(B) Write a R program to create a Data Frame which contain details of 5 employees and display summary of the data using R.

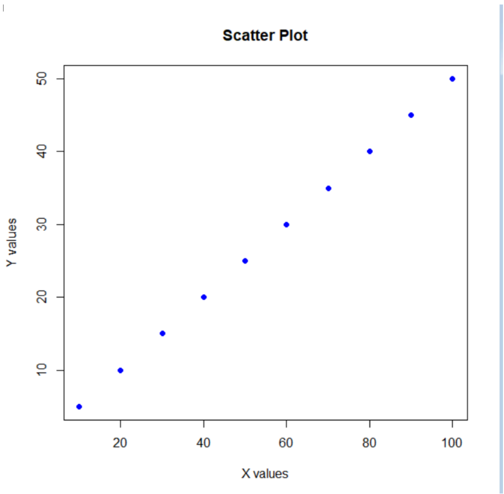
```r
employee_data <- data.frame(
    EmpID = c(101, 102, 103, 104, 105),
    EmpName= c("Alice", "Bob", "Carol", "Doe", "John"),
    EmpAge= c(30, 28, 35, 29, 31),
    Department= c("HR", "Finance", "IT", "Marketing", "Engineering"),
    Salary= c(60000, 75000, 85000, 70000, 95000)
)

print("Employee Data: ")
print(employee_data)

print("Summary: ")
summary(employee_data)
```

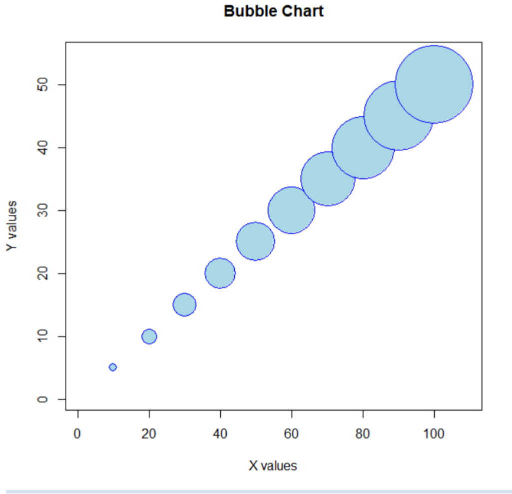
```
>
> print("Employee Data: ")
[1] "Employee Data: "
> print(employee_data)
  EmpID EmpName EmpAge  Department Salary
1   101   Alice     30          HR  60000
2   102     Bob     28     Finance  75000
3   103   Carol     35          IT  85000
4   104     Doe     29   Marketing  70000
5   105    John     31 Engineering  95000
>
> print("Summary: ")
[1] "Summary: "
> summary(employee_data)
     EmpID        EmpName              EmpAge        Department       
 Min.   :101   Length:5           Min.   :28.0   Length:5          
 1st Qu.:102   Class :character   1st Qu.:29.0   Class :character  
 Median :103   Mode  :character   Median :30.0   Mode  :character  
 Mean   :103                      Mean   :30.6                     
 3rd Qu.:104                      3rd Qu.:31.0                     
 Max.   :105                      Max.   :35.0                     
     Salary     
 Min.   :60000  
 1st Qu.:70000  
 Median :75000  
 Mean   :77000  
 3rd Qu.:85000  
 Max.   :95000  
>
```

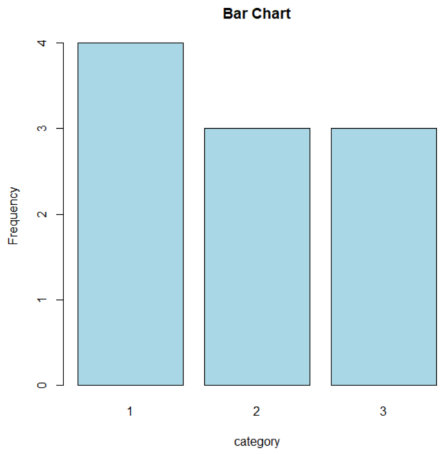
(B) For anydataset visialize the following types of chart : Scatterplot. Bubble Chart, Bar Chart , Dot Plots ,Histogram ,Box Plot ,Pie Chart

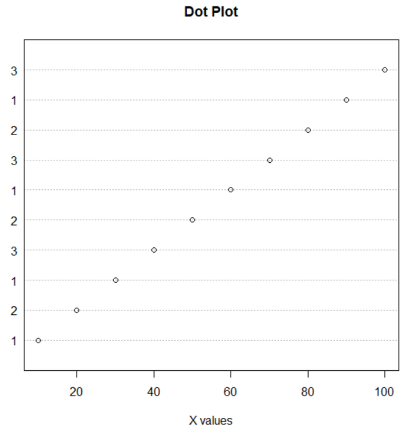
```
data <- data.frame(
  x = c(10,20,30,40,50,60,70,80,90,100),
  y = c(5, 10, 15, 20, 25, 30, 35, 40, 45, 50),
  size = c(1,2,3,4,5,6,7,8,9,10),
  category = c(1,2,1,3,2,1, 3,2,1,3),
  value = c(100,200,300,400,500,600,700,800,900,1000)
)

print(data)

# Scatterplot
plot(data$x, data$y,
  main = "Scatter Plot",
  xlab = "X values",
  ylab = "Y values",
  col = "blue",
  pch = 19
)

# Bubble Chart
symbols(data$x, data$y,
    circles = data$size,
    inches = 0.5,
    fg = 'blue',
    bg = 'lightblue',
    main = 'Bubble Chart',
    xlab = "X values",
    ylab = "Y values"
)

# Bar Plot
barplot(table(data$category),
  main = "Bar Chart",
  xlab = "category",
  ylab = "Frequency",
  col = "lightblue"
)
```

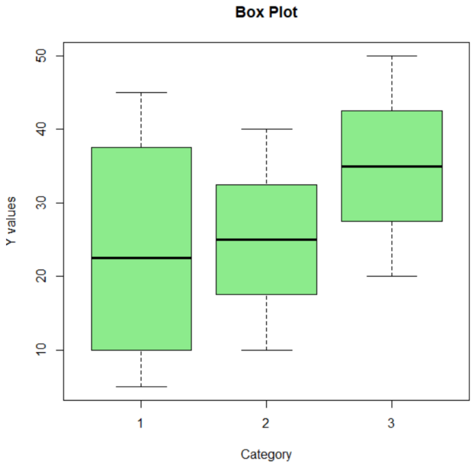
```r
# Dot Chart
dotchart(data$x,
  labels = data$category,
  main = "Dot Plot",
  xlab = "X values"
)

# Box Plot
boxplot(data$y~data$category,
  main = "Box Plot",
  xlab = "Category",
  ylab = "Y values",
  col = "lightgreen"
)

# Histogram
hist(data$x,
  main = "Histogram of X values",
  xlab = "X values",
  col = "lightblue",
  border = "black"
)

# pie chart
pie(data$value,
  labels = data$category,
  main = "Pie Chart",
  col = c("lightblue", "lightgreen", "lightpink", "yellow", "orange")
)
```

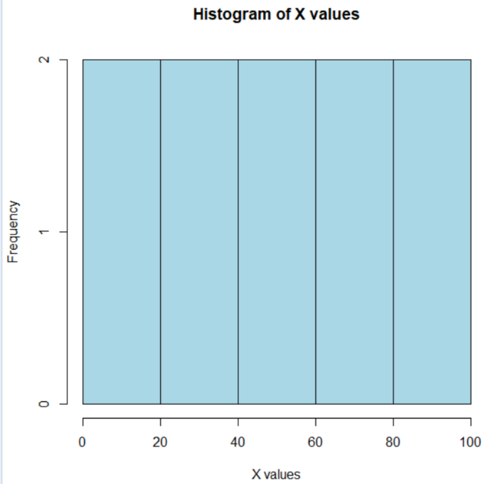


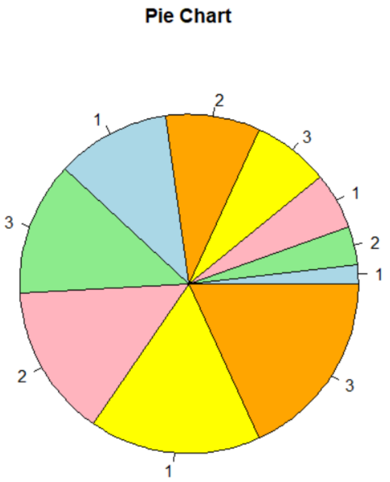



Bar Chart
Frequency
0
1
2
3
4
1
2
3
category


Dot Plot
3
1
2
3
1
2
3
1
2
1
20
40
60
80
100
X values


Box Plot
Y values
10
20
30
40
50
1
2
3
Category

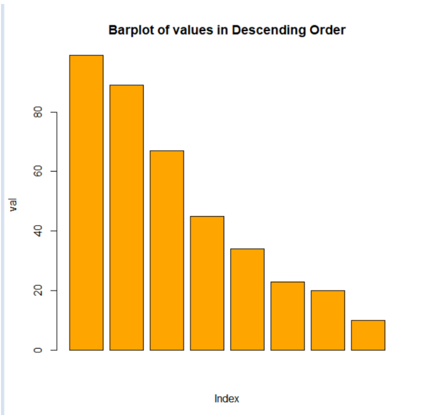

Histogram of X values
Frequency
0
1
2
0
20
40
60
80
100
X values


Pie Chart
1
2
3
1
2
1
3
1
2
3

(B) Write the script in R to sort the values contained in the following vector in ascending order and descending order: (23, 45, 10, 34, 89, 20, 67, 99). Demonstrate the output using graph.

```r
data_vec <- c(23, 45, 10, 34, 89, 20, 67, 99)

ascending_order <- sort(data_vec)

descending_order <- sort(data_vec, decreasing = TRUE)

print("Sorted in Ascending Order:")
print(ascending_order)

print("Sorted in Descending Order:")
print(descending_order)

barplot(ascending_order, main="Barplot of values in Ascending Order",
  col="orange", xlab = "Index", ylab = "val", yslim=c(0, 100))

barplot(descending_order, main="Barplot of values in Descending Order",
  col="orange", xlab = "Index", ylab = "val", yslim=c(0, 100))
```

```
> data_vec <- c(23, 45, 10, 34, 89, 20, 67, 99)
>
> ascending_order <- sort(data_vec)
>
> descending_order <- sort(data_vec, decreasing = TRUE)
>
> print("Sorted in Ascending Order:")
[1] "Sorted in Ascending Order:"
> print(ascending_order)
[1] 10 20 23 34 45 67 89 99
>
> print("Sorted in Descending Order:")
[1] "Sorted in Descending Order:"
> print(descending_order)
[1] 99 89 67 45 34 23 20 10
>
```



