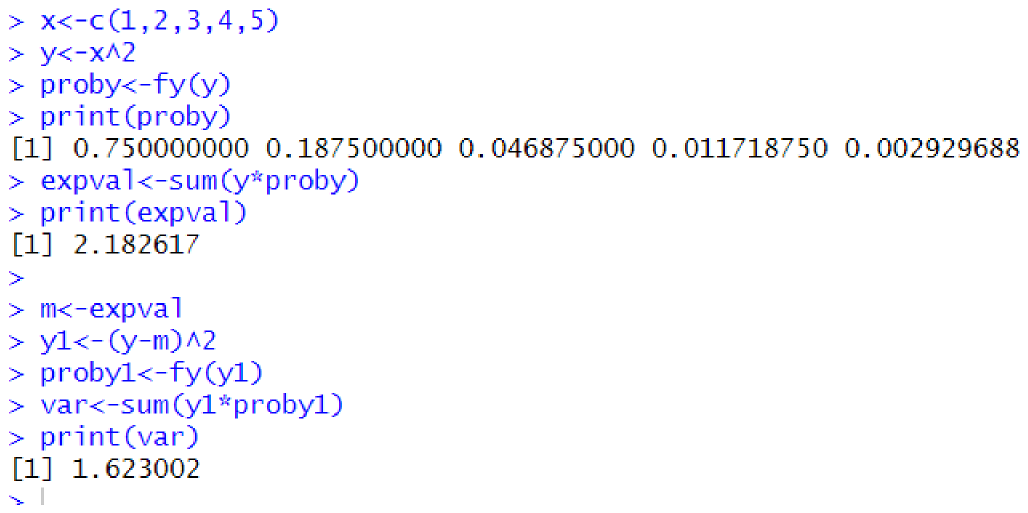
5. Let X be a geometric random variable with probability distribution

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

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.



```
> source("C:/Users/vans9/OneDrive/Desktop/R LAB/lab4.R")
[1] 0.88
[1] 0.88
[1] 0.88
10 with absolute error < 6.7e-05
[1] 10
[1] 9
[1] 0.3676297
[1] 903083.7
[1] 903083.6
enter the vaue of x 3
[1] 0.046875
[1] 0.750000000 0.187500000 0.046875000 0.011718750 0.002929688
[1] 2.182617
[1] 1.623002
```

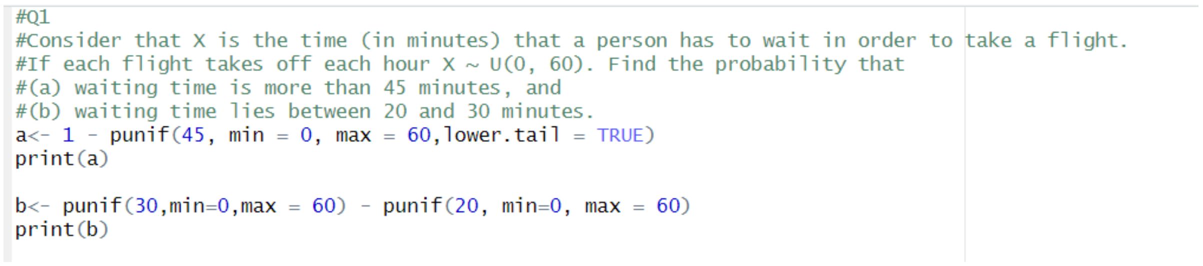


```
> x<-c(1,2,3,4,5)
> y<-x^2
> proby<-fy(y)
> print(proby)
[1] 0.750000000 0.187500000 0.046875000 0.011718750 0.002929688
> expval<-sum(y*proby)
> print(expval)
[1] 2.182617
>
> m<-expval
> y1<-(y-m)^2
> proby1<-fy(y1)
> var<-sum(y1*proby1)
> print(var)
[1] 1.623002
>
```

LAB 5

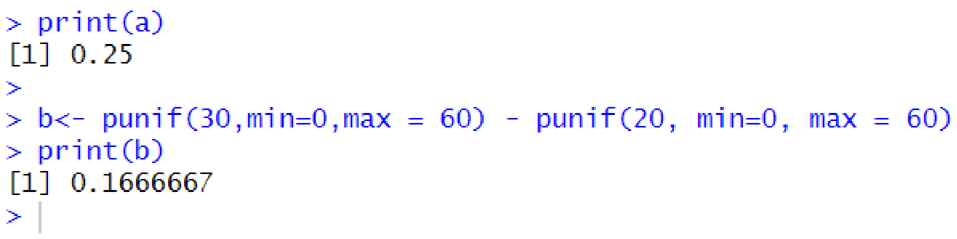
Q1

Code :



```
#Q1
#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
#(b) waiting time lies between 20 and 30 minutes.
a<- 1 - punif(45, min = 0, max = 60,lower.tail = TRUE)
print(a)

b<- punif(30,min=0,max = 60) - punif(20, min=0, max = 60)
print(b)
```

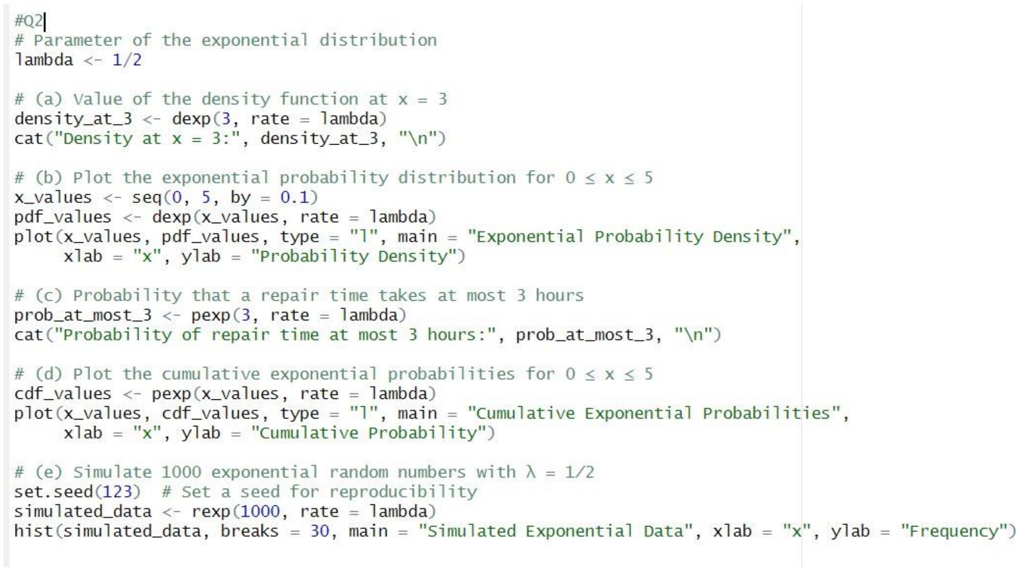
Output :



```
> print(a)
[1] 0.25
>
> b<- punif(30,min=0,max = 60) - punif(20, min=0, max = 60)
> print(b)
[1] 0.1666667
> 
```

