Namrah Sharma

3CS3

Roll No: 102217054

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 maxôimum and minimum of this vector.

```
> # Create a vector with a few numbers
> numbers <- c(5, 10, 15, 20, 25, 30)
>
> # Finding the max and min values in the vector
> max_value <- max(numbers)
> min_value <- min(numbers)
>
> # Displaying the results
> cat("The maximum value is:", max_value, "\n")
The maximum value is: 30
> cat("The minimum value is:", min_value, "\n")
The minimum value 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.

```
> input <- readline("Enter a number: ")
Enter a number: 5
> num <- as.integer(input)
>
> if (is.na(num)) {
+     cat("Please enter a valid integer.\n")
+ } else if (num < 0) {
+     cat("Error: Factorial is not defined for negative numbers.\n")
+ } else {
+     fact <- 1
+     for (i in 1:num) {
+         fact <- fact * i
+     }
+     cat("The factorial of", num, "is", fact, "\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.

```
> n <- as.integer(readline("Enter the number of terms: "))
Enter the number of terms: 8
> if (is.na(n) || n <= 0) {
+     cat("Please enter a positive integer.\n")
+ } else {
+     fibonacci <- numeric(n)
+     if (n >= 1) fibonacci[1] <- 0
+     if (n >= 2) fibonacci[2] <- 1
+
+     for (i in 3:n) {
+         fibonacci[i] <- fibonacci[i - 1] + fibonacci[i - 2]
+     }
+
+     cat("The first", n, "terms of the Fibonacci sequence are:\n")
+     print(fibonacci)
+ }
The first 8 terms of the Fibonacci sequence are:
[1]  0  1  1  2  3  5  8 13
>
>
```

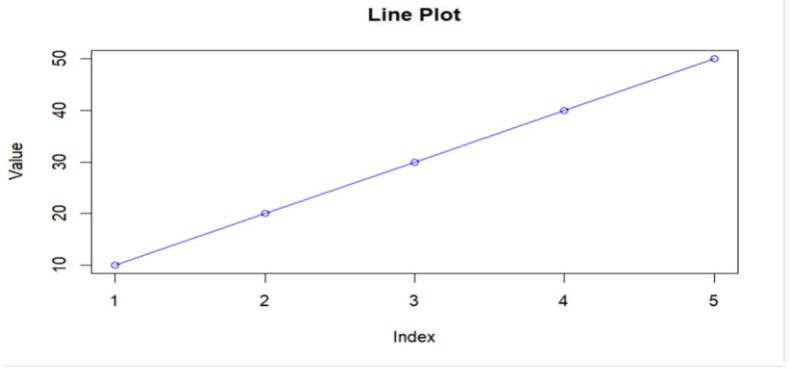
(4) Write an R program to make a simple calculator which can add, subtract, multiply and divide.

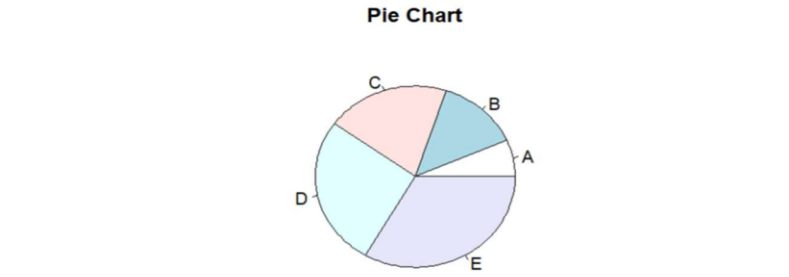
```
> cat("Simple Calculator\n")
Simple Calculator
> cat("1. Add\n")
1. Add
> cat("2. Subtract\n")
2. Subtract
> cat("3. Multiply\n")
3. Multiply
> cat("4. Divide\n")
4. Divide
>
> choice <- as.integer(readline("Enter choice (1/2/3/4): "))
Enter choice (1/2/3/4): 3
>
> if (choice %in% 1:4) {
+     num1 <- as.numeric(readline("Enter first number: "))
+     num2 <- as.numeric(readline("Enter second number: "))
+
+     result <- switch(choice,
+                      `1` = num1 + num2,
+                      `2` = num1 - num2,
+                      `3` = num1 * num2,
+                      `4` = if (num2 != 0) num1 / num2 else "Error: Division by zero")
+
+     cat("Result:", result, "\n")
+ } else {
+     cat("Invalid choice.\n")
+ }
Enter first number: 2
Enter second number: 3
Result: 6
>
```