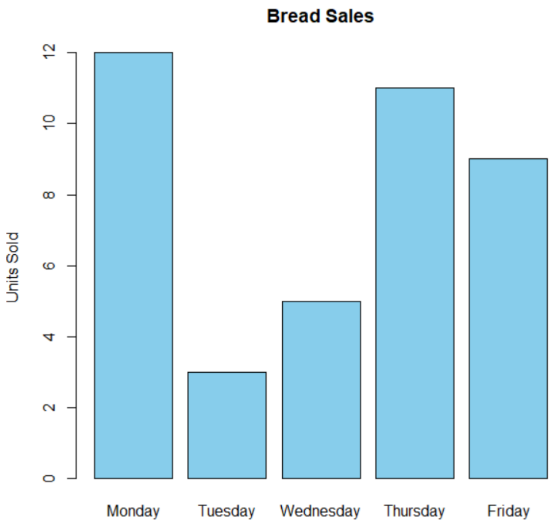
(B) The following table shows the number of units of different products sold on different da

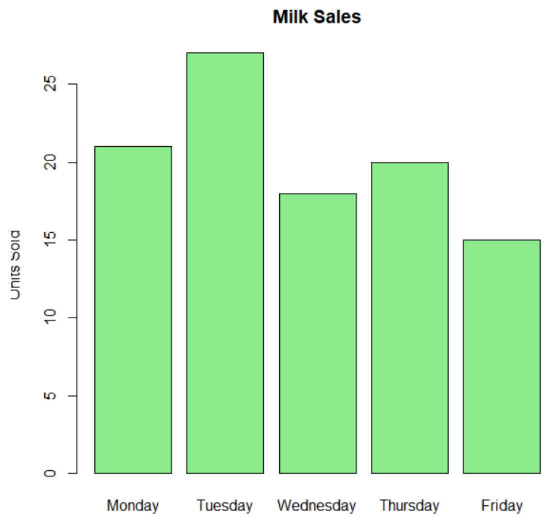
Product	Monday	Tuesday	Wednesday	Thursday	Friday
Bread	12	3	5	11	9
Milk	21	27	18	20	15
Cola Cans	10	1	33	6	12
Chocolate bars	6	7	4	13	12
Detergent	5	8	12	20	23

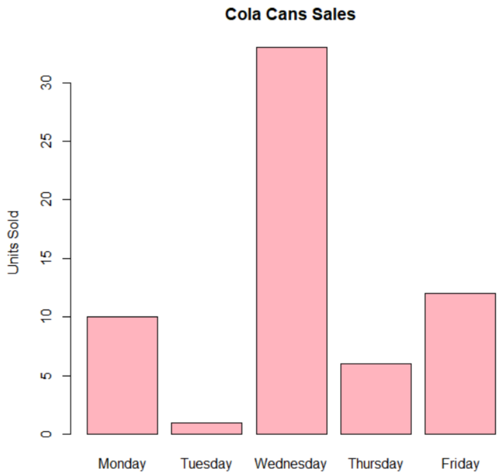
Create five sample numeric vectors from this data visualize data using R.

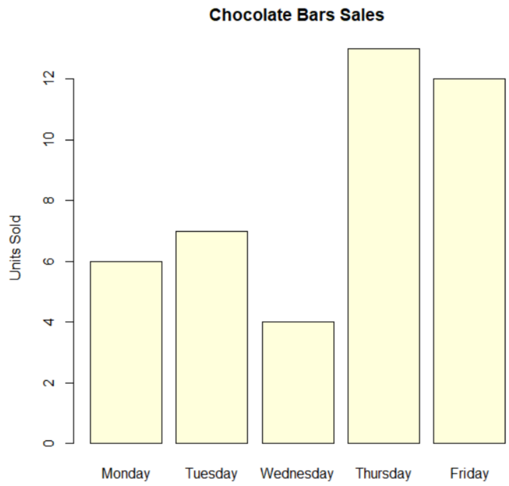
```
bread <- c(12, 3, 5, 11, 9)
milk <- c(21, 27, 18, 20, 15)
cola_cans <- c(10, 1, 33, 6, 12)
chocolate_bars <- c(6, 7, 4, 13, 12)
detergent <- c(5, 8, 12, 20, 23)

print(bread)
print(milk)
print(cola_cans)
print(chocolate_bars)
print(detergent)

days <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")

barplot(bread, names.arg=days, col="skyblue", main="Bread Sales",
ylab="Units Sold")


barplot(milk, names.arg=days, col="lightgreen", main="Milk Sales",
ylab="Units Sold")


barplot(cola_cans, names.arg=days, col="lightpink", main="Cola Cans
Sales", ylab="Units Sold")


barplot(chocolate_bars, names.arg=days, col="lightyellow",
main="Chocolate Bars Sales", ylab="Units Sold")


barplot(detergent, names.arg=days, col="skyblue", main="Detergent
Sales", ylab="Units Sold")
```

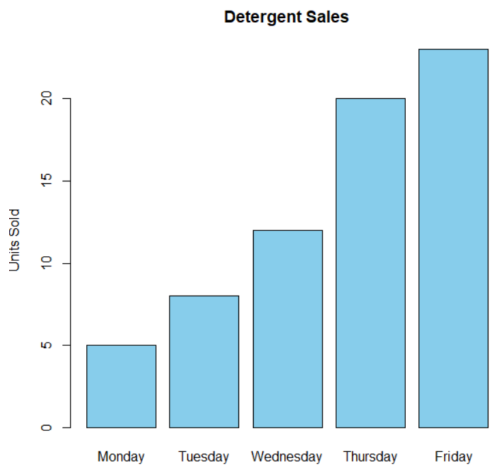
```
> bread <- c(12, 3, 5, 11, 9)
> milk <- c(21, 27, 18, 20, 15)
> cola_cans <- c(10, 1, 33, 6, 12)
> chocolate_bars <- c(6, 7, 4, 13, 12)
> detergent <- c(5, 8, 12, 20, 23)
>
> print(bread)
[1] 12  3  5 11  9
> print(milk)
[1] 21 27 18 20 15
> print(cola_cans)
[1] 10  1 33  6 12
> print(chocolate_bars)
[1]  6  7  4 13 12
> print(detergent)
[1]  5  8 12 20 23
>
> days <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
> 
```






Cola Cans Sales
Units Sold
0
5
10
15
20
25
30
Monday
Tuesday
Wednesday
Thursday
Friday


Chocolate Bars Sales
Units Sold
0
2
4
6
8
10
12
Monday
Tuesday
Wednesday
Thursday
Friday


Detergent Sales
Units Sold
0
5
10
15
20
Monday
Tuesday
Wednesday
Thursday
Friday

(b) Consider the following data frame given below:

Subject	Class	Marks
1	1	56
2	2	75
3	1	48
4	2	69
5	1	84
6	2	53

(i) Create a subset of subject less than 4 by using subset () funcon and demonstrate the output.

