# PROBABILITY AND STATISTICS LAB REPORT


**Submitted By:**

Name - Rimjhim Mittal

Roll number -102103430

Batch - 3COE16


**Submitted To:**

Dr. Rajanish Rai

July 2023 – December 2023

Assignment1

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

```
c <- c(5, 10, 15, 20, 25, 30)

max_value <- max(c)
min_value <- min(c)

print(paste("Maximum:", max_value))
print(paste("Minimum:", min_value))

```

```
> print(paste("Maximum:", max_value))
[1] "Maximum: 30"
> print(paste("Minimum:", min_value))
[1] "Minimum: 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.

```
factorial_function <- function(n) {
  if(n < 0) {
    return("Error: Input number is negative!")
  } else if(n == 0) {
    return(1)
  } else {
    return(n * factorial_function(n-1))
  }
}

number <- as.integer(readline(prompt="Enter a number: "))
result <- factorial_function(number)
cat("The factorial of", number, "is:", result, "\n")

```

```
> number <- as.integer(readline(prompt="Enter a number: "))
Enter a number: 5
> result <- factorial_function(number)
> cat("The factorial of", number, "is:", result, "\n")
The factorial of 5 is: 120
>
```

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

```r
# Function to generate the first n terms of the Fibonacci sequence
generate_fibonacci <- function(n) {
  a <- 0
  b <- 1

  if (n < 1) {
    cat("Please enter a valid positive integer for n.\n")
    return(NULL)
  }
  cat("Fibonacci Sequence (First", n, "terms):")
  for (i in 1:n) {
    cat(" ", a)
    next_term <- a + b
    a <- b
    b <- next_term
  }
  cat("\n")
}

# Get input from the user for the number of terms (n)
n <- as.integer(readline(prompt = "Enter the number of Fibonacci terms (n): "))
generate_fibonacci(n)
```

```
> n <- as.integer(readline(prompt = "Enter the number of Fibona
Enter the number of Fibonacci terms (n): 7
> generate_fibonacci(n)
Fibonacci Sequence (First 7 terms):  0  1  1  2  3  5  8
>
```

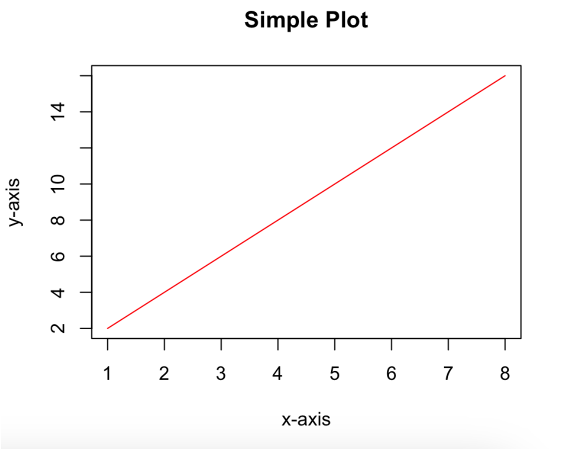
(4) Write an R program to make a simple calculator which can add, subtract, multiply and divide.

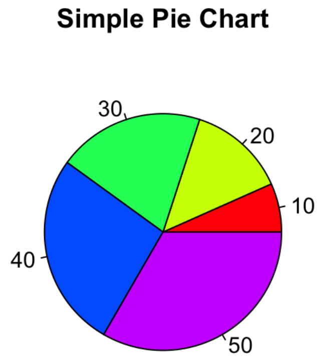
```r
add <- function(a, b) {
  return(a + b)
}
subtract <- function(a, b) {
  return(a - b)
}
multiply <- function(a, b) {
  return(a * b)
}
divide <- function(a, b) {
  if (b == 0) {
    cat("Error: Division by zero is not allowed.\n")
    return(NULL)
  }
  return(a / b)
}
cat("Simple Calculator\n")
cat("1. Addition\n")
cat("2. Subtraction\n")
cat("3. Multiplication\n")
cat("4. Division\n")
choice <- as.integer(readline("Enter your choice (1/2/3/4): "))

num1 <- as.numeric(readline("Enter the first number: "))
num2 <- as.numeric(readline("Enter the second number: "))
result <- NULL
if (choice == 1) {
  result <- add(num1, num2)
} else if (choice == 2) {
  result <- subtract(num1, num2)
} else if (choice == 3) {
  result <- multiply(num1, num2)
} else if (choice == 4) {
  result <- divide(num1, num2)
} else {
  cat("Invalid choice. Please select a valid operation (1/2/3/4).\n")
}
```

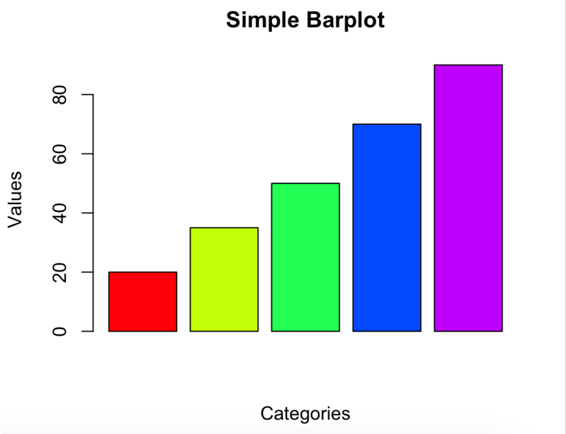
```
Result:  30
>
```

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