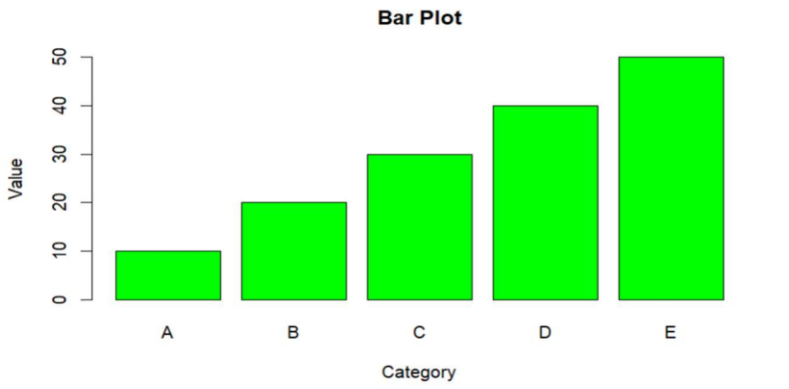
(5) Explore plot, pie, barplot etc. (the plotting options) which are built-in functions in R.

```r
# Sample data
data <- c(10, 20, 30, 40, 50)

# Plot
plot(data, type = "o", col = "blue", xlab = "Index", ylab = "Value", main = "Line Plot")

# Pie chart
pie(data, labels = c("A", "B", "C", "D", "E"), main = "Pie Chart")

# Bar plot
barplot(data, names.arg = c("A", "B", "C", "D", "E"), col = "green", xlab = "Category", ylab = "Value", main = "Bar Plot")
```

```
> # Sample data
> data <- c(10, 20, 30, 40, 50)
> # Plot
> plot(data, type = "o", col = "blue", xlab = "Index", ylab = "Value", main = "Line Plot")
>
```







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)

```
> chest <- c(rep("Gold", 20), rep("Silver", 30), rep("Bronze", 50))
> sample_space <- sample(chest, 10)
> print(sample_space)
 [1] "Bronze" "Bronze" "Bronze" "Silver" "Silver" "Silver" "Silver" "Gold"   "Gold"   "Bronze"
> 
```

(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)

```
> chest <- c(rep("Gold", 20), rep("Silver", 30), rep("Bronze", 50))
> sample_space <- sample(chest, 10)
> print(sample_space)
 [1] "Bronze" "Bronze" "Bronze" "Silver" "Silver" "Silver" "Silver" "Gold"   "Gold"   "Bronze"
> probabilities <- c(Success = 0.9, Failure = 0.1)
> 
> 
> sample_space <- sample(names(probabilities), size = 10, replace = TRUE, prob = probabilities)
> print(sample_space)
 [1] "Success" "Success" "Success" "Success" "Success" "Success" "Success" "Success" "Success"
[10] "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.

```
> simulate_birthday_prob <- function(n, N = 10000) {
+   count <- 0
+   for (i in 1:N) {
+     birthdays <- sample(1:365, n, replace = TRUE)
+     if (any(duplicated(birthdays))) {
+       count <- count + 1
+     }
+   }
+   return(count / N)
+ }
> 
> 
> probabilities <- sapply(1:30, simulate_birthday_prob)
> print(probabilities)
 [1] 0.0000 0.0030 0.0097 0.0183 0.0295 0.0425 0.0581 0.0745 0.0983 0.1250 0.1413 0.1680 0.1898
[14] 0.2203 0.2516 0.2810 0.3157 0.3413 0.3726 0.4144 0.4392 0.4680 0.5043 0.5491 0.5690 0.6026
[27] 0.6230 0.6453 0.6755 0.7100
> 
```

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