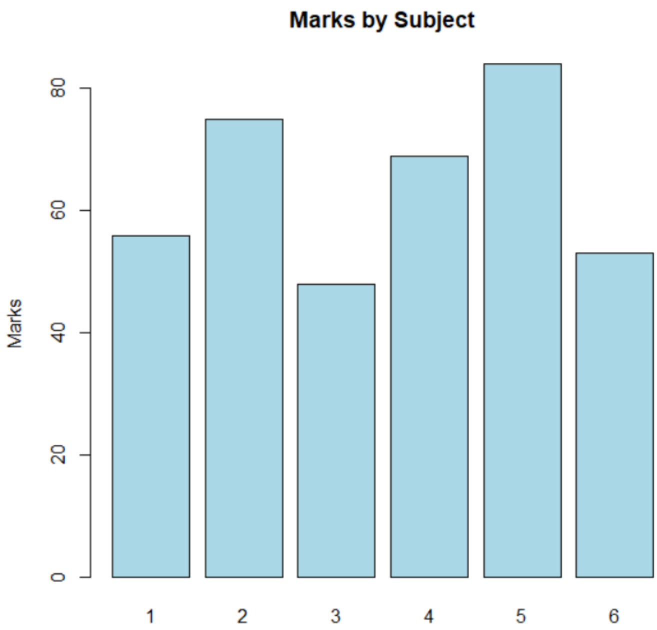
(ii) Create a subject where the subject column is less than 3 and the class equals to 2 by using [] brackets and demonstrate the output using R

(iii) Visualize data

```
df <- data.frame(
  Subject = c(1, 2, 3, 4, 5, 6),
  Class = c(1, 2, 1, 2, 1, 2),
  Marks = c(56, 75, 48, 69, 84, 53)
)

print("Original Data Frame:")
print(df)

# Create a subset where Subject is less than 4

subset_subject <- subset(df, Subject < 4)

print("Subset where Subject is less than 4")
print(subset_subject)

# Create a subset where Subject is less than 3 and Class equals 2 using
[]

bracket_subset <- df[df$Subject < 3 & df$Class == 2, ]

print("Subset where Subject is less than 3 and Class equals 2:")
```

```r
print(bracket_subset)

# Visualize

barplot(df$Marks, names.arg=df$Subject, col="lightblue", main="Marks by Subject", ylab="Marks")
```

```
> print("Subset where Subject is less than 4")
[1] "Subset where Subject is less than 4"
> print(subset_subject)
  Subject Class Marks
1       1     1    56
2       2     2    75
3       3     1    48
>
> # Create a subset where Subject is less than 3 and Class equals 2 using []
>
> bracket_subset <- df[df$Subject < 3 & df$Class == 2, ]
>
> print("Subset where Subject is less than 3 and Class equals 2:")
[1] "Subset where Subject is less than 3 and Class equals 2:"
> print(bracket_subset)
  Subject Class Marks
2       2     2    75
```



(b)The data analyst of Argon technology Mr. John needs to enter the salaries of 10 employees in R. The salaries of the employees are given in the following table:

Sr. No.	Name of employees	Salaries
1	Vivek	21000
2	Karan	55000
3	James	67000
4	Soham	50000
5	Renu	54000
6	Farah	40000
7	Hetal	30000
8	Mary	70000
9	Ganesh	20000
10	Krish	15000

i) Which R command will Mr. John use to enter these values demonstrate the output.

ii) Now Mr. John wants to add the salaries of 5 new employees in the existing table, which command he will use to join datasets with new values in R. Demonstrate the output.

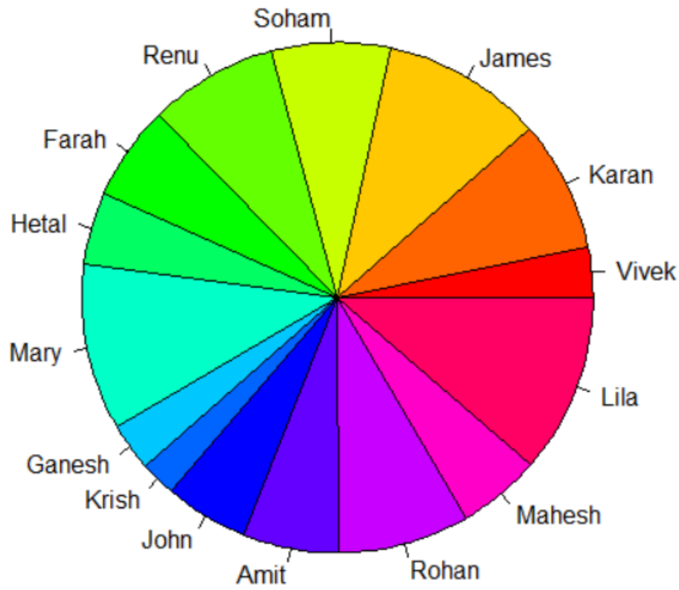
(iii) Visialize the data using chart .

```
# Step 1: Create a data frame with 10 employees and their salaries

df = data.frame(
  srNo = c(1,2,3,4,5,6,7,8,9,10),
  empName = c("Vivek", "Karan", "James", "Soham", "Renu", "Farah",
"Hetal", "Mary", "Ganesh", "Krish"),
  salaries = c(21000, 55000, 67000, 50000, 54000, 40000, 30000, 70000,
20000, 15000)
)

print(df)

# Step 2: Create a data frame for 5 new employees

new_df <- data.frame(
  srNo = 11:15,
  empName = c("John", "Amit", "Rohan", "Mahesh", "Lila"),
  salaries = c(35000, 40000, 55000, 35000, 75000)
)

updated_df <- rbind(df, new_df)

print("Updated Employee Salaries Data Frame with 5 New Employees:")
print(updated_df)
```

```r
# Step 3: Visualize the Data Using a Chart

pie(updated_df$salaries, labels=updated_df$empName,
col=rainbow(length(updated_df$salaries)),
main="Proportion of Employee Salaries")
```

```
> df = data.frame(
+ srNo = c(1,2,3,4,5,6,7,8,9,10),
+ empName = c("Vivek", "Karan", "James", "Soham", "Renu", "Farah", "Hetal", "Ma$
+ salaries = c(21000, 55000, 67000, 50000, 54000, 40000, 30000, 70000, 20000, 1$
+ )
>
> print(df)
   srNo empName salaries
1     1   Vivek    21000
2     2   Karan    55000
3     3   James    67000
4     4   Soham    50000
5     5    Renu    54000
6     6   Farah    40000
7     7   Hetal    30000
8     8    Mary    70000
9     9  Ganesh    20000
10   10   Krish    15000
>

> updated_df <- rbind(df, new_df)
>
> print("Updated Employee Salaries Data Frame with 5 New Employees:")
[1] "Updated Employee Salaries Data Frame with 5 New Employees:"
> print(updated_df)
   srNo empName salaries
1     1   Vivek    21000
2     2   Karan    55000
3     3   James    67000
4     4   Soham    50000
5     5    Renu    54000
6     6   Farah    40000
7     7   Hetal    30000
8     8    Mary    70000
9     9  Ganesh    20000
10   10   Krish    15000
11   11    John    35000
12   12    Amit    40000
13   13   Rohan    55000
14   14  Mahesh    35000
15   15    Lila    75000
```

Proportion of Employee Salaries



(b) Analyse and visualize churn modelling data using R.

```r
data <- read.csv("E:/College/SEM-VII/BDA/PRACS/ChurnData.csv")
head(data)

#summary
summary(data)

#structure
str(data)

#checking missing values
sum(is.na(data))

#visualization

#scatter plot

plot(data$age, data$balance,
  main = "Bar Plot",
  xlab = "Age",
  ylab = "Balance",
  col = "blue",
  pch = 19
)

#bubble chart

symbols(data$age, data$income,
  circles = data$churn,
  inches = 0.5,
  fg = 'blue',
  bg = 'light blue',
  main = 'Bubble Chart',
  xlab = "Age",
  ylab = "Income"
)

# Bar Plot

barplot(table(data$income),
  main = "Bar Plot",
```

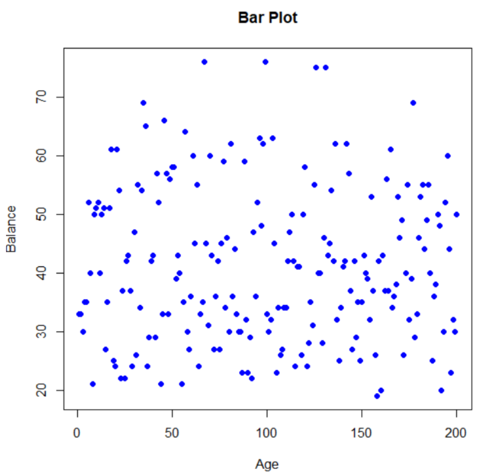
```r
  xlab = "Income",
  ylab = "Count",
  col = "lightpink"
)

# Box Plot

boxplot(data$age~data$income,
  main = "Box Plot",
  xlab = "Age",
  ylab = "Income",
  col = "lightyellow"
)

# pie chart

age_counts <- table(data$age)

pie(age_counts,
 main = "Age Distribution",
 col = c("lightblue", "lightpink")
)

# Histogram

hist(data$tollmon,
  main = "Histogram",
  xlab = "TOLLMON",
  ylab = "Frequency",
  col = "lightblue",
  border = "black",
  breaks = 20
)
```