Q2

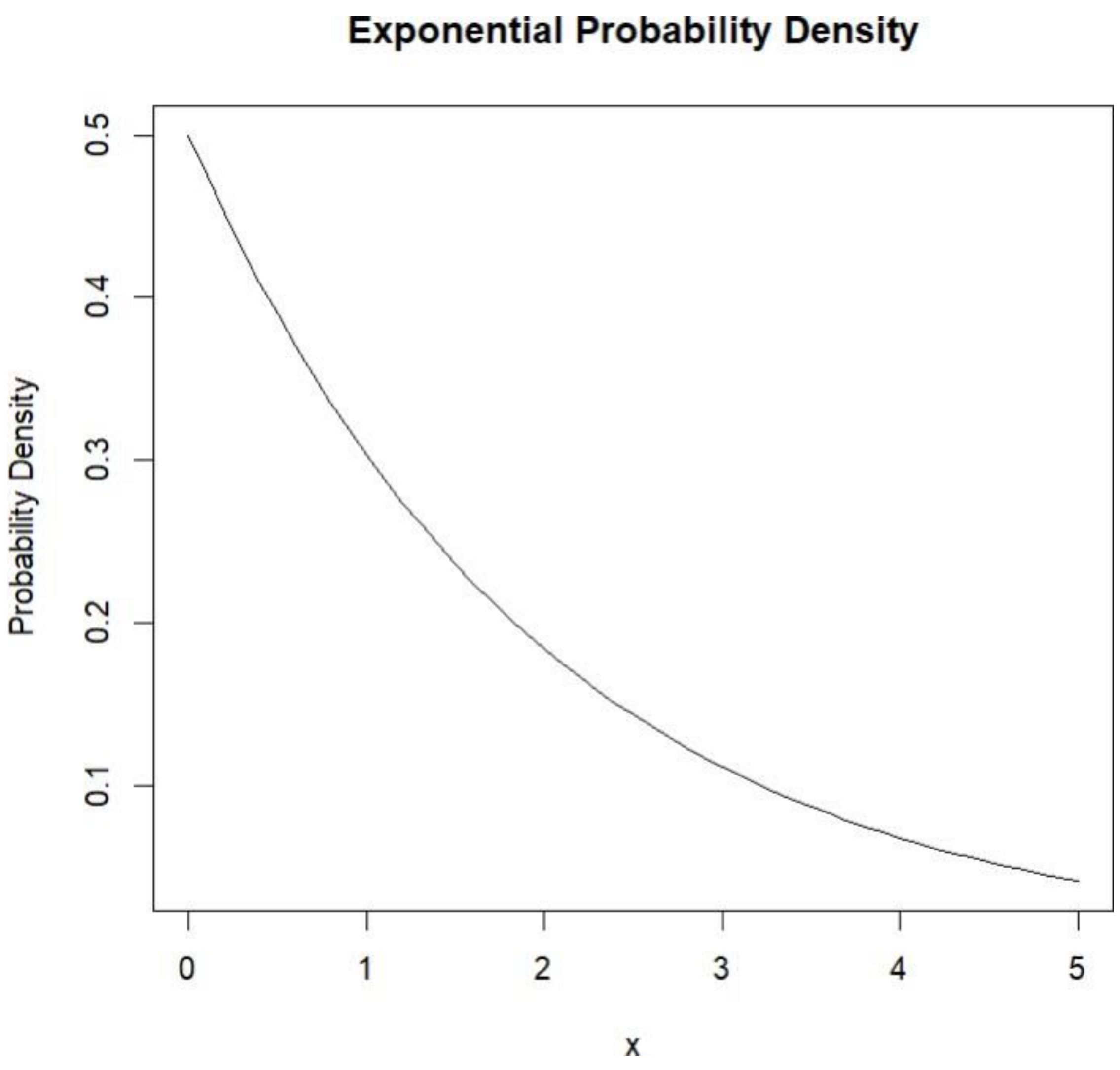
Code :



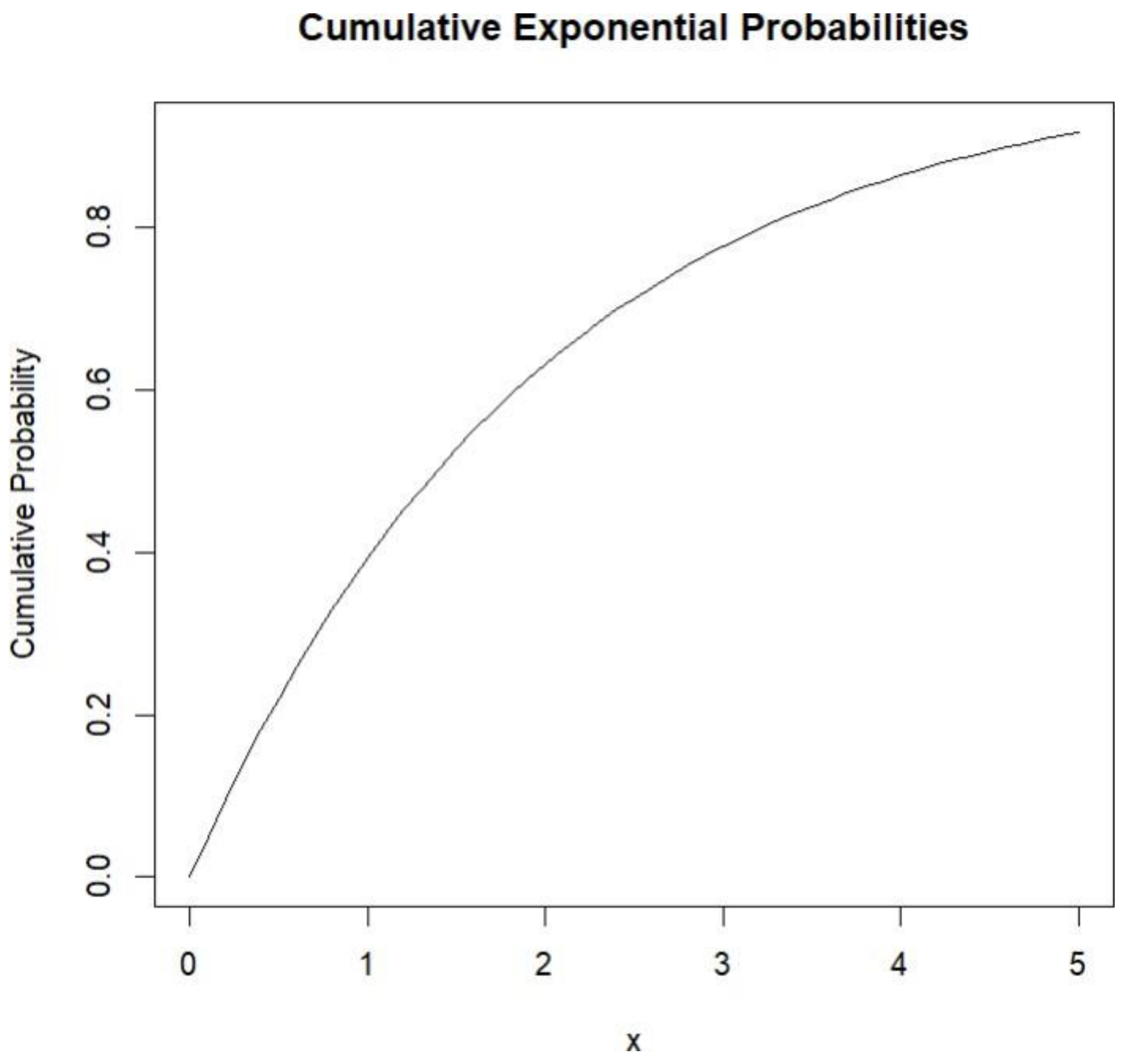
```
#Q2
# Parameter of the exponential distribution
lambda <- 1/2

# (a) Value of the density function at x = 3
density_at_3 <- dexp(3, rate = lambda)
cat("Density at x = 3:", density_at_3, "\n")

# (b) Plot the exponential probability distribution for 0 ≤ x ≤ 5
x_values <- seq(0, 5, by = 0.1)
pdf_values <- dexp(x_values, rate = lambda)
plot(x_values, pdf_values, type = "l", main = "Exponential Probability Density",
     xlab = "x", ylab = "Probability Density")

# (c) Probability that a repair time takes at most 3 hours
prob_at_most_3 <- pexp(3, rate = lambda)
cat("Probability of repair time at most 3 hours:", prob_at_most_3, "\n")

# (d) Plot the cumulative exponential probabilities for 0 ≤ x ≤ 5
cdf_values <- pexp(x_values, rate = lambda)
plot(x_values, cdf_values, type = "l", main = "Cumulative Exponential Probabilities",
     xlab = "x", ylab = "Cumulative Probability")

# (e) Simulate 1000 exponential random numbers with λ = 1/2
set.seed(123)  # Set a seed for reproducibility
simulated_data <- rexp(1000, rate = lambda)
hist(simulated_data, breaks = 30, main = "Simulated Exponential Data", xlab = "x", ylab = "Frequency")
```