```r
x <- c(seq(from=1,to=8))
y <- c(seq(from=2,to=16,by=2))
colors <- rainbow(5)

plot(x, y, main="Simple Plot", xlab="x-axis", ylab="y-axis",pch=2,cex=3,col=colors,type="l")

pie_values <- c(10, 20, 30, 40, 50)
pie(pie_values, labels=pie_values, col=colors, main="Simple Pie Chart")

bar_values <- c(20, 35, 50, 70, 90)
barplot(bar_values, main="Simple Barplot", xlab="Categories", ylab="Values",col=colors)
```



ASSIGNMENT2

(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)
(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)

~~~
#(a) Sample space for drawing 10 coins:
coins <- c(rep("gold", 20), rep("silver", 30), rep("bronze", 50))
sample(coins, 10, replace=TRUE)

```
~~~

~~~
 [1] "silver" "bronze" "silver" "silver" "bronze" "silver" "gold"   "silver" "silver" "gold"
~~~

~~~
#(b) Sample space for surgical procedures:
outcomes <- c(rep("success", 9), "failure")
sample(outcomes, 10, replace=TRUE)

```
~~~

~~~
 [1] "success" "success" "success" "success" "failure" "success" "success" "success" "success" "success"
~~~

(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?

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

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

~~~~r
#R simulation for birthday paradox
simulate_birthday <- function(n, num_simulations=10000) {
  matches <- 0
  for(i in 1:num_simulations) {
    birthdays <- sample(1:365, n, replace=TRUE)
    if(length(unique(birthdays)) < n) matches <- matches + 1
  }
  return(matches / num_simulations)
}

# Test for n=5 to n=30 (adjust as needed)
for(n in 5:77)
  cat("For", n, "people, estimated probability:", simulate_birthday(n), "\n")

```
~~~~

```
For 20 people, estimated probability: 0.4132
For 21 people, estimated probability: 0.4394
For 22 people, estimated probability: 0.4905
For 23 people, estimated probability: 0.5054
For 24 people, estimated probability: 0.5362
For 25 people, estimated probability: 0.5718
For 26 people, estimated probability: 0.5969
For 27 people, estimated probability: 0.626
For 28 people, estimated probability: 0.661
For 29 people, estimated probability: 0.6796
For 30 people, estimated probability: 0.7146
For 31 people, estimated probability: 0.729
For 32 people, estimated probability: 0.7453
For 33 people, estimated probability: 0.7805
For 34 people, estimated probability: 0.7936
For 35 people, estimated probability: 0.8137
```

N=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?

````
```{r}
# Function to compute conditional probability
conditional_probability <- function(prob_cloudy, prob_rain, prob_cloudy_given_rain) {
  prob_rain_given_cloudy <- (prob_cloudy_given_rain * prob_rain) / prob_cloudy
  return(prob_rain_given_cloudy)
}

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

# Calculate the conditional probability
prob_rain_given_cloudy <- conditional_probability(prob_cloudy, prob_rain, prob_cloudy_given_rain)
cat("Conditional Probability of Rain given Cloudy:", prob_rain_given_cloudy, "\n")
```
````

```
Conditional Probability of Rain given 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:

(a) Print first few rows of this dataset.

(b) Find the structure of this dataset.

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

(d) Find the mean of the sepal length.

(e) Find the median of the sepal length.

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

(g) Find the standard deviation and variance.

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

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

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

# (a) Print first few rows
head(iris)

# (b) Structure of the dataset
str(iris)

# (c) Range of sepal length
range_sepal_length <- range(iris$Sepal.Length)
print("Range of Sepal Length:")
print(range_sepal_length)

# (d) Mean of sepal length
mean_sepal_length <- mean(iris$Sepal.Length)
print("Mean of Sepal Length:")
print(mean_sepal_length)

# (e) Median of sepal length
median_sepal_length <- median(iris$Sepal.Length)
print("Median of Sepal Length:")
print(median_sepal_length)

# (f) First and third quartiles and interquartile range
quartiles_sepal_length <- quantile(iris$Sepal.Length, c(0.25, 0.75))
iqr_sepal_length <- diff(quartiles_sepal_length)
print("First Quartile:")
print(quartiles_sepal_length[1])
print("Third Quartile:")
print(quartiles_sepal_length[2])
print("Interquartile Range:")
print(iqr_sepal_length)

# (g) Standard deviation and variance
std_dev_sepal_length <- sd(iris$Sepal.Length)
variance_sepal_length <- var(iris$Sepal.Length)
print("Standard Deviation of Sepal Length:")
print(std_dev_sepal_length)
print("Variance of Sepal Length:")
print(variance_sepal_length)

# Repeat (c) to (g) for other attributes: sepal.width, petal.length, and petal.width

# (h) Summary of the dataset
summary(iris)
```

| | Sepal.Length <dbl> | Sepal.Width <dbl> | Petal.Length <dbl> | Petal.Width <dbl> | Species <fctr> |
|---|---|---|---|---|---|
| 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 |

6 rows

(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 takes the vector as input and gives the mode value as output.

````
```{r}
calculate_mode <- function(x) {
  uniq_x <- unique(x)
  counts <- table(x)
  mode_val <- uniq_x[which.max(counts)]
  return(mode_val)
}