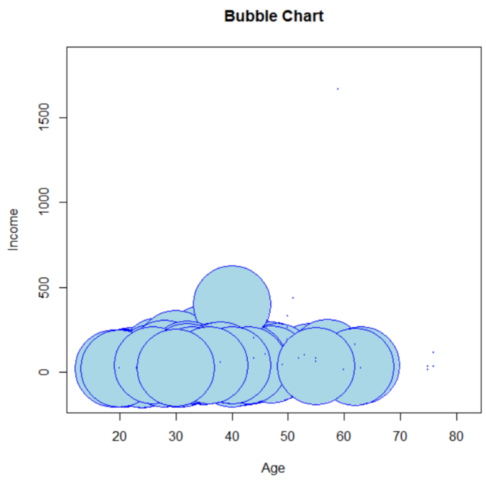
```
> data <- read.csv("E:/College/SEM-VII/BDA/PRACS/ChurnData.csv")
> head(data)
  tenure age address income ed employ equip callcard wireless longmon tollmon
1     11  33       7    136  5      5     0        1        1    4.40   20.75
2     33  33      12     33  2      0     0        0        0    9.45    0.00
3     23  30       9     30  1      2     0        0        0    6.30    0.00
4     38  35       5     76  2     10     1        1        1    6.05   45.00
5      7  35      14     80  2     15     0        1        0    7.10   22.00
6     68  52      17    120  1     24     0        1        0   20.70    0.00
  equipmon cardmon wiremon longten tollten cardten voice pager internet
1      0.0   15.25    35.7   42.00  211.45     125     1     1        0
2      0.0    0.00     0.0  288.80    0.00       0     0     0        0
3      0.0    0.00     0.0  157.05    0.00       0     0     0        0
4     50.1   23.25    64.9  239.55 1873.05     880     1     1        1
5      0.0   23.75     0.0   47.45  166.10     145     1     0        0
6      0.0   22.00     0.0 1391.05    0.00    1505     0     0        0
  callwait confer ebill loglong logtoll lninc custcat churn
1        1      1     0   1.482   3.033 4.913       4     1
2        0      0     0   2.246   3.240 3.497       1     1
3        0      1     0   1.841   3.240 3.401       3     0
4        1      1     1   1.800   3.807 4.331       4     0
5        1      1     0   1.960   3.091 4.382       3     0
6        0      0     0   3.030   3.240 4.787       1     0

> #summary
> summary(data)
     tenure           age           address          income
 Min.   : 1.00   Min.   :19.00   Min.   : 0.00   Min.   :   9.00
 1st Qu.:16.75   1st Qu.:31.00   1st Qu.: 3.00   1st Qu.:  31.00
 Median :33.50   Median :40.00   Median : 9.00   Median :  48.00
 Mean   :35.51   Mean   :41.16   Mean   :11.65   Mean   :  75.13
 3rd Qu.:55.25   3rd Qu.:51.00   3rd Qu.:18.00   3rd Qu.:  80.00
 Max.   :72.00   Max.   :76.00   Max.   :48.00   Max.   :1668.00
       ed             employ          equip          callcard        wireless
 Min.   :1.000   Min.   : 0.00   Min.   :0.000   Min.   :0.000   Min.   :0.00
 1st Qu.:2.000   1st Qu.: 3.00   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.00
 Median :3.000   Median : 7.50   Median :0.000   Median :1.000   Median :0.00
 Mean   :2.825   Mean   :10.22   Mean   :0.425   Mean   :0.705   Mean   :0.29
 3rd Qu.:4.000   3rd Qu.:17.00   3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.:1.00
 Max.   :5.000   Max.   :44.00   Max.   :1.000   Max.   :1.000   Max.   :1.00
    longmon          tollmon          equipmon         cardmon
 Min.   : 1.100   Min.   : 0.00   Min.   : 0.00   Min.   :  0.00
 1st Qu.: 5.537   1st Qu.: 0.00   1st Qu.: 0.00   1st Qu.:  0.00
 Median : 8.250   Median : 0.00   Median : 0.00   Median : 12.50
 Mean   :11.789   Mean   :13.24   Mean   :15.78   Mean   : 14.36
 3rd Qu.:14.300   3rd Qu.:24.75   3rd Qu.:33.01   3rd Qu.: 20.75
 Max.   :62.300   Max.   :68.50   Max.   :63.25   Max.   :109.25
```

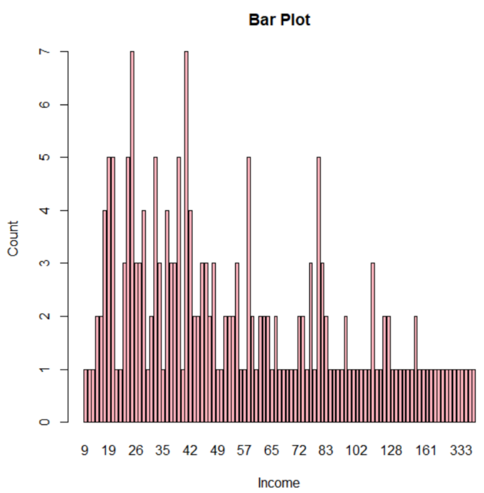
```
   wiremon          longten          tollten          cardten     
 Min.   :  0.00   Min.   :   1.10   Min.   :   0.0   Min.   :   0.0  
 1st Qu.:  0.00   1st Qu.:  79.34   1st Qu.:   0.0   1st Qu.:   0.0  
 Median :  0.00   Median : 289.52   Median :   0.0   Median : 342.5  
 Mean   : 12.22   Mean   : 577.77   Mean   : 507.0   Mean   : 650.7  
 3rd Qu.: 23.46   3rd Qu.: 806.76   3rd Qu.: 724.2   3rd Qu.: 921.2  
 Max.   :109.70   Max.   :4333.00   Max.   :4938.6   Max.   :7515.0  
     voice            pager           internet        callwait         confer    
 Min.   :0.000   Min.   :0.000   Min.   :0.00   Min.   :0.000   Min.   :0.00  
 1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.00   1st Qu.:0.000   1st Qu.:0.00  
 Median :0.000   Median :0.000   Median :0.00   Median :0.000   Median :0.00  
 Mean   :0.295   Mean   :0.275   Mean   :0.44   Mean   :0.455   Mean   :0.46  
 3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.:1.00   3rd Qu.:1.000   3rd Qu.:1.00  
 Max.   :1.000   Max.   :1.000   Max.   :1.00   Max.   :1.000   Max.   :1.00  
     ebill         loglong          logtoll          lninc          custcat     
 Min.   :0.00   Min.   :0.095   Min.   :1.749   Min.   :2.197   Min.   :1.000  
 1st Qu.:0.00   1st Qu.:1.712   1st Qu.:3.227   1st Qu.:3.434   1st Qu.:2.000  
 Median :0.00   Median :2.110   Median :3.240   Median :3.871   Median :2.000  
 Mean   :0.44   Mean   :2.193   Mean   :3.229   Mean   :3.951   Mean   :2.475  
 3rd Qu.:1.00   3rd Qu.:2.660   3rd Qu.:3.240   3rd Qu.:4.382   3rd Qu.:3.000  
 Max.   :1.00   Max.   :4.132   Max.   :4.227   Max.   :7.419   Max.   :4.000  
     churn     
 Min.   :0.00  
 1st Qu.:0.00  
```

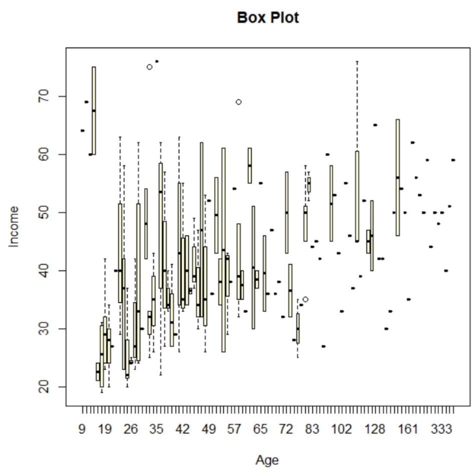
```
     churn     
 Min.   :0.00  
 1st Qu.:0.00  
 Median :0.00  
 Mean   :0.29  
 3rd Qu.:1.00  
 Max.   :1.00  
> 
> #structure
> str(data)
'data.frame':	200 obs. of  28 variables:
 $ tenure  : num  11 33 23 38 7 68 42 9 35 49 ...
 $ age     : num  33 33 30 35 35 52 40 21 50 51 ...
 $ address : num  7 12 9 5 14 17 7 1 26 27 ...
 $ income  : num  136 33 30 76 80 120 37 17 140 63 ...
 $ ed      : num  5 2 1 2 2 1 2 2 2 4 ...
 $ employ  : num  5 0 2 10 15 24 8 2 21 19 ...
 $ equip   : num  0 0 0 1 0 0 1 0 0 0 ...
 $ callcard: num  1 0 0 1 1 1 1 0 1 1 ...
 $ wireless: num  1 0 0 1 0 0 1 0 0 0 ...
 $ longmon : num  4.4 9.45 6.3 6.05 7.1 ...
 $ tollmon : num  20.8 0 0 45 22 ...
 $ equipmon: num  0 0 0 50.1 0 0 36.9 0 0 0 ...
```

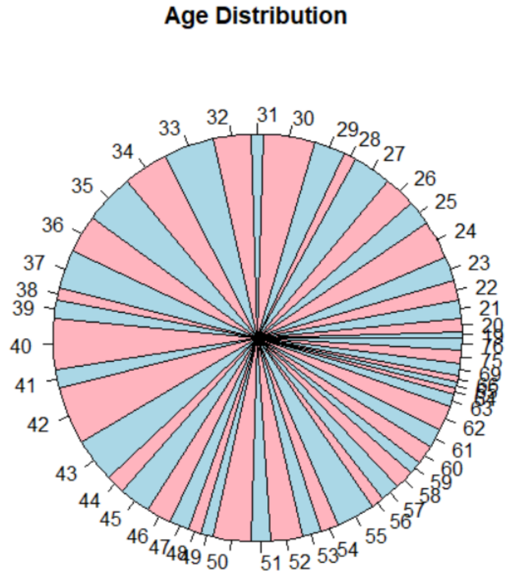
```
 $ cardmon : num  15.2 0 0 23.2 23.8 ...
 $ wiremon : num  35.7 0 0 64.9 0 0 37.4 0 0 0 ...
 $ longten : num  42 288.8 157.1 239.6 47.5 ...
 $ tollten : num  211 0 0 1873 166 ...
 $ cardten : num  125 0 0 880 145 ...
 $ voice   : num  1 0 0 1 1 0 1 0 0 0 ...
 $ pager   : num  1 0 0 1 0 0 0 0 0 0 ...
 $ internet: num  0 0 0 1 0 0 1 0 0 1 ...
 $ callwait: num  1 0 0 1 1 0 1 0 1 1 ...
 $ confer  : num  1 0 1 1 1 0 1 0 1 0 ...
 $ ebill   : num  0 0 0 1 0 0 1 0 0 1 ...
 $ loglong : num  1.48 2.25 1.84 1.8 1.96 ...
 $ logtoll : num  3.03 3.24 3.24 3.81 3.09 ...
 $ lninc   : num  4.91 3.5 3.4 4.33 4.38 ...
 $ custcat : num  4 1 3 4 3 1 4 1 3 2 ...
 $ churn   : num  1 1 0 0 0 0 0 0 0 0 ...
>
> #checking missing values
> sum(is.na(data))
[1] 0
```