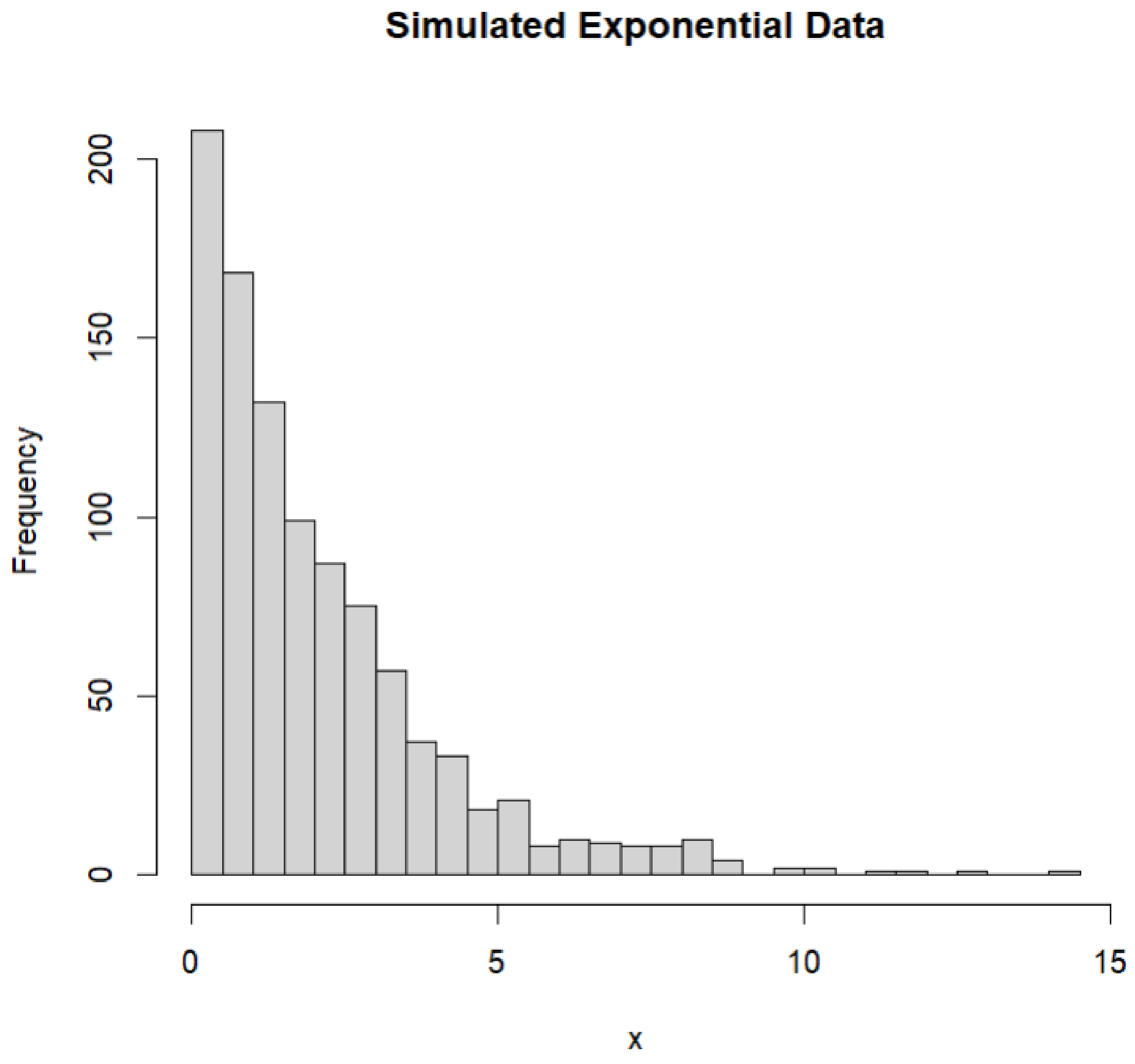
Output :

```
> # (a) Value of the density function at x = 3
> density_at_3 <- dexp(3, rate = lambda)
> cat("Density at x = 3:", density_at_3, "\n")
Density at x = 3: 0.1115651
```



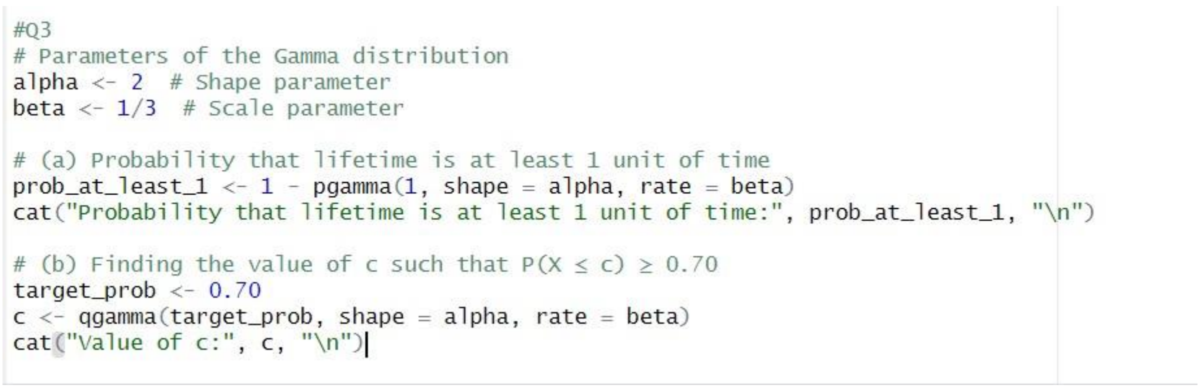
```
> # (c) Probability that a repair time takes at most 3 hours
> prob_at_most_3 <- pexp(3, rate = lambda)
> cat("Probability of repair time at most 3 hours:", prob_at_most_3, "\n")
Probability of repair time at most 3 hours: 0.7768698
```


Cumulative Exponential Probabilities
Cumulative Probability
0.0
0.2
0.4
0.6
0.8
0
1
2
3
4
5
x



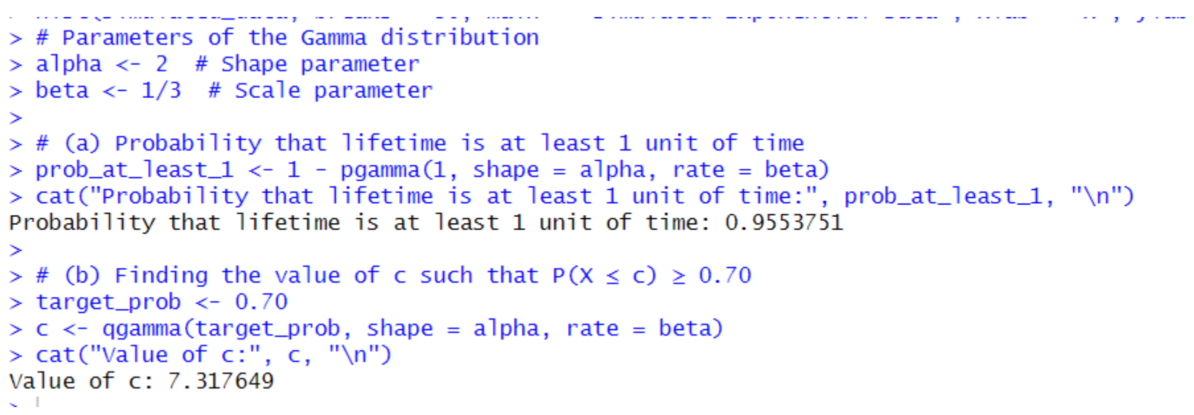
Q3

Code :



```
#Q3
# Parameters of the Gamma distribution
alpha <- 2  # Shape parameter
beta <- 1/3  # Scale parameter

# (a) Probability that lifetime is at least 1 unit of time
prob_at_least_1 <- 1 - pgamma(1, shape = alpha, rate = beta)
cat("Probability that lifetime is at least 1 unit of time:", prob_at_least_1, "\n")

# (b) Finding the value of c such that P(X ≤ c) ≥ 0.70
target_prob <- 0.70
c <- qgamma(target_prob, shape = alpha, rate = beta)
cat("Value of c:", c, "\n")
```

Output :