```
> m <- 1
> prob <- 0
> while (prob <= 0.5) {
+   m <- m + 1
+   prob <- simulate_birthday_prob(m)
+ }
> cat("Smallest n for probability > 0.5 is", m, "\n")
Smallest n for probability > 0.5 is 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 probôability 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?

```
> bayes_theorem <- function(pA, pB, pBA) {
+   pAB <- (pBA * pA) / pB
+   return(pAB)
+ }
>
> pRain <- 0.2
> pCloudy <- 0.4
> pCloudyRain <- 0.85
>
> result <- bayes_theorem(pRain, pCloudy, pCloudyRain)
> print(result)
[1] 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.

```
> data(iris)
>
> # (a) Print the first few rows
> 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.

```
> # (b) Find the structure
> 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.

```
> # (c) Range of sepal length
> range(iris$Sepal.Length)
[1] 4.3 7.9
> |
```

(d) Find the mean of the sepal length.

```
> # (d) Mean of sepal length
> mean(iris$Sepal.Length)
[1] 5.843333
> |
```

(e) Find the median of the sepal length.

```
> # (e) Median of sepal length
> median(iris$Sepal.Length)
[1] 5.8
>
```

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

```
> # (f) Quartiles and interquartile range
> quartiles <- quantile(iris$Sepal.Length, probs = c(0.25, 0.75))
> IQR <- diff(quartiles)
> quartiles
25% 75%
5.1 6.4
> IQR
75%
1.3
> |
```

(g) Find the standard deviation and variance.

```
> # (g) Standard deviation and variance
> sd(iris$Sepal.Length)
[1] 0.8280661
> var(iris$Sepal.Length)
[1] 0.6856935
> # (g) Standard deviation and variance
> sd(iris$Sepal.Length)
[1] 0.8280661
> var(iris$Sepal.Length)
[1] 0.6856935
> |
```

(h) Try doing the above exercises for

sepal.width

```
> # (h) Repeat for other attributes
> # Sepal Width
> range(iris$Sepal.Width)
[1] 2.0 4.4
> mean(iris$Sepal.Width)
[1] 3.057333
> median(iris$Sepal.Width)
[1] 3
> quartiles <- quantile(iris$Sepal.Width, probs = c(0.25, 0.75))
> IQR <- diff(quartiles)
> quartiles
25% 75%
2.8 3.3
> IQR
75%
0.5
> sd(iris$Sepal.Width)
[1] 0.4358663
> var(iris$Sepal.Width)
[1] 0.1899794
>
```

petal.length

```
> # Petal Length
> range(iris$Petal.Length)
[1] 1.0 6.9
> mean(iris$Petal.Length)
[1] 3.758
> median(iris$Petal.Length)
[1] 4.35
> quartiles <- quantile(iris$Petal.Length, probs = c(0.25, 0.75))
> IQR <- diff(quartiles)
> quartiles
25% 75%
1.6 5.1
> IQR
75%
3.5
> sd(iris$Petal.Length)
[1] 1.765298
> var(iris$Petal.Length)
[1] 3.116278
>
```

petal.width.

```
> # Petal Width
> range(iris$Petal.Width)
[1] 0.1 2.5
> mean(iris$Petal.Width)
[1] 1.199333
> median(iris$Petal.Width)
[1] 1.3
> quartiles <- quantile(iris$Petal.Width, probs = c(0.25, 0.75))
> IQR <- diff(quartiles)
> quartiles
25% 75%
0.3 1.8
> IQR
75%
1.5
> sd(iris$Petal.Width)
[1] 0.7622377
> var(iris$Petal.Width)
[1] 0.5810063
>
```

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

```
> # (i) Summary of the dataset
> 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
>
```

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.

```
> get_mode <- function(v) {
+   uniq_values <- unique(v)
+   mode_value <- uniq_values[which.max(tabulate(match(v, uniq_values)))]
+   return(mode_value)
+ }
>
> b <- c(5, 6, 5, 6, 5)
> mode_b <- get_mode(b)
> cat("Mode of vector b:", mode_b, "\n")
Mode of vector b: 5
>
```