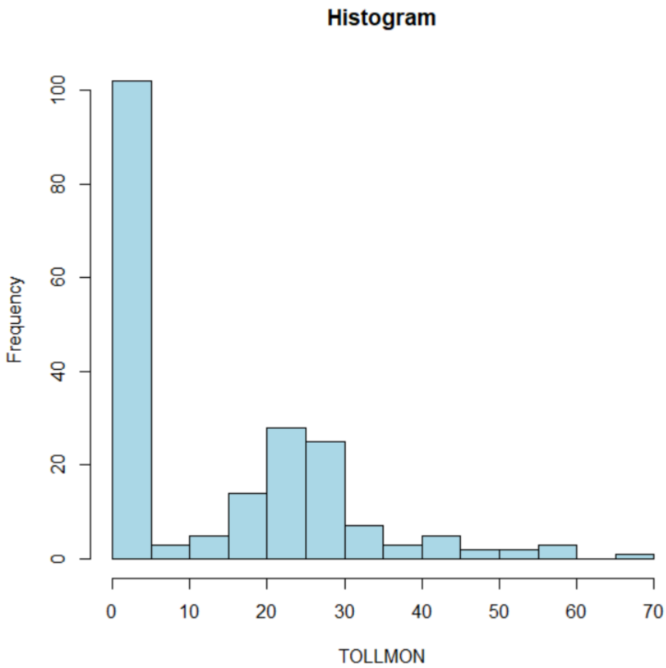









Age Distribution
31
30
29
28
27
26
25
24
23
22
21
20
76
75
69
63
62
61
60
59
58
57
56
55
54
53
52
51
50
49
47
46
45
44
43
42
41
40
39
38
37
36
35
34
33
32


Histogram
Frequency
0
20
40
60
80
100
0
10
20
30
40
50
60
70
TOLLMON

(b) Analyse and visualize IRIS data using R.

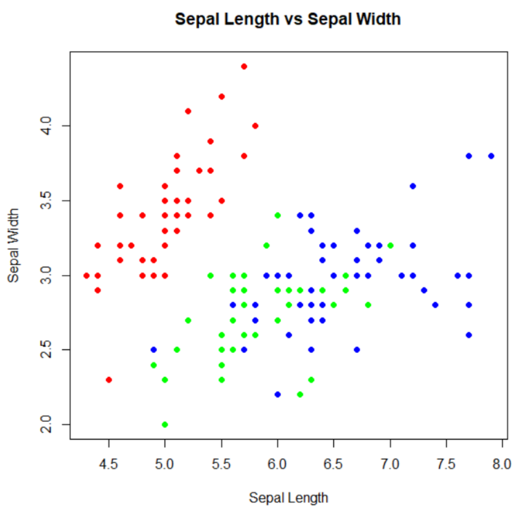
```r
library(dplyr)

data = read.csv("E:/College/SEM-VII/BDA/PRACS/iris.csv")
head(data)

# summary
summary(data)

# structure
str(data)

# cleaning
sum(is.na(data))

# remove missing values (if)
data <- data %>%
  filter(!is.na(SepalLengthCm))

# Scatter Plot

colors <- as.factor(data$Species)
color_palette <- c("red", "green", "blue")

plot(data$SepalLengthCm, data$SepalWidthCm,
    main = "Sepal Length vs Sepal Width",
    xlab = "Sepal Length",
    ylab = "Sepal Width",
    col = color_palette[colors],
    pch = 19
)

# Box Plot

boxplot(data$PetalLengthCm~data$Species,
    main = "Petal Length Distribution by Species",
    xlab = "Petal Length",
    ylab = "Species",
    col = c("lightblue", "lightyellow", "lightpink")
)
```

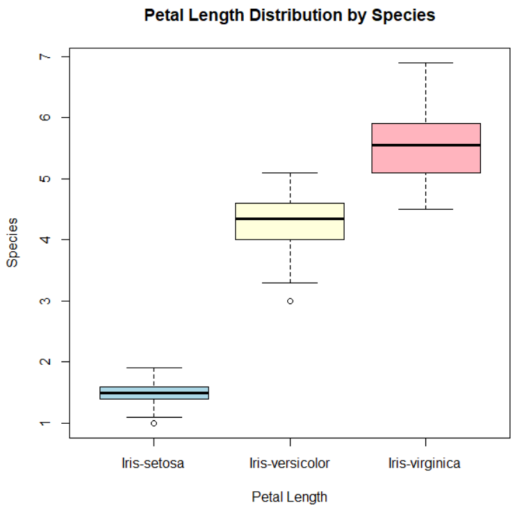
```r
# Histogram

hist(data$SepalLengthCm,
    main = "Sepal Length Distribution",
    xlab = "Sepal Length",
    ylab = "Frequnecy",
    col = "lightyellow",
    breaks= 15,
    border = "black"
)

# Pair plot

pairs(data[1:4],
    main = "Pairwise Plot of Iris Feature",
    col = color_palette[colors],
)

# Correlation Matrix

corr_matrix <- cor(data[,1:4])
print(corr_matrix)

# Heatmap

heatmap(corr_matrix,
    main = "Correlation Heatmap of Iris Features",
    col = heat.colors(256)
)
```

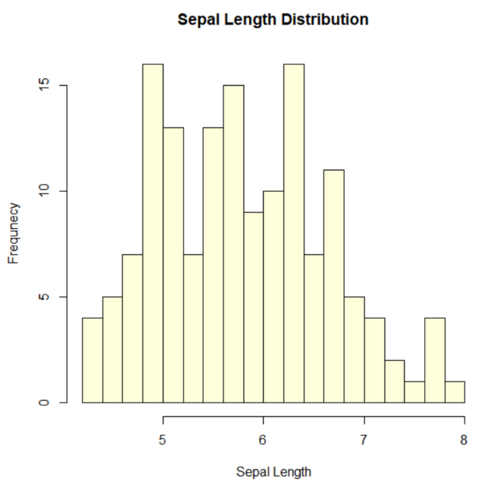
```
>
> data = read.csv("E:/College/SEM-VII/BDA/PRACS/iris.csv")
> head(data)
  Id SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm     Species
1  1           5.1          3.5           1.4          0.2 Iris-setosa
2  2           4.9          3.0           1.4          0.2 Iris-setosa
3  3           4.7          3.2           1.3          0.2 Iris-setosa
4  4           4.6          3.1           1.5          0.2 Iris-setosa
5  5           5.0          3.6           1.4          0.2 Iris-setosa
6  6           5.4          3.9           1.7          0.4 Iris-setosa
>
> # summary
> summary(data)
       Id         SepalLengthCm    SepalWidthCm   PetalLengthCm
 Min.   :  1.00   Min.   :4.300   Min.   :2.000   Min.   :1.000
 1st Qu.: 38.25   1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600
 Median : 75.50   Median :5.800   Median :3.000   Median :4.350
 Mean   : 75.50   Mean   :5.843   Mean   :3.054   Mean   :3.759
 3rd Qu.:112.75   3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100
 Max.   :150.00   Max.   :7.900   Max.   :4.400   Max.   :6.900
  PetalWidthCm     Species
 Min.   :0.100   Length:150
 1st Qu.:0.300   Class :character
 Median :1.300   Mode  :character
```

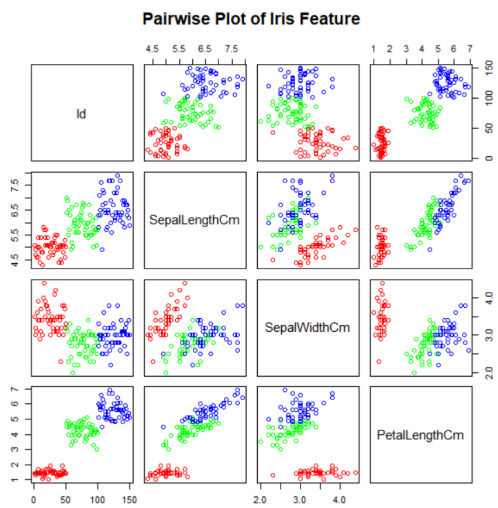
```
1st Qu.:0.300   Class :character
Median :1.300   Mode  :character
Mean   :1.199
3rd Qu.:1.800
Max.   :2.500
>
> # structure
> str(data)
'data.frame':	150 obs. of  6 variables:
 $ Id           : int  1 2 3 4 5 6 7 8 9 10 ...
 $ SepalLengthCm: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ SepalWidthCm : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ PetalLengthCm: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ PetalWidthCm : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species      : chr  "Iris-setosa" "Iris-setosa" "Iris-setosa" "Iris-setosa" $
>
> # cleaning
> sum(is.na(data))
[1] 0
>
> # remove missing values (if)
> data <- data %>%
+   filter(!is.na(SepalLengthCm))
>
```