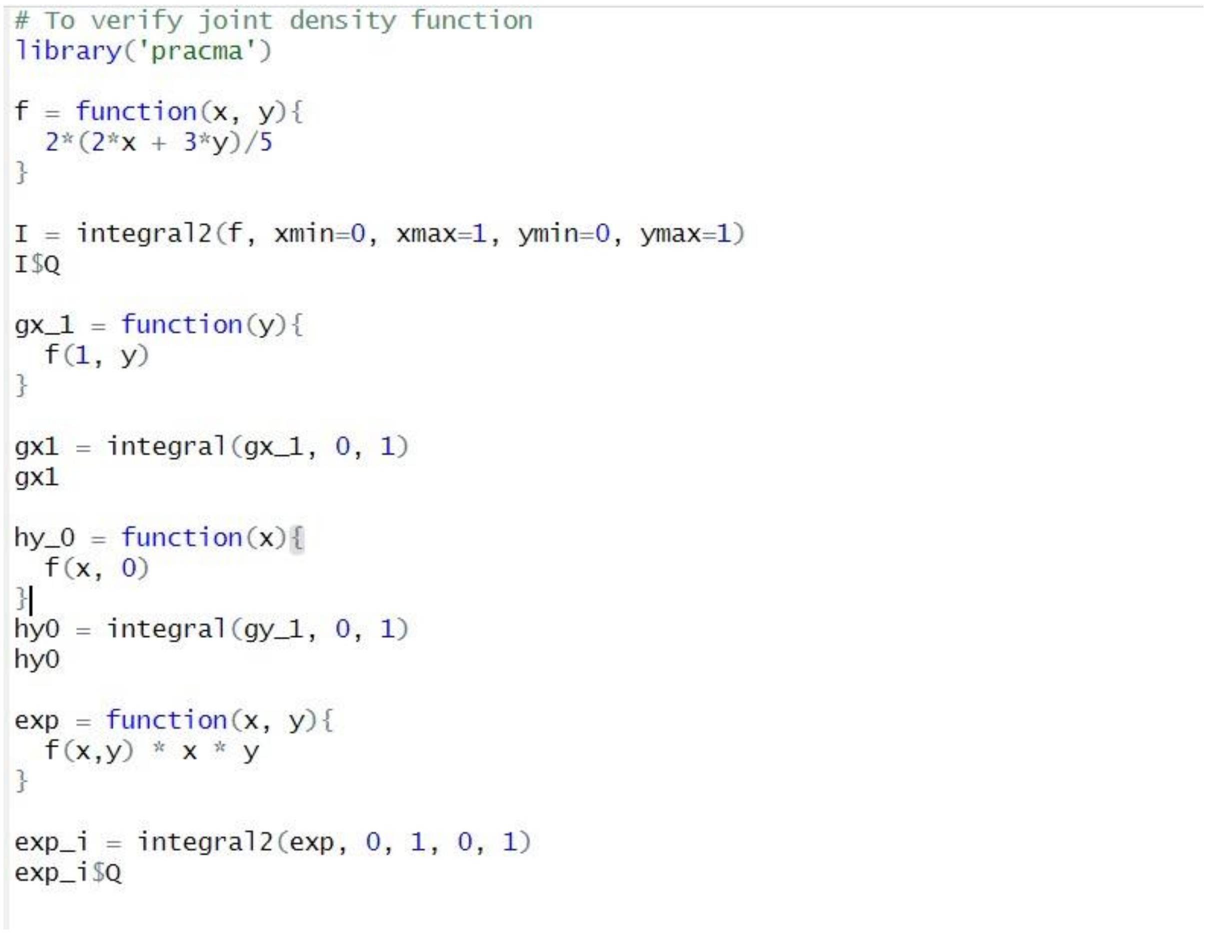


```
> # Parameters of the Gamma distribution
> alpha <- 2  # Shape parameter
> beta <- 1/3  # Scale parameter
>
> # (a) Probability that lifetime is at least 1 unit of time
> prob_at_least_1 <- 1 - pgamma(1, shape = alpha, rate = beta)
> cat("Probability that lifetime is at least 1 unit of time:", prob_at_least_1, "\n")
Probability that lifetime is at least 1 unit of time: 0.9553751
>
> # (b) Finding the value of c such that P(X ≤ c) ≥ 0.70
> target_prob <- 0.70
> c <- qgamma(target_prob, shape = alpha, rate = beta)
> cat("Value of c:", c, "\n")
Value of c: 7.317649
```

LAB 6

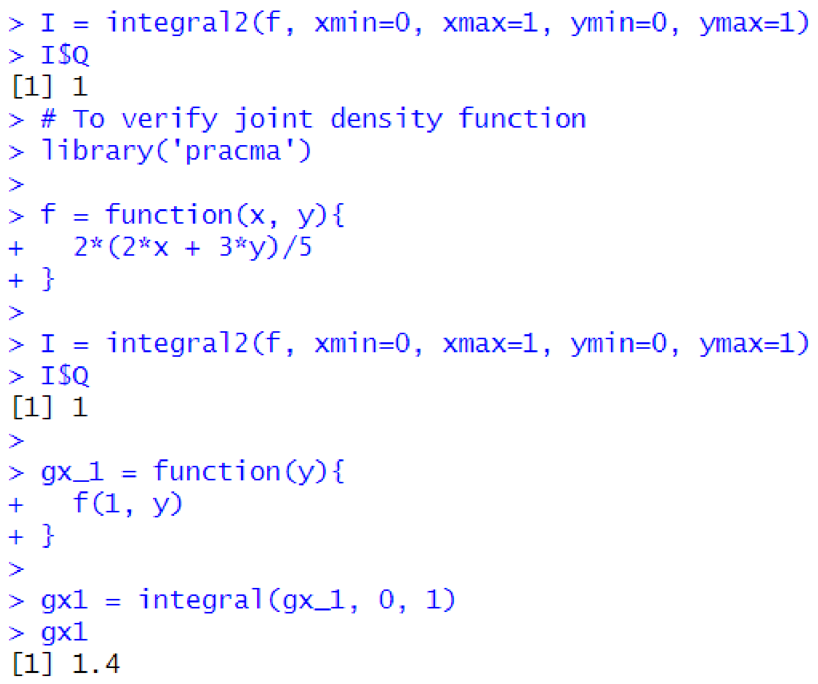
Q1 .

Code :

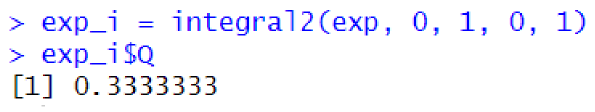


```
# To verify joint density function
library('pracma')

f = function(x, y){
  2*(2*x + 3*y)/5
}

I = integral2(f, xmin=0, xmax=1, ymin=0, ymax=1)
I$Q

gx_1 = function(y){
  f(1, y)
}

gx1 = integral(gx_1, 0, 1)
gx1

hy_0 = function(x){
  f(x, 0)
}
hy0 = integral(gy_1, 0, 1)
hy0

exp = function(x, y){
  f(x,y) * x * y
}

exp_i = integral2(exp, 0, 1, 0, 1)
exp_i$Q
```

Output :



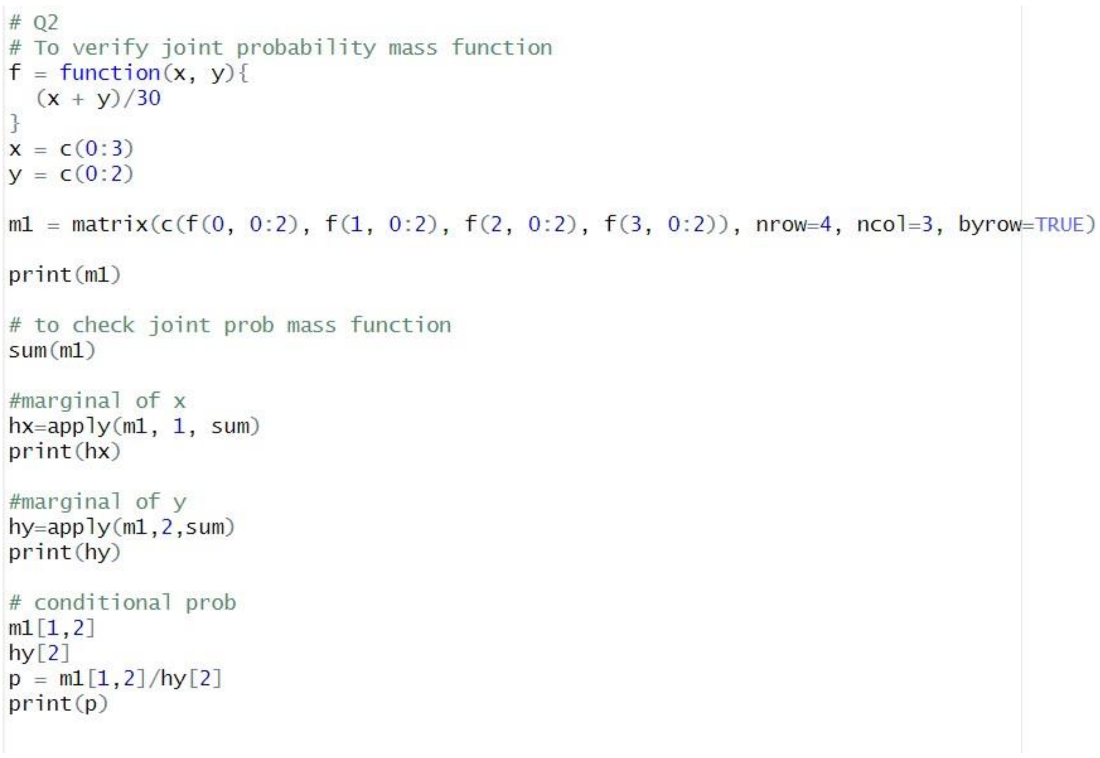
```
> I = integral2(f, xmin=0, xmax=1, ymin=0, ymax=1)
> I$Q
[1] 1
> # To verify joint density function
> library('pracma')
>
> f = function(x, y){
+   2*(2*x + 3*y)/5
+ }
>
> I = integral2(f, xmin=0, xmax=1, ymin=0, ymax=1)
> I$Q
[1] 1
>
> gx_1 = function(y){
+   f(1, y)
+ }
>
> gx1 = integral(gx_1, 0, 1)
> gx1
[1] 1.4
```