# Example usage:
data <- c(1, 2, 2, 3, 3, 3, 4, 4, 5)
mode_result <- calculate_mode(data)
cat("Mode of the dataset:", mode_result, "\n")

```
````

```
Mode of the dataset: 3
```

ASSIGNMENT 3

#q1-- P(9)-P(6) for 7 to 9
nine<-pbinom(9,size=12,prob=1/6) #we give the probability of one success
six<-pbinom(6,size=12,prob=1/6)
result<-nine-six
cat("Probability of getting 7,8 or 9 sixes",result)

#q2
pnorm(84, mean=72, sd=15.2, lower.tail = FALSE) #false so that we get value higher than 84 i.e. 84 and more
#or
x<-1-(pnorm(84, mean=72, sd=15.2))
x

#q3 poisson distribution
dpois(0, lambda=5) #we use 0 because no car arrives i.e. x value is 0

ppois(50,lambda=50)-ppois(47,lambda=50) # lambda will be 5*10 as there is 10 hour time slot

```
```{r}
size<- 12
prob<- 1/6
print(pbinom(9, size, prob) - pbinom(6, size, prob))
```
```

```
[1] 0.001291758
```

```
```{r}
1- pnorm(84, mean = 72, sd = 15.2)
```
```

```
[1] 0.2149176
```

```
```{r}
a <- dpois(0, lambda = 5)
ans<-dpois(48, lambda = 50)+dpois(49, lambda = 50)+dpois(50, lambda = 50)
ans
```
```

```
[1] 0.1678485
```

#q4 hypergeometric distribution

dhyper(3,m=17,n=233,k=5)   #m is no. of defective (which we know), n is no. of non-defective(which we calculated),

```{r}
dhyper(3,m=17,n=233,k=5)   #m is no. of defective (
```

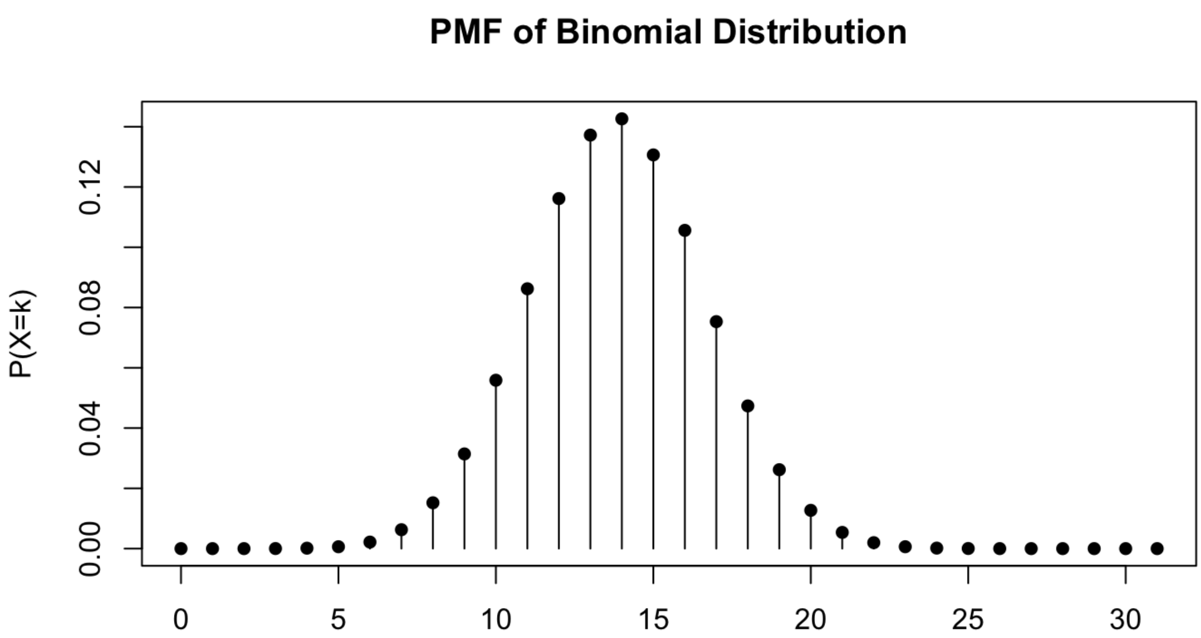
```
[1] 0.002351153
```

```
#q5
#(a) binomial distribution
#(b)
x<-seq(0,31)
pmf<-c()
for(i in 1:length(x)){
  pmf[i]<-dbinom(x[i], size=31, prob=0.447)
}

#(c)
y<-seq(0,31)
pmf<-c()
for(i in 1:length(y)){
  pmf[i]<-pbinom(y[i], size=31, prob=0.447)
}
#(d) mean=n*p, var=n*p*q=n*p*(1-p), sd=sqrt(var)
mean<-31*0.447
mean
var<-31*0.447*(1-0.447)
var
sd<-sqrt(var)
sd
```

```{r}
n <- 31
p <- 0.447
k <- 0:n
pmf <- dbinom(k, n, p)
plot(k, pmf, type="h", main="PMF of Binomial Distribution", xlab="k", ylab="P(X=k)", xlim=c(0,n
points(k, pmf, pch=16)