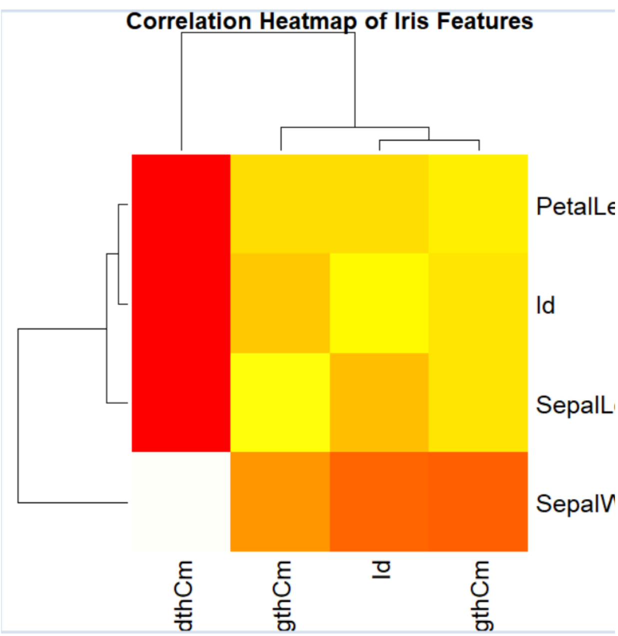
```
> corr_matrix <- cor(data[,1:4])
> print(corr_matrix)
                      Id SepalLengthCm SepalWidthCm PetalLengthCm
Id             1.0000000     0.7166763   -0.3977288     0.8827473
SepalLengthCm  0.7166763     1.0000000   -0.1093692     0.8717542
SepalWidthCm  -0.3977288    -0.1093692    1.0000000    -0.4205161
PetalLengthCm  0.8827473     0.8717542   -0.4205161     1.0000000
```


Sepal Length vs Sepal Width
Sepal Width
Sepal Length


Petal Length Distribution by Species
Species
Iris-setosa
Iris-versicolor
Iris-virginica
Petal Length


Sepal Length Distribution
Frequnecy
Sepal Length


Pairwise Plot of Iris Feature
Id
SepalLengthCm
SepalWidthCm
PetalLengthCm


Correlation Heatmap of Iris Features
PetalLe
Id
SepalL
SepalW
dthCm
gthCm
Id
gthCm

(b) Analyse and visualize supermarket data using R.

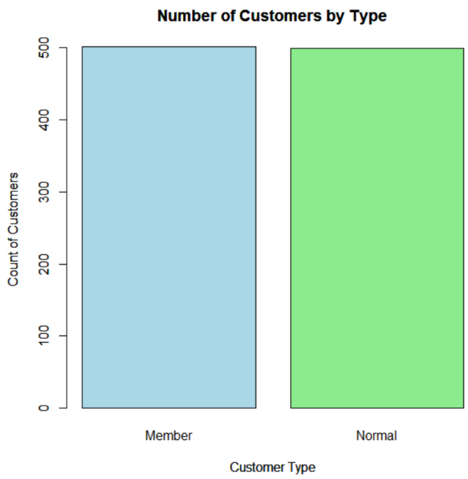
```r
library(ggplot2)
library(dplyr)

supermarket_data <-
read.csv("E:/College/SEM-VII/BDA/PRACS/supermarket.csv")
head(supermarket_data)

# Summary of the dataset
summary(supermarket_data)

# structure of the dataset
str(supermarket_data)

# Check for missing values
sum(is.na(supermarket_data))

# Scatter plot
plot(supermarket_data$Quantity, supermarket_data$Total,
    main = "Total Price vs Quantity Purchased",
    xlab = "Quantity",
    ylab = "Total Price ($)",
    col = "blue",
    pch = 19
)

# Bar plot
customer_type_counts <- table(supermarket_data$Customer.type)
barplot(customer_type_counts,
    main = "Number of Customers by Type",
    xlab = "Customer Type",
    ylab = "Count of Customers",
    col = c("lightblue", "lightgreen")
)

# Box plot
boxplot(supermarket_data$Unit.price ~ supermarket_data$Product.line,
    main = "Unit Price Distribution by Product Line",
    xlab = "Product Line",
    ylab = "Unit Price ($)",
```

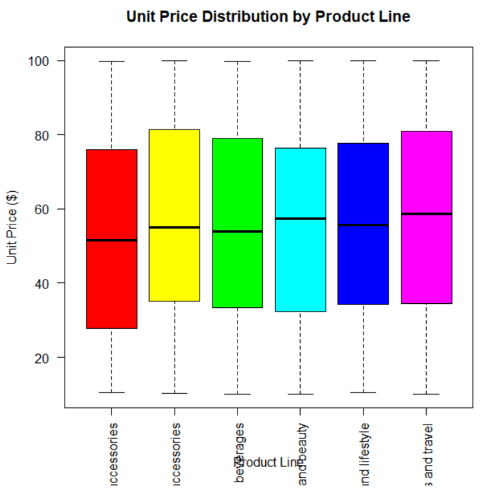
```r
    col = rainbow(length(unique(supermarket_data$Product.line))),
    las = 2
)

# Pie chart for Gender Distribution using base R
gender_counts <- table(supermarket_data$Gender)
pie(gender_counts,
    main = "Gender Distribution",
    col = c("lightblue", "pink")
)
```

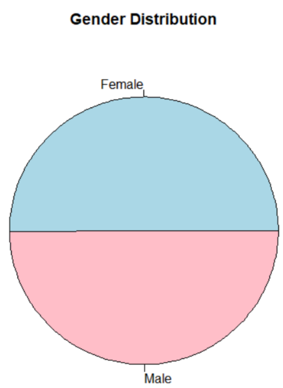
```
> supermarket_data <- read.csv("E:/College/SEM-VII/BDA/PRACS/supermarket.csv")
> head(supermarket_data)
   Invoice.ID Branch      City Customer.type Gender            Product.line
1 750-67-8428      A    Yangon        Member Female       Health and beauty
2 226-31-3081      C Naypyitaw        Normal Female Electronic accessories
3 631-41-3108      A    Yangon        Normal   Male      Home and lifestyle
4 123-19-1176      A    Yangon        Member   Male       Health and beauty
5 373-73-7910      A    Yangon        Normal   Male       Sports and travel
6 699-14-3026      C Naypyitaw        Normal   Male Electronic accessories
  Unit.price Quantity Tax.5.    Total       Date  Time     Payment   cogs
1      74.69        7 26.1415 548.9715  1/5/2019 13:08     Ewallet 522.83
2      15.28        5  3.8200  80.2200  3/8/2019 10:29        Cash  76.40
3      46.33        7 16.2155 340.5255  3/3/2019 13:23 Credit card 324.31
4      58.22        8 23.2880 489.0480 1/27/2019 20:33     Ewallet 465.76
5      86.31        7 30.2085 634.3785  2/8/2019 10:37     Ewallet 604.17
6      85.39        7 29.8865 627.6165 3/25/2019 18:30     Ewallet 597.73
  gross.margin.percentage gross.income Rating
1               4.761905      26.1415    9.1
2               4.761905       3.8200    9.6
3               4.761905      16.2155    7.4
4               4.761905      23.2880    8.4
5               4.761905      30.2085    5.3
6               4.761905      29.8865    4.1
>
```

```
> # Summary of the dataset
> summary(supermarket_data)
  Invoice.ID           Branch              City           Customer.type     
 Length:1000        Length:1000        Length:1000        Length:1000       
 Class :character   Class :character   Class :character   Class :character  
 Mode  :character   Mode  :character   Mode  :character   Mode  :character  



    Gender          Product.line         Unit.price       Quantity    
 Length:1000        Length:1000        Min.   :10.08   Min.   : 1.00  
 Class :character   Class :character   1st Qu.:32.88   1st Qu.: 3.00  
 Mode  :character   Mode  :character   Median :55.23   Median : 5.00  
                                       Mean   :55.67   Mean   : 5.51  
                                       3rd Qu.:77.94   3rd Qu.: 8.00  
                                       Max.   :99.96   Max.   :10.00  
     Tax.5.            Total              Date               Time          
 Min.   : 0.5085   Min.   :  10.68   Length:1000        Length:1000       
 1st Qu.: 5.9249   1st Qu.: 124.42   Class :character   Class :character  
 Median :12.0880   Median : 253.85   Mode  :character   Mode  :character  
 Mean   :15.3794   Mean   : 322.97                                        
 3rd Qu.:22.4453   3rd Qu.: 471.35                                        
 Max.   :49.6500   Max.   :1042.65                                        
   Payment               cogs        gross.margin.percentage  gross.income    
 Length:1000        Min.   : 10.17   Min.   :4.762           Min.   : 0.5085  

   Payment               cogs        gross.margin.percentage  gross.income    
 Length:1000        Min.   : 10.17   Min.   :4.762           Min.   : 0.5085  
 Class :character   1st Qu.:118.50   1st Qu.:4.762           1st Qu.: 5.9249  
 Mode  :character   Median :241.76   Median :4.762           Median :12.0880  
                    Mean   :307.59   Mean   :4.762           Mean   :15.3794  
                    3rd Qu.:448.90   3rd Qu.:4.762           3rd Qu.:22.4453  
                    Max.   :993.00   Max.   :4.762           Max.   :49.6500  
     Rating      
 Min.   : 4.000  
 1st Qu.: 5.500  
 Median : 7.000  
 Mean   : 6.973  
 3rd Qu.: 8.500  
 Max.   :10.000  
> 
> # structure of the dataset
> str(supermarket_data)
'data.frame':   1000 obs. of  17 variables:
 $ Invoice.ID             : chr  "750-67-8428" "226-31-3081" "631-41-3108" "123$
 $ Branch                 : chr  "A" "C" "A" "A" ...
 $ City                   : chr  "Yangon" "Naypyitaw" "Yangon" "Yangon" ...
 $ Customer.type          : chr  "Member" "Normal" "Normal" "Member" ...
 $ Gender                 : chr  "Female" "Female" "Male" "Male" ...
 $ Product.line           : chr  "Health and beauty" "Electronic accessories" "$
```

```
> str(supermarket_data)
'data.frame':   1000 obs. of  17 variables:
 $ Invoice.ID             : chr  "750-67-8428" "226-31-3081" "631-41-3108" "123$
 $ Branch                 : chr  "A" "C" "A" "A" ...
 $ City                   : chr  "Yangon" "Naypyitaw" "Yangon" "Yangon" ...
 $ Customer.type          : chr  "Member" "Normal" "Normal" "Member" ...
 $ Gender                 : chr  "Female" "Female" "Male" "Male" ...
 $ Product.line           : chr  "Health and beauty" "Electronic accessories" "$
 $ Unit.price             : num  74.7 15.3 46.3 58.2 86.3 ...
 $ Quantity               : int  7 5 7 8 7 7 6 10 2 3 ...
 $ Tax.5.                 : num  26.14 3.82 16.22 23.29 30.21 ...
 $ Total                  : num  549 80.2 340.5 489 634.4 ...
 $ Date                   : chr  "1/5/2019" "3/8/2019" "3/3/2019" "1/27/2019" .$
 $ Time                   : chr  "13:08" "10:29" "13:23" "20:33" ...
 $ Payment                : chr  "Ewallet" "Cash" "Credit card" "Ewallet" ...
 $ cogs                   : num  522.8 76.4 324.3 465.8 604.2 ...
 $ gross.margin.percentage: num  4.76 4.76 4.76 4.76 4.76 ...
 $ gross.income           : num  26.14 3.82 16.22 23.29 30.21 ...
 $ Rating                 : num  9.1 9.6 7.4 8.4 5.3 4.1 5.8 8 7.2 5.9 ...
>
> # Check for missing values
> sum(is.na(supermarket_data))
[1] 0
>
```