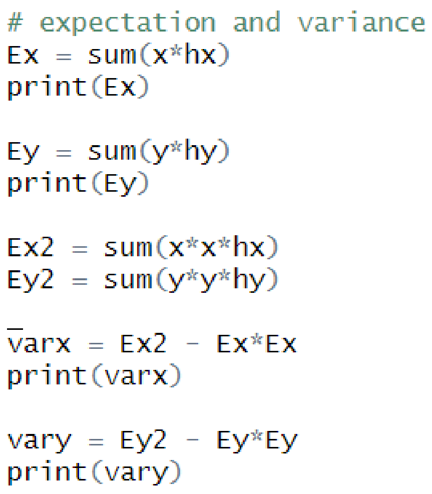
```
> exp_i = integral2(exp, 0, 1, 0, 1)
> exp_i$Q
[1] 0.3333333
```

Q2.

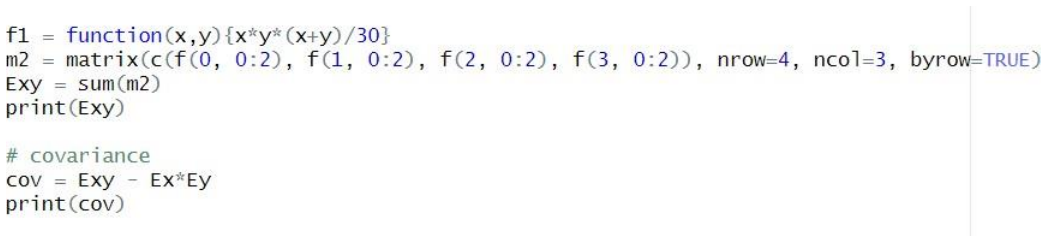
Code :



```
# Q2
# To verify joint probability mass function
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)

print(m1)

# to check joint prob mass function
sum(m1)

#marginal of x
hx=apply(m1, 1, sum)
print(hx)

#marginal of y
hy=apply(m1,2,sum)
print(hy)

# conditional prob
m1[1,2]
hy[2]
p = m1[1,2]/hy[2]
print(p)
```

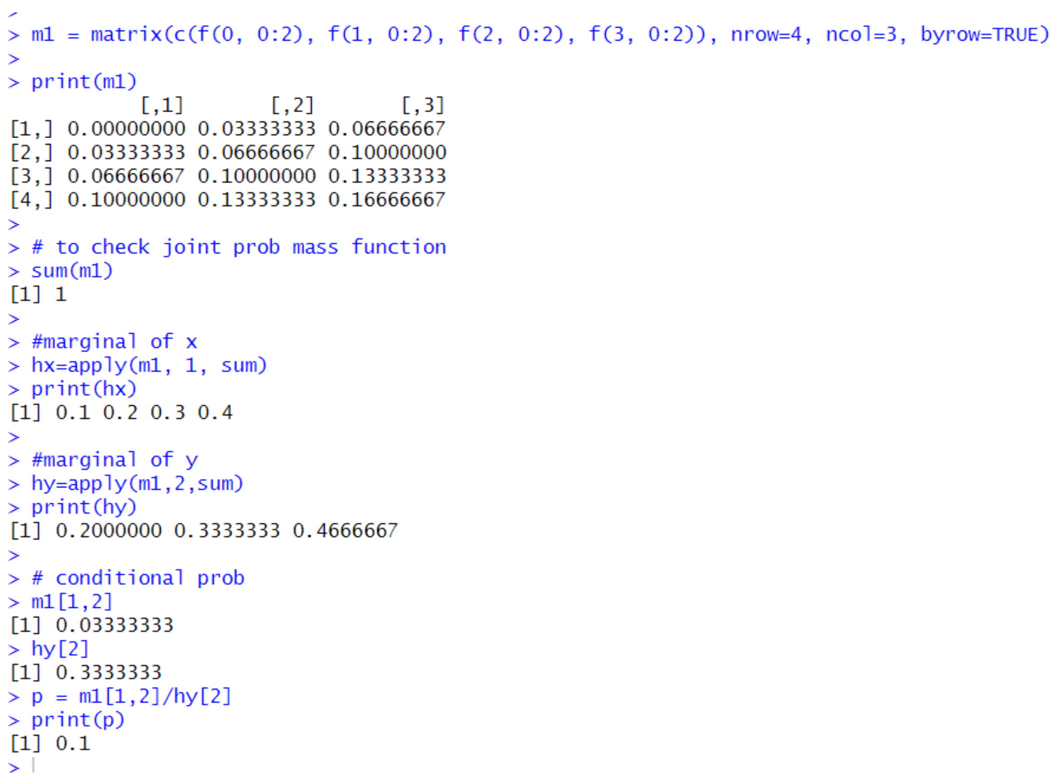


```
# expectation and variance
Ex = sum(x*hx)
print(Ex)

Ey = sum(y*hy)
print(Ey)

Ex2 = sum(x*x*hx)
Ey2 = sum(y*y*hy)

varx = Ex2 - Ex*Ex
print(varx)

vary = Ey2 - Ey*Ey
print(vary)
```

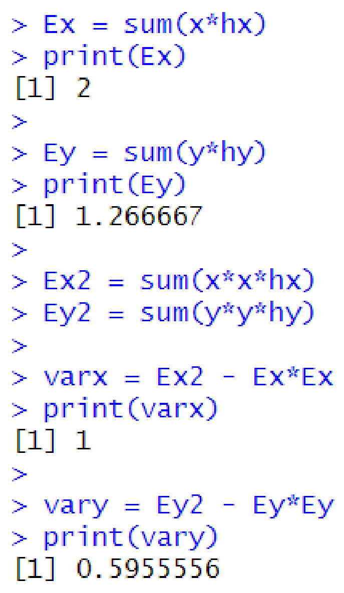


```
f1 = function(x,y){x*y*(x+y)/30}
m2 = matrix(c(f(0, 0:2), f(1, 0:2), f(2, 0:2), f(3, 0:2)), nrow=4, ncol=3, byrow=TRUE)
Exy = sum(m2)
print(Exy)

# covariance
cov = Exy - Ex*Ey
print(cov)
```