```



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:

```
x_values <- c(0, 1, 2, 3, 4)
p_x_values <- c(0.41, 0.37, 0.16, 0.05, 0.01)

expected_value1 <- sum(x_values * p_x_values)

expected_value2 <- weighted.mean(x_values, p_x_values)

expected_value3 <- c(x_values %*% p_x_values)

expected_value1
expected_value2
expected_value3
```

Output:

```
> x_values <- c(0, 1, 2, 3, 4)
> p_x_values <- c(0.41, 0.37, 0.16, 0.05, 0.01)
>
> expected_value1 <- sum(x_values * p_x_values)
>
> expected_value2 <- weighted.mean(x_values, p_x_values)
>
> expected_value3 <- c(x_values %*% p_x_values)
>
> expected_value1
[1] 0.88
> expected_value2
[1] 0.88
> expected_value3
[1] 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:

```
f_t <- function(t) {
  return(t * 0.1 * exp(-0.1 * t))
}

result <- integrate(f_t, lower = 0, upper = Inf)

expected_value_T <- result$value

expected_value_T
```

Output:

```
> f_t <- function(t) {
+   return(t * 0.1 * exp(-0.1 * t))
+ }
>
> result <- integrate(f_t, lower = 0, upper = Inf)
>
> expected_value_T <- result$value
>
> expected_value_T
[1] 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:

```
x_values <- c(0, 1, 2, 3)
p_x_values <- c(0.1, 0.2, 0.2, 0.5)

y_values <- 10 * x_values - 12

expected_value_Y <- sum(y_values * p_x_values)

expected_value_Y
```

Output:

```
> x_values <- c(0, 1, 2, 3)
> p_x_values <- c(0.1, 0.2, 0.2, 0.5)
>
> y_values <- 10 * x_values - 12
>
> expected_value_Y <- sum(y_values * p_x_values)
>
> expected_value_Y
[1] 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.
($k$th moment = $E(X^k)$, Mean = first moment and Variance = second moment – Mean$^2$.

Code:

```
f_x_first_moment <- function(x) {
  return(x * 0.5 * exp(-abs(x)))
}

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

first_moment <- integrate(f_x_first_moment, lower = 1, upper = 10)$value

second_moment <- integrate(f_x_second_moment, lower = 1, upper =
10)$value
mean_X <- first_moment

variance_X <- second_moment - mean_X^2

first_moment
second_moment
mean_X
variance_X
```

Output

```
>
> first_moment
[1] 0.3676297
> second_moment
[1] 0.9169292
> mean_X
[1] 0.3676297
> variance_X
[1] 0.7817776
>
>
```

5.

Let X be a geometric random variable with probability distribution

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

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:

```
pdf3<-function(x){
  (0.75*((0.25)^(x-1)))
}
probablevaluesofy<-c(1,4,9,16,25)
mean_y<-0
variance_y<-0
for(i in 1:length(probablevaluesofy)){
  y<-probablevaluesofy[i]
  probability_y<-pdf3(sqrt(y))

  mean_y=mean_y+(y*(probability_y))
  variance_y=variance_y+((y^2)*(probability_y))#secondmoment

}
variance_y=variance_y-(mean_y^2)
print(mean_y)
print(variance_y)
pdf3(3)
```

Output:

```
+   mean_y=mean_y+(y*(probability_y).
+   variance_y=variance_y+((y^2)*(pr
+
+ }
> variance_y=variance_y-(mean_y^2)
> print(mean_y)
[1] 2.182617
> print(variance_y)
[1] 7.614112
> pdf3(3)
[1] 0.046875
>
```

**ASSIGNMENT-5**

```
#Experiment 5

# Q1
a=punif(45,0,60,lower.tail = FALSE)
b=punif(30,0,60)-punif(20,0,60)
cat("Waiting time lies more than 45 minutes : ",a)
cat("Waiting time lies between 20 and 30 minutes : ",b)

# Q2

c=dexp(3,1/2)
cat("Value of density function at x=3 is : ",c)

x<-seq(0,5,by=0.02)   #0<=x<=5
px<-dexp(x,rate=1/2)
plot(x,px)

d=pexp(3,1/2)        #x<=3
cat("Prob that repair time takes atmost 3 hours is : ",d)

x<-seq(0,5,by=0.02)
px<-pexp(x,rate=1/2)
plot(x,px)

sample=rexp(1000,rate=0.5)        # r: array of random samples of prob values
plot(sample)
plot(density(sample))

# Q3

ans= pgamma(1,shape=2,scale=1/3,lower.tail=FALSE)   #x>=1
cat("Prob that lifetime of equipment is atleast 1 unit of time : ",ans)

ans2= qgamma(0.70,shape=2,scale=1/3)   #x<=c>=0.70
cat("value of c : ",ans2)

#Experiment 6

# Q1

install.packages('pracma')
library('pracma')
ft=function(x,y){
```

```
  2*(2*x+3*y)/5
}
i=integral2(ft,xmin=0,xmax=1,ymin=0,ymax=1)
print(i)
```

**ASSIGNMENT-6**

**Q1.**

```
library('pracma')
f<-function(x,y){
 return (2*(2*x+3*y)/5)};