(b) Analyse and visualize Loan data using R.

```r
loan_data <- read.csv("E:/College/SEM-VII/BDA/PRACS/loan.csv")
head(loan_data)

# Summary of the dataset
summary(loan_data)

# structure of the dataset
str(loan_data)

# Check for missing values
sum(is.na(loan_data))

# Summary of the key columns
summary(loan_data)

# Bar plot for Gender distribution
barplot(table(loan_data$Gender),
 main = "Gender Distribution of Loan Applicants",
 xlab = "Gender",
 ylab = "Count",
 col = c("lightblue", "lightpink")
)

# Box plot for Loan Amount by Education level
boxplot(LoanAmount ~ Education,
 data = loan_data,
 main = "Loan Amount by Education Level",
 xlab = "Education Level",
 ylab = "Loan Amount",
 col = c("lightblue", "lightgreen"))

# Scatter plot of Applicant Income vs Loan Amount
plot(loan_data$ApplicantIncome, loan_data$LoanAmount,
 main = "Applicant Income vs Loan Amount",
 xlab = "Applicant Income",
 ylab = "Loan Amount",
 col = "blue", pch = 19
)

# Histogram for Loan Amount distribution
```

```r
hist(loan_data$LoanAmount,
 main = "Loan Amount Distribution",
 xlab = "Loan Amount",
 col = "lightblue", border = "black")
)

# Pie chart for Marital Status
marital_status <- table(loan_data$Married)
pie(marital_status,
 main = "Marital Status Distribution",
 col = c("lightblue", "lightgreen")
)
```

```
> loan_data <- read.csv("E:/College/SEM-VII/BDA/PRACS/loan.csv")
> head(loan_data)
   Loan_ID Gender Married Dependents    Education Self_Employed ApplicantIncome
1 LP001002   Male      No          0     Graduate            No            5849
2 LP001003   Male     Yes          1     Graduate            No            4583
3 LP001005   Male     Yes          0     Graduate           Yes            3000
4 LP001006   Male     Yes          0 Not Graduate            No            2583
5 LP001008   Male      No          0     Graduate            No            6000
6 LP001011   Male     Yes          2     Graduate           Yes            5417
  CoapplicantIncome LoanAmount Loan_Amount_Term Credit_History Property_Area
1                 0         NA              360              1         Urban
2              1508        128              360              1         Rural
3                 0         66              360              1         Urban
4              2358        120              360              1         Urban
5                 0        141              360              1         Urban
6              4196        267              360              1         Urban
  Loan_Status
1           Y
2           N
3           Y
4           Y
5           Y
```

```
> # Summary of the dataset
> summary(loan_data)
   Loan_ID             Gender            Married           Dependents       
 Length:614         Length:614         Length:614         Length:614        
 Class :character   Class :character   Class :character   Class :character  
 Mode  :character   Mode  :character   Mode  :character   Mode  :character  
```

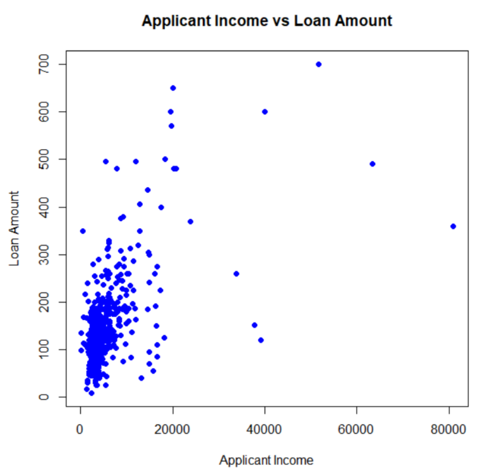
```
  Education         Self_Employed      ApplicantIncome CoapplicantIncome
 Length:614         Length:614         Min.   :  150   Min.   :    0    
 Class :character   Class :character   1st Qu.: 2878   1st Qu.:    0    
 Mode  :character   Mode  :character   Median : 3812   Median : 1188    
                                       Mean   : 5403   Mean   : 1621    
                                       3rd Qu.: 5795   3rd Qu.: 2297    
                                       Max.   :81000   Max.   :41667    
```

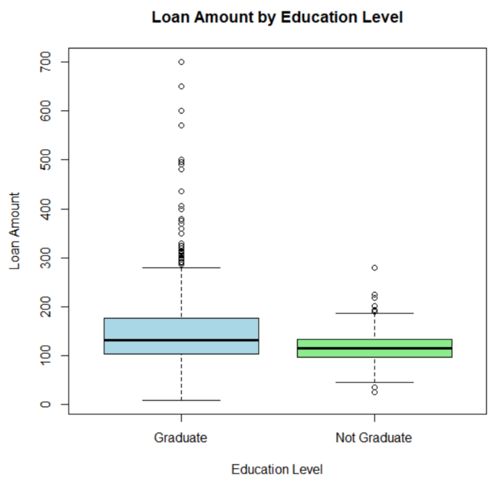
```
   LoanAmount    Loan_Amount_Term Credit_History   Property_Area     
 Min.   :  9.0   Min.   : 12      Min.   :0.0000   Length:614        
 1st Qu.:100.0   1st Qu.:360      1st Qu.:1.0000   Class :character  
 Median :128.0   Median :360      Median :1.0000   Mode  :character  
 Mean   :146.4   Mean   :342      Mean   :0.8422                     
 3rd Qu.:168.0   3rd Qu.:360      3rd Qu.:1.0000                     
 Max.   :700.0   Max.   :480      Max.   :1.0000                     
```

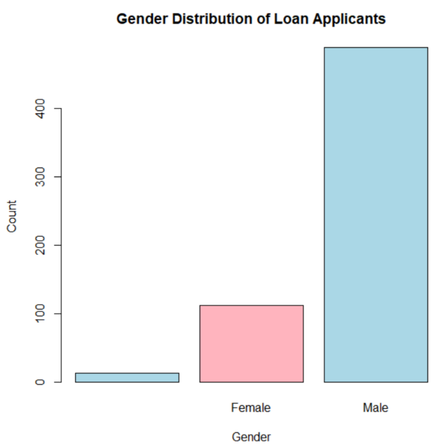
```
   LoanAmount    Loan_Amount_Term Credit_History   Property_Area     
 Min.   :  9.0   Min.   : 12      Min.   :0.0000   Length:614        
 1st Qu.:100.0   1st Qu.:360      1st Qu.:1.0000   Class :character  
 Median :128.0   Median :360      Median :1.0000   Mode  :character  
 Mean   :146.4   Mean   :342      Mean   :0.8422                     
 3rd Qu.:168.0   3rd Qu.:360      3rd Qu.:1.0000                     
 Max.   :700.0   Max.   :480      Max.   :1.0000                     
 NA's   :22      NA's   :14       NA's   :50                         
 Loan_Status       
 Length:614        
 Class :character  
 Mode  :character  
```