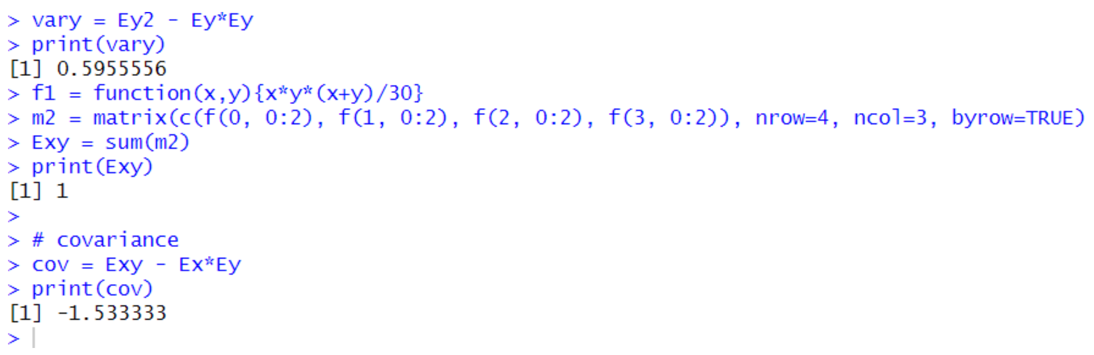
Output :



```
> 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)
>
> 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
>
> # to check joint prob mass function
> sum(m1)
[1] 1
>
> #marginal of x
> hx=apply(m1, 1, sum)
> print(hx)
[1] 0.1 0.2 0.3 0.4
>
> #marginal of y
> hy=apply(m1,2,sum)
> print(hy)
[1] 0.2000000 0.3333333 0.4666667
>
> # conditional prob
> m1[1,2]
[1] 0.03333333
> hy[2]
[1] 0.3333333
> p = m1[1,2]/hy[2]
> print(p)
[1] 0.1
> 
```



```
> Ex = sum(x*hx)
> print(Ex)
[1] 2
>
> Ey = sum(y*hy)
> print(Ey)
[1] 1.266667
>
> Ex2 = sum(x*x*hx)
> Ey2 = sum(y*y*hy)
>
> varx = Ex2 - Ex*Ex
> print(varx)
[1] 1
>
> vary = Ey2 - Ey*Ey
> print(vary)
[1] 0.5955556
```

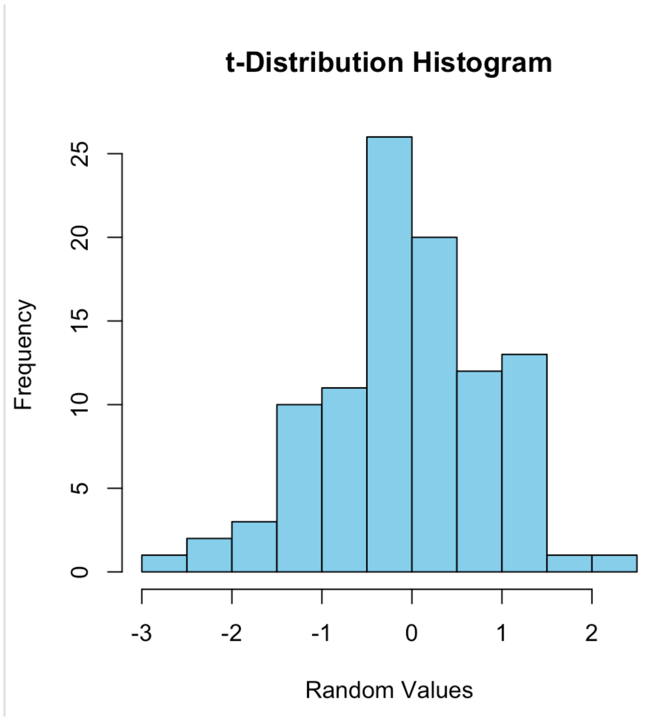


```
> vary = Ey2 - Ey*Ey
> print(vary)
[1] 0.5955556
> f1 = function(x,y){x*y*(x+y)/30}
> m2 = matrix(c(f(0, 0:2), f(1, 0:2), f(2, 0:2), f(3, 0:2)), nrow=4, ncol=3, byrow=TRUE)
> Exy = sum(m2)
> print(Exy)
[1] 1
>
> # covariance
> cov = Exy - Ex*Ey
> print(cov)
[1] -1.533333
> 
```

LAB 7

Q1
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.

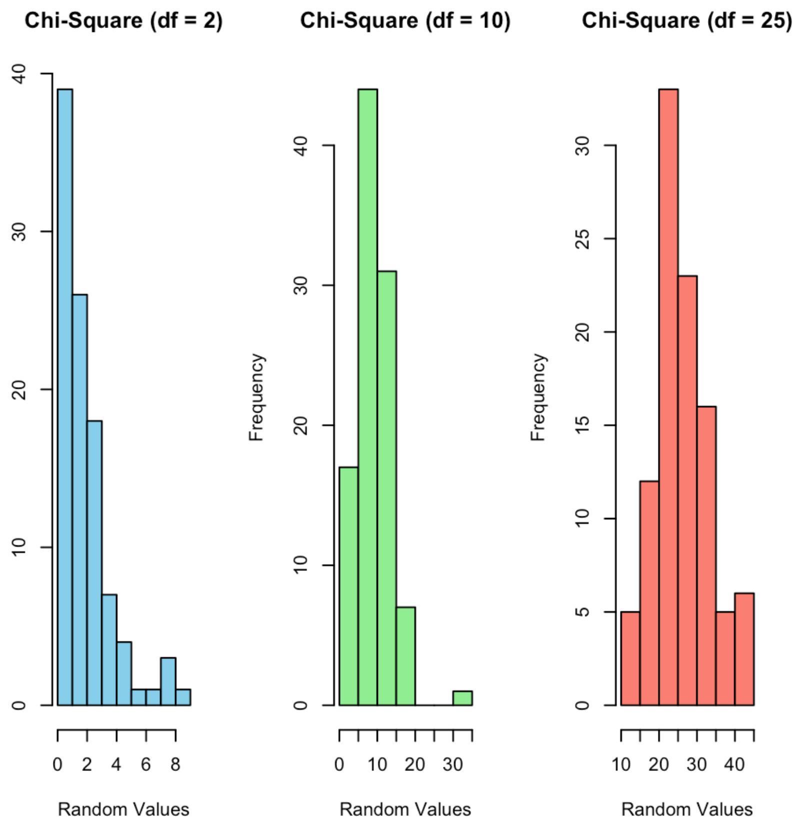
Code:
```
n <- 100
df <- n  -  1

t_distribution <- rt(n, df)

hist(t_distribution, main = "t-Distribution Histogram", xlab = "Random Values", col =
"skyblue")
```



Q2
Use the rchisq(n, df) function in r to investigate the chi-square distribution with n = 100 and df = 2, 10, 25.

Code:
```
n <- 100
```

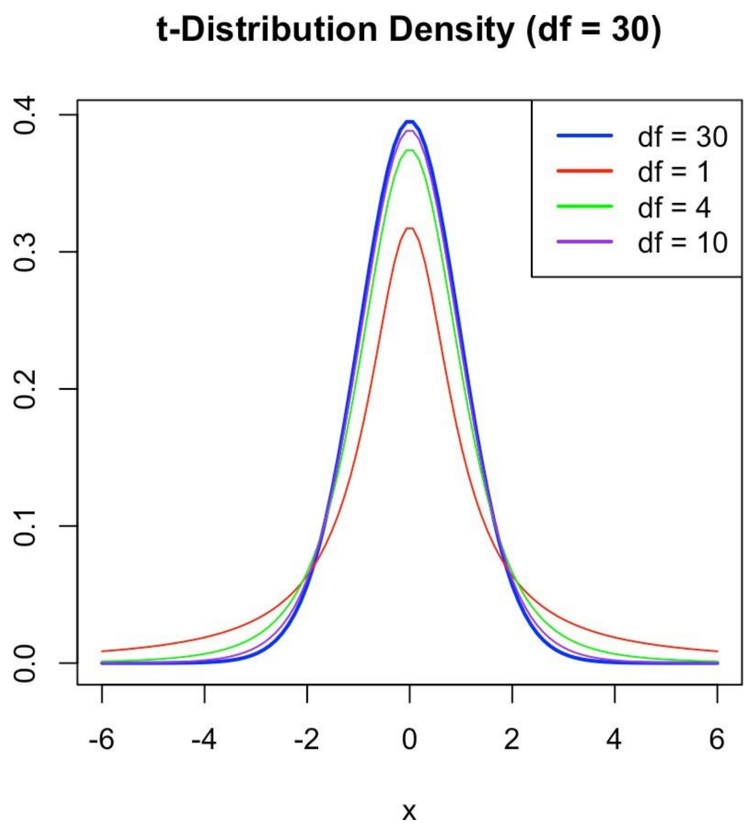
```
degrees_of_freedom <- c(2, 10, 25)

chi_square_2 <- rchisq(n, degrees_of_freedom[1])
chi_square_10 <- rchisq(n, degrees_of_freedom[2])
chi_square_25 <- rchisq(n, degrees_of_freedom[3])

par(mfrow = c(1, 3)) # Set up a 1x3 grid for the plots

hist(chi_square_2, main = "Chi-Square (df = 2)", xlab = "Random Values", col = "skyblue")
hist(chi_square_10, main = "Chi-Square (df = 10)", xlab = "Random Values", col =
"lightgreen")
hist(chi_square_25, main = "Chi-Square (df = 25)", xlab = "Random Values", col =
"salmon")

par(mfrow = c(1, 1))
```