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

g<-function(y){
  f(1,y)};
gx<-integral(g,0,1);
gx

h<-function(x){
  f(x,0)};
hx<-integral(g,0,1);
hx

e<-function(x,y){
  (x*y)*f(x,y)
}
ex<-integral2(e,xmin=0,xmax=1,ymin=0,ymax=1);
ex$Q
```

```
> install.packages('pracma')
Installing package into ‘C:/Users/CSED/AppData/Local/R/win-library/4.2’
(as ‘lib’ is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.2/pracma_2.4.2.zip'
Content type 'application/zip' length 1726565 bytes (1.6 MB)
downloaded 1.6 MB

package ‘pracma’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\CSED\AppData\Local\Temp\RtmpgHRsbi\downloaded_packages
```

```
> library('pracma')
> f<-function(x,y){
+  return (2*(2*x+3*y)/5)};
>
> I<-integral2(f,xmin=0,xmax=1,ymin=0,ymax=1);
> I$Q
[1] 1
>
> g<-function(y){
+    f(1,y)};
> gx<-integral(g,0,1);
> gx
[1] 1.4
>
```

```
> h<-function(x){
+    f(x,0)};
> hx<-integral(g,0,1);
> hx
[1] 1.4
>
> e<-function(x,y){
+    (x*y)*f(x,y)
+ }
> ex<-integral2(e,xmin=0,xmax=1,ymin=0,ymax=1);
> ex$Q
[1] 0.3333333
>
```

**Q2.**

```
f<-function(x,y){
  (x+y)/30;
}
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);
M1
sum(M1)

gx<-apply(M1,1,sum);
hy<-apply(M1,2,sum);

p<-M1[1,2]/hy[2]
p
```

gx
hy

```
> f<-function(x,y){
+   (x+y)/30;
+ }
> 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);
> 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
> sum(M1)
[1] 1
>
```

```
> gx<-apply(M1,1,sum);
> hy<-apply(M1,2,sum);
>
> p<-M1[1,2]/hy[2]
> p
[1] 0.1
>
> gx
[1] 0.1 0.2 0.3 0.4
> hy
[1] 0.2000000 0.3333333 0.4666667
```

#expectd value of x
Ex <- sum(x*gx)
Ey <- sum(y*hy)
vx <- sum(((x-Ex)^2)*gx)
vy <- sum(((y-Ey)^2)*hy)
Exy <- sum(outer(x,y)*M1)
Ex
Ey
vx
vy
Exy

```
> Ex
[1] 2
> Ey
[1] 1.266667
> vx
[1] 1
> vy
[1] 0.5955556
> Exy
[1] 2.4
> 
```

**ASSIGNMENT-7**

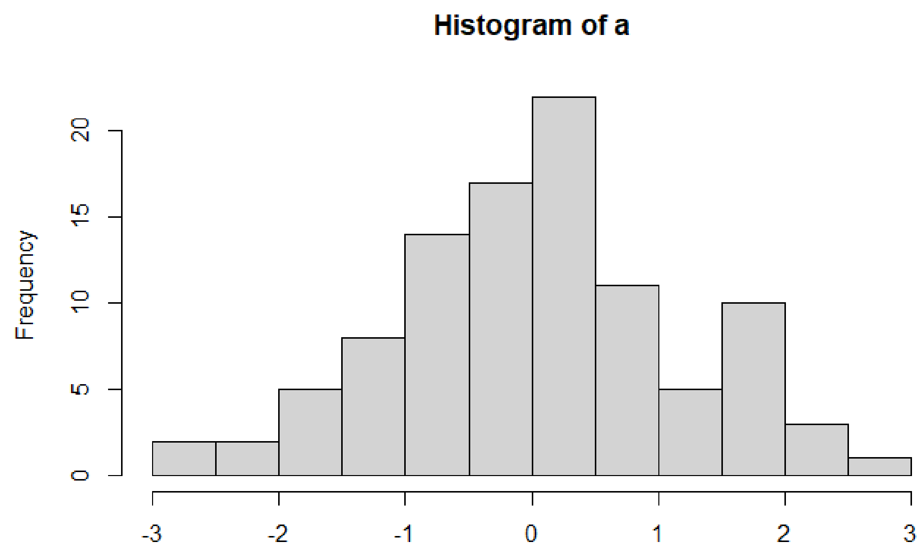
#Degree of Freedom = 0,1,2
#DOF = 1 => t-dist., Chi-dist.
#DOF = 2 => F-dist.

# 1
n = 100
df = n-1

a = rt(n,df)
a

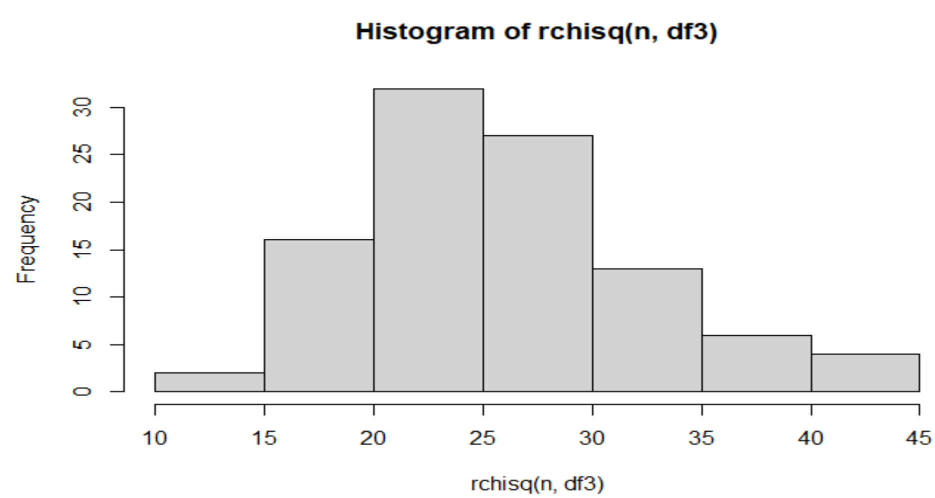
hist(a)

```
> n = 100
> df = n-1
> 
> a = rt(n,df)
> a
  [1]  1.39332019  0.03180591 -1.18600945  0.62623858  0.20838742  0.56848402 -0.32500402 -0.50328136  0.81175213  1.47924271  0.04856383
 [12]  2.11651176 -0.42481306  0.17957238 -0.43175402  0.32990356  1.59133850 -1.01092749 -0.11526643 -0.45085815  2.22903464 -0.75066221
 [23]  0.12177895  1.13101739 -1.02212162  1.57886617  1.58691583 -2.59656612  0.47177346  0.46198426  0.54720265  0.44559177 -0.11450803
 [34] -0.99133872 -0.86005907  0.05668013  2.04522151  1.10912779 -1.00716479 -0.82021365 -0.78940677  0.47248357  0.29500732  2.62610745
 [45] -1.79548234  0.06787310  0.10153243 -0.93679874 -0.01864863 -1.50852495  1.95684485 -0.23934590  1.94580553 -2.13457179 -1.24646384
 [56]  1.55038380 -0.86673401  0.57992464  0.86084090 -0.49547637 -0.67086311  1.54625078 -0.35864589 -0.95687044  0.40991182 -1.00780009
 [67] -1.63676540  0.36963992  1.66066995 -1.40298319 -0.32160832 -0.79021188 -2.71654457 -2.11748689  0.99964930 -0.01946425  0.05859012
 [78]  1.25104356  0.04188284 -1.02640907  0.95171479 -0.08293738 -0.76753552  0.39677752  0.60156622  1.93213842  0.72905973  1.52089522
 [89]  0.04887256  0.63315410 -0.10796615 -0.05388835 -0.52065821 -0.06286029 -1.82706876 -0.83349144 -1.87219917 -0.37989099  0.29736610
[100]  0.40247798
> 
> hist(a)
> 
```




# 2
n = 100
df1 = 2
df2 = 10
df3 = 25

hist(rchisq(n,df1))
hist(rchisq(n,df2))
hist(rchisq(n,df3))



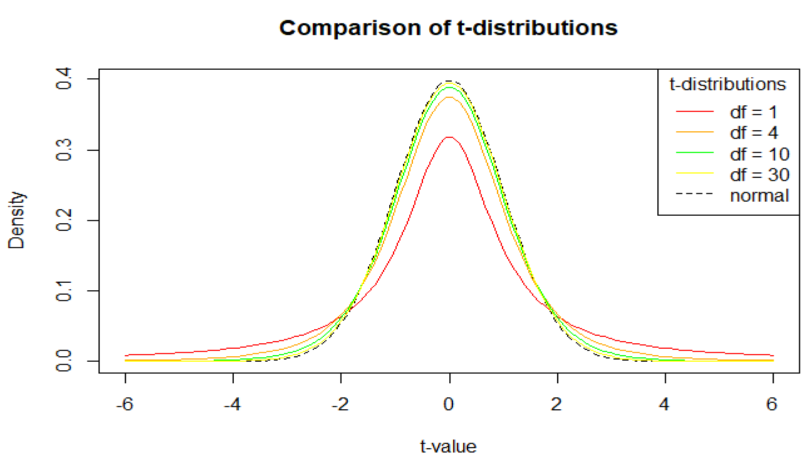

# 3
x = seq(-6,6,length.out = 100)
# Generate a vector of 100 values between -6 and 6
x <- seq(-6, 6, length = 100)

```
# Degrees of freedom
df = c(1,4,10,30)
colour = c("red", "orange", "green", "yellow","black")

# Plot a normal distribution
plot(x, dnorm(x), type = "l", lty = 2, xlab = "t-value", ylab = "Density",
          main = "Comparison of t-distributions", col = "black")

# Add the t-distributions to the plot
for (i in 1:4){
lines(x, dt(x, df[i]), col = colour[i])
}

# Add a legend
legend("topright", c("df = 1", "df = 4", "df = 10", "df = 30", "normal"),
          col = colour, title = "t-distributions", lty = c(1,1,1,1,2))
```




```
# 4
# a
df1 = 10
df2 = 20
alpha = 0.05

per = qf(1-alpha,df1,df2)
per
```

```
> df1 = 10
> df2 = 20
> alpha = 0.05
>
> per = qf(1-alpha,df1,df2)
> per
[1] 2.347878
```