Q3
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:
```
x <- seq(-6, 6, length.out = 100)

t_dist_df1 <- dt(x, df = 1)
```

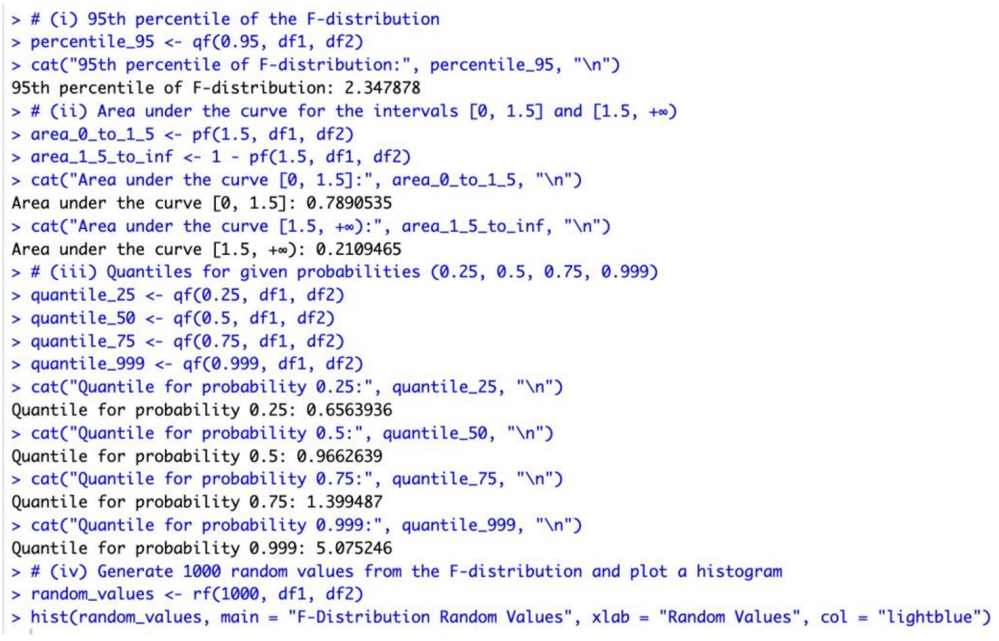
```
t_dist_df4 <- dt(x, df = 4)
t_dist_df10 <- dt(x, df = 10)
t_dist_df30 <- dt(x, df = 30)

plot(x, t_dist_df30, type = "l", col = "blue", lwd = 2, xlab = "x", ylab = "Density", main = "t-
Distribution Density (df = 30)")

lines(x, t_dist_df1, col = "red")
lines(x, t_dist_df4, col = "green")
lines(x, t_dist_df10, col = "purple")

legend("topright", legend = c("df = 30", "df = 1", "df = 4", "df = 10"), col = c("blue", "red",
"green", "purple"), lwd = 2)
```



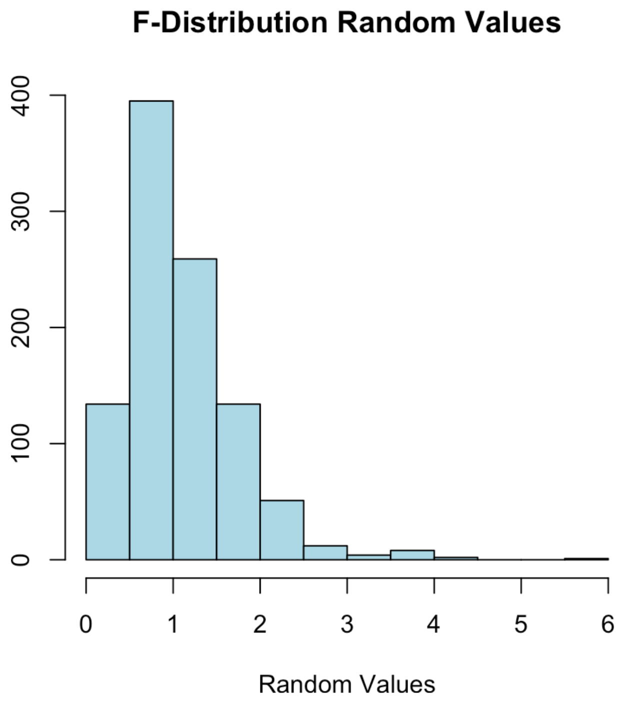
Q4
Write a r-code
(i) To find the 95th percentile of the F-distribution with (10, 20) degrees of freedom.
(ii) To calculate the area under the curve for the interval [0, 1.5] and the interval [1.5, +∞) of a F-curve with v1 = 10 and v2 = 20 (USE pf()).
(iii) To calculate the quantile for a given area (= 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.

<u>Code:</u>

```r
# Parameters for the F-distribution
df1 <- 10
df2 <- 20

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

# (ii) Area under the curve for the intervals [0, 1.5] and [1.5, +∞)
area_0_to_1_5 <- pf(1.5, df1, df2)
area_1_5_to_inf <- 1 - pf(1.5, df1, df2)
cat("Area under the curve [0, 1.5]:", area_0_to_1_5, "\n")
cat("Area under the curve [1.5, +∞):", area_1_5_to_inf, "\n")

# (iii) Quantiles for given probabilities (0.25, 0.5, 0.75, 0.999)
quantile_25 <- qf(0.25, df1, df2)
quantile_50 <- qf(0.5, df1, df2)
quantile_75 <- qf(0.75, df1, df2)
quantile_999 <- qf(0.999, df1, df2)
cat("Quantile for probability 0.25:", quantile_25, "\n")
cat("Quantile for probability 0.5:", quantile_50, "\n")
cat("Quantile for probability 0.75:", quantile_75, "\n")
cat("Quantile for probability 0.999:", quantile_999, "\n")

# (iv) Generate 1000 random values from the F-distribution and plot a histogram
random_values <- rf(1000, df1, df2)
hist(random_values, main = "F-Distribution Random Values", xlab = "Random Values", col
= "lightblue")
```



```
> # (i) 95th percentile of the F-distribution
> percentile_95 <- qf(0.95, df1, df2)
> cat("95th percentile of F-distribution:", percentile_95, "\n")
95th percentile of F-distribution: 2.347878
> # (ii) Area under the curve for the intervals [0, 1.5] and [1.5, +∞)
> area_0_to_1_5 <- pf(1.5, df1, df2)
> area_1_5_to_inf <- 1 - pf(1.5, df1, df2)
> cat("Area under the curve [0, 1.5]:", area_0_to_1_5, "\n")
Area under the curve [0, 1.5]: 0.7890535
> cat("Area under the curve [1.5, +∞):", area_1_5_to_inf, "\n")
Area under the curve [1.5, +∞): 0.2109465
> # (iii) Quantiles for given probabilities (0.25, 0.5, 0.75, 0.999)
> quantile_25 <- qf(0.25, df1, df2)
> quantile_50 <- qf(0.5, df1, df2)
> quantile_75 <- qf(0.75, df1, df2)
> quantile_999 <- qf(0.999, df1, df2)
> cat("Quantile for probability 0.25:", quantile_25, "\n")
Quantile for probability 0.25: 0.6563936
> cat("Quantile for probability 0.5:", quantile_50, "\n")
Quantile for probability 0.5: 0.9662639
> cat("Quantile for probability 0.75:", quantile_75, "\n")
Quantile for probability 0.75: 1.399487
> cat("Quantile for probability 0.999:", quantile_999, "\n")
Quantile for probability 0.999: 5.075246
> # (iv) Generate 1000 random values from the F-distribution and plot a histogram
> random_values <- rf(1000, df1, df2)
> hist(random_values, main = "F-Distribution Random Values", xlab = "Random Values", col = "lightblue")
```