```
# b
df1 = 10
df2 = 20
area1 = pf(1.5,df1,df2)
area2 = 1 - area1

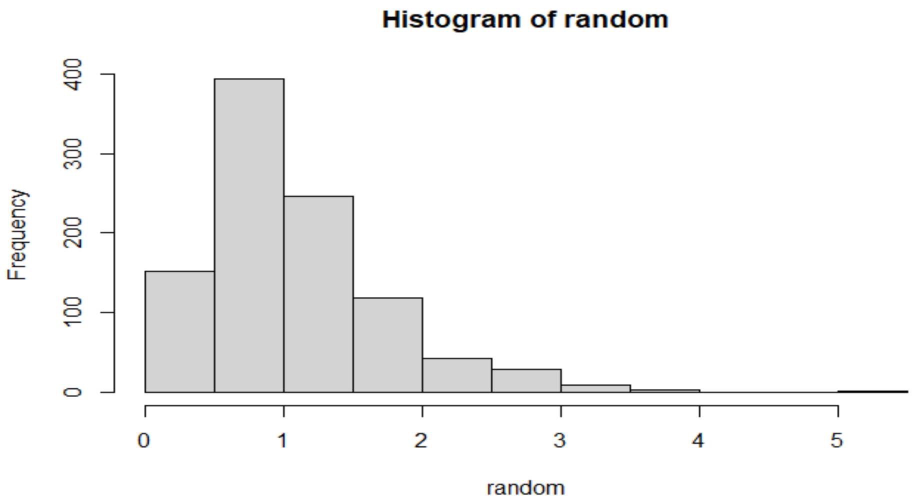
area1
area2
```

```
> df1 = 10
> df2 = 20
> area1 = pf(1.5,df1,df2)
> area2 = 1 - area1
>
> area1
[1] 0.7890535
> area2
[1] 0.2109465
>
```

```
# c
df1 = 10
df2 = 20
quant = c(0.25,0.5,0.75,0.999)
result = qf(quant,df1,df2)
print(result)
```

```
> df1 = 10
> df2 = 20
> quant = c(0.25,0.5,0.75,0.999)
> result = qf(quant,df1,df2)
> print(result)
[1] 0.6563936 0.9662639 1.3994874 5.0752462
>
```

```
# d
df1 = 10
df2 = 20
random = rf(1000,df1,df2)
hist(random)
```



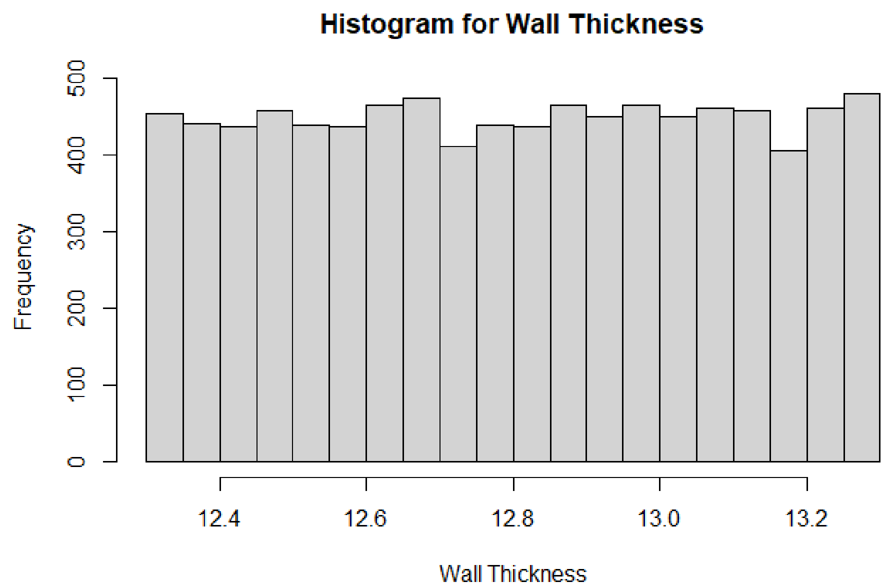

**ASSIGNMENT-8**

# 1
# a
df = read.csv(file.choose())
df

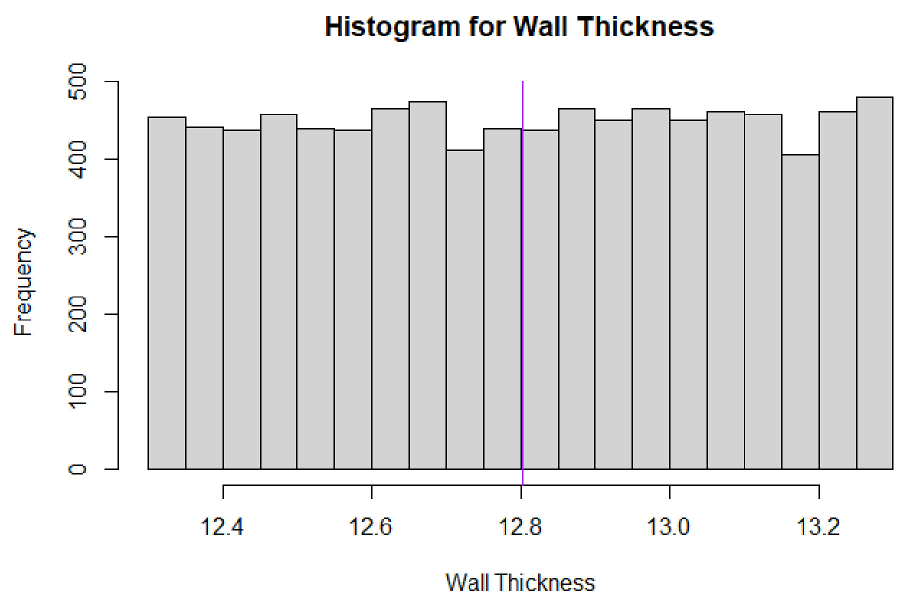
# b
nrow(df)
head(df,10)

```
> nrow(df)
[1] 9000
> head(df,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
>
```

# c
mean = mean(df$Wall.Thickness)
hist(df$Wall.Thickness,main="Histogram for Wall Thickness",xlab="Wall Thickness")




# d
abline(v = mean, col = "purple")




# 2

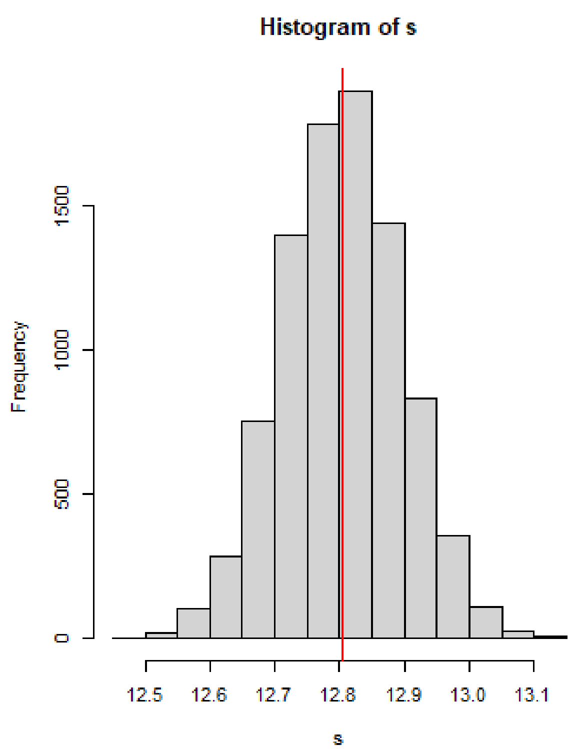
# a
n = 9000
s = c()
s1 = c()
s2 = c()

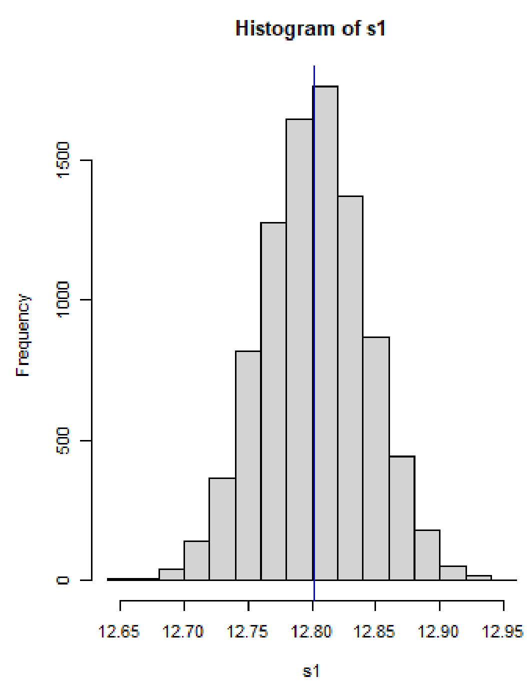
```
i = 0

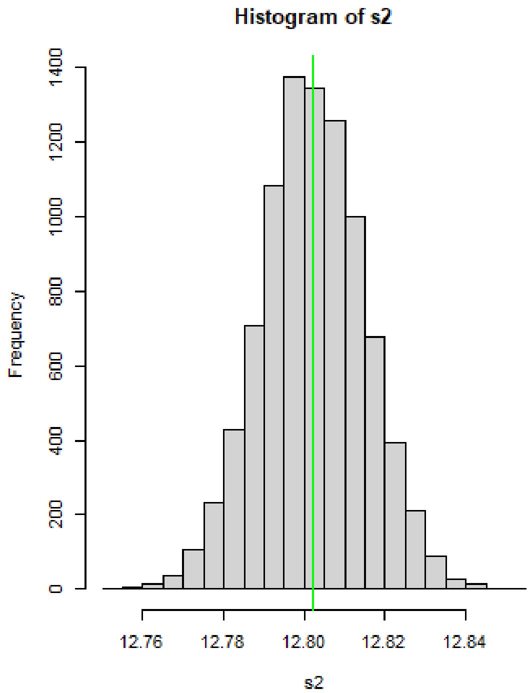
for (i in 1:n){
s[i] = mean(sample(df$Wall.Thickness,10,replace=T))
s1[i] = mean(sample(df$Wall.Thickness,50,replace=T))
s2[i] = mean(sample(df$Wall.Thickness,500,replace=T))
}

# b
par(mfrow  = c(1,3))

hist(s)
abline(v=mean(s), col = 'red')
hist(s1)
abline(v=mean(s1), col = 'blue')
hist(s2)
abline(v=mean(s2), col = 'green')
```




Histogram of s2
Frequency
0
200
400
600
800
1000
1200
1400
12.76
12.78
12.80
12.82
12.84
s2