F-Distribution Random Values
0
100
200
300
400
0
1
2
3
4
5
6
Random Values

LAB 8

#step 1

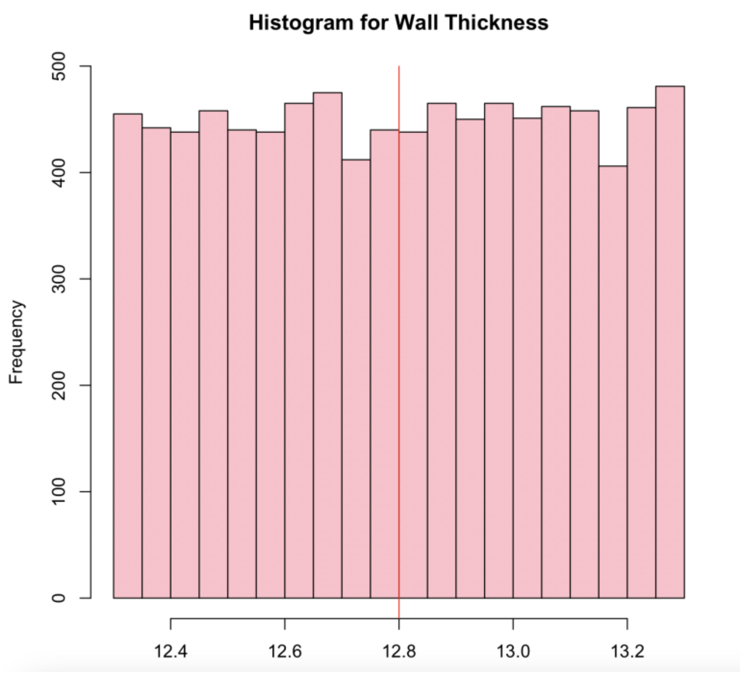
```
> data<-íead.csv( le.choose())
```

#Step 2 - Validate data foí coííectness

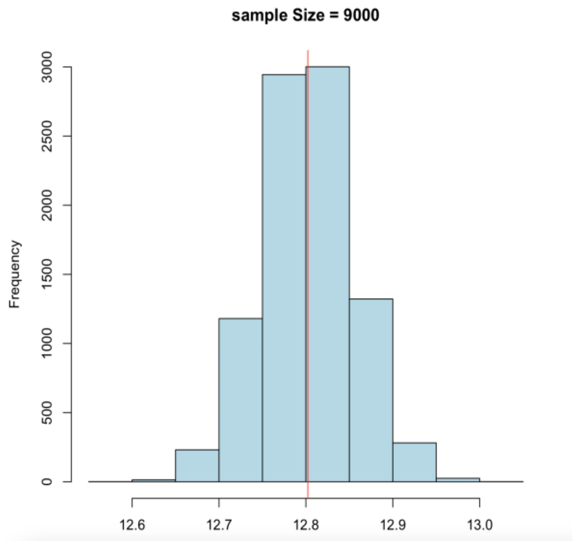
```
> #___________________
>
> #Count of Rows and columns
> dim(data)
[1] 9000    1
> #View top 10 íows of the dataset
> head(data,10)
  Wall.Thickness
1       12.35487
2       12.61742
3       12.36972
4       13.22335
5       13.15919
6       12.67549
7       12.36131
8       12.44468
9       12.62977
10       12.90381
```

#Step 3 - Calculate the population mean and plot the obseívations

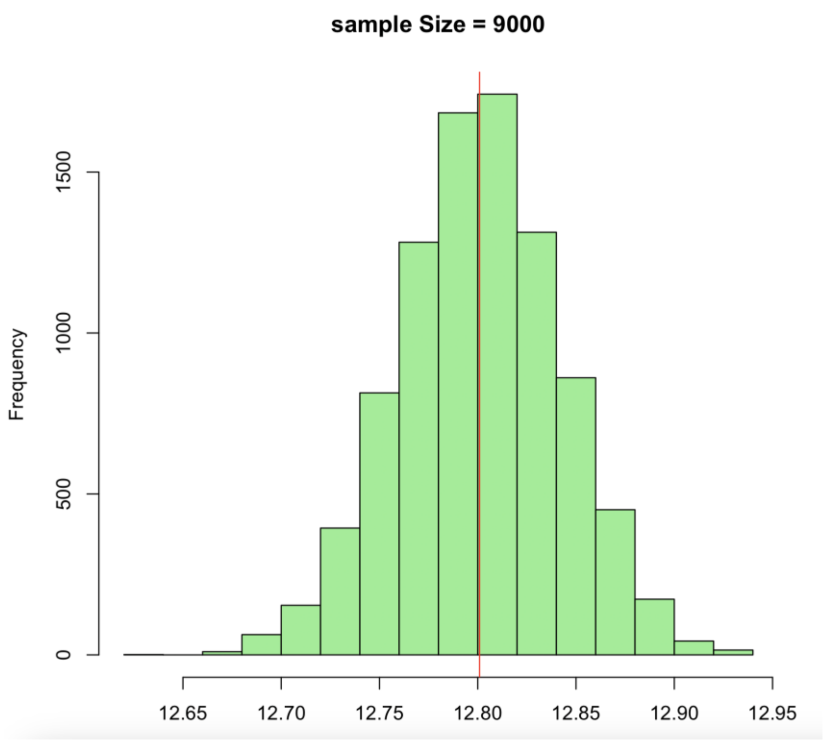
```
> #________________________
>
> #Calculate the population mean
> mean(data$Wall.Thickness)
[1] 12.80205
> #Calculate the population mean
> mean(data$Wall.Thickness)
[1] 12.80205
> #Plot all the obseívations in the data
> hist(data$Wall.Thickness,col = "pink",main = "Histogíam foí Wall Thickness",xlab
= "wall thickness")
> abline(v=12.8,col="íed",lty=1)
```



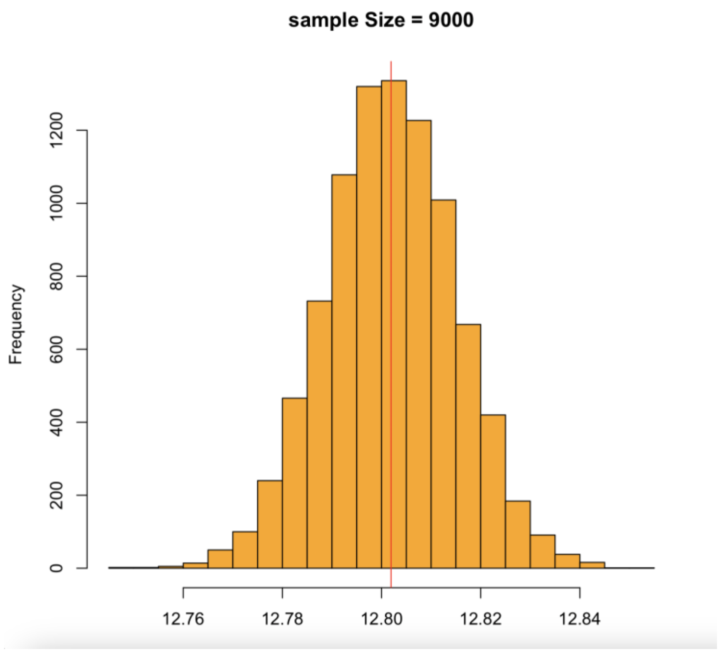
```
> s30<-c()
> s50<-c()
> s500<-c()
> n=9000
> foí(i in 1:n){
+   s30[i]=mean(sample(data$Wall.Thickness,30, íeplace=TRUE))
+   s50[i]=mean(sample(data$Wall.Thickness,50, íeplace=TRUE))
+  s500[i]=mean(sample(data$Wall.Thickness,500, íeplace=TRUE))}
> hist(s30,col="lightblue", main="sample Size = 9000",xlab="wall THickness")
> abline(v=mean(s30),col="íed")
```



```
> hist(s50,col="lightgíeen", main="sample Size = 9000",xlab="wall THickness")
> abline(v=mean(s50),col="íed")
```



```
> hist(s500,col="oíange", main="sample Size = 9000",xlab="wall THickness")
> abline(v=mean(s500),col="íed")
```


sample Size = 9000
Frequency