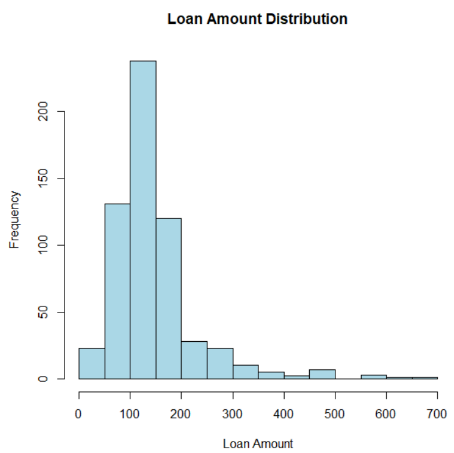
```
> # structure of the dataset
> str(loan_data)
data.frame':   614 obs. of  13 variables:
$ Loan_ID           : chr  "LP001002" "LP001003" "LP001005" "LP001006" ...
$ Gender            : chr  "Male" "Male" "Male" "Male" ...
$ Married           : chr  "No" "Yes" "Yes" "Yes" ...
$ Dependents        : chr  "0" "1" "0" "0" ...
$ Education         : chr  "Graduate" "Graduate" "Graduate" "Not Graduate" ...
```

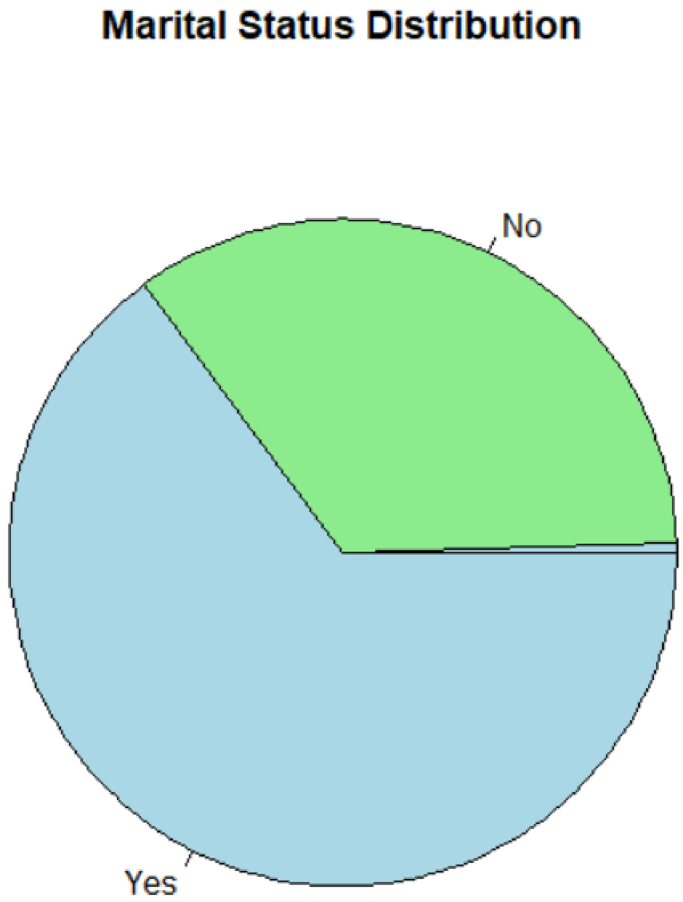
```
>
> # structure of the dataset
> str(loan_data)
'data.frame':	614 obs. of  13 variables:
 $ Loan_ID          : chr  "LP001002" "LP001003" "LP001005" "LP001006" ...
 $ Gender           : chr  "Male" "Male" "Male" "Male" ...
 $ Married          : chr  "No" "Yes" "Yes" "Yes" ...
 $ Dependents       : chr  "0" "1" "0" "0" ...
 $ Education        : chr  "Graduate" "Graduate" "Graduate" "Not Graduate" ...
 $ Self_Employed    : chr  "No" "No" "Yes" "No" ...
 $ ApplicantIncome  : int  5849 4583 3000 2583 6000 5417 2333 3036 4006 12841 .$
 $ CoapplicantIncome: num  0 1508 0 2358 0 ...
 $ LoanAmount       : int  NA 128 66 120 141 267 95 158 168 349 ...
 $ Loan_Amount_Term : int  360 360 360 360 360 360 360 360 360 360 ...
 $ Credit_History   : int  1 1 1 1 1 1 1 0 1 1 ...
 $ Property_Area    : chr  "Urban" "Rural" "Urban" "Urban" ...
 $ Loan_Status      : chr  "Y" "N" "Y" "Y" ...
> # Check for missing values
> sum(is.na(loan_data))
[1] 86
> # Summary of the key columns
> summary(loan_data)
   Loan_ID             Gender             Married            Dependents
 Length:614         Length:614         Length:614         Length:614
 Class :character   Class :character   Class :character   Class :character
```










Marital Status Distribution
No
Yes

11. The following table shows the number of units of different products sold on different days

Product	Monday	Tuesday	Wednesday	Thursday	Friday
Bread	12	3	5	11	9
Milk	21	27	18	20	15
Cola Cans	10	1	33	6	12
Chocolate bars	6	7	4	13	12
Detergent	5	8	12	20	23

(i) Create five sample numeric vectors from this data.

(ii) Name and explain the operators used to form data subsets in R.

```
# Create Five Sample Numeric Vectors from the Given Data
bread_sales <- c(12, 3, 5, 11, 9)
milk_sales <- c(21, 27, 18, 20, 15)
cola_cans_sales <- c(10, 1, 33, 6, 12)
chocolate_bars_sales <- c(6, 7, 4, 13, 12)
detergent_sales <- c(5, 8, 12, 20, 23)

print("Bread Sales:")
print(bread_sales)

print("Milk Sales:")
print(milk_sales)

print("Cola Cans Sales:")
print(cola_cans_sales)

print("Chocolate Bars Sales:")
print(chocolate_bars_sales)

print("Detergent Sales:")
print(detergent_sales)

# Operators Used to Form Data Subsets in R

# 1 Sqaure Bracket

bread_subset <- bread_sales[1:3]
print(bread_subset)

# 2 Dollar Sign
```

```r
sales_df <- data.frame(
  Day = c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday"),
  Bread = bread_sales,
  Milk = milk_sales,
  ColaCans = cola_cans_sales,
  ChocolateBars = chocolate_bars_sales,
  Detergent = detergent_sales
)

milk_column <- sales_df$Milk
print(milk_column)

# 3 Subset Function subset()

milk_high_sales <- subset(sales_df, Milk > 20)
print(milk_high_sales)

# 4 Logical operators

bread_high_sales <- bread_sales[bread_sales > 5]
print(bread_high_sales)

# 5 Negative Indices

cola_cans_subset <- cola_cans_sales[-c(1,3)]
print(cola_cans_subset)
```

```
> print("Bread Sales:")
[1] "Bread Sales:"
> print(bread_sales)
[1] 12  3  5 11  9
>
> print("Milk Sales:")
[1] "Milk Sales:"
> print(milk_sales)
[1] 21 27 18 20 15
>
> print("Cola Cans Sales:")
[1] "Cola Cans Sales:"
> print(cola_cans_sales)
[1] 10  1 33  6 12
>
> print("Chocolate Bars Sales:")
[1] "Chocolate Bars Sales:"
> print(chocolate_bars_sales)
[1]  6  7  4 13 12
>
> print("Detergent Sales:")
[1] "Detergent Sales:"
> print(detergent_sales)
[1]  5  8 12 20 23

> # Operators Used to Form Data Subsets in R
>
> # 1 Sqaure Bracket
>
> bread_subset <- bread_sales[1:3]
> print(bread_subset)
[1] 12  3  5
>
> # 2 Dollar Sign
>
> sales_df <- data.frame(
+   Day = c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday"),
+   Bread = bread_sales,
+   Milk = milk_sales,
+   ColaCans = cola_cans_sales,
+   ChocolateBars = chocolate_bars_sales,
+   Detergent = detergent_sales
+ )
>
> milk_column <- sales_df$Milk
> print(milk_column)
[1] 21 27 18 20 15
>
> # 3 Subset Function subset()
>
> milk_high_sales <- subset(sales_df, Milk > 20)
> print(milk_high_sales)
      Day Bread Milk ColaCans ChocolateBars Detergent
1  Monday    12   21       10             6         5
2 Tuesday     3   27        1             7         8
>

>
> # 4 Logical operators
>
> bread_high_sales <- bread_sales[bread_sales > 5]
> print(bread_high_sales)
[1] 12 11  9
>
> # 5 Negative Indices
>
> cola_cans_subset <- cola_cans_sales[-c(1,3)]
> print(cola_cans_subset)
[1]  1  6 12
```

12. Which function is used to concatenate text values in R. Write a script to concatenate text and numerical values in R.

Text 1: Ram has scored

Text 2: 89

Text 3: marks

Text 4: in Mathematics

```
text1 <- "Ram has scored"
text2 <- 89
text3 <- "marks"
text4 <- "in Mathematics"

result <- paste(text1, text2, text3, text4)

print(result)
```

```
> text1 <- "Ram has scored"
> text2 <- 89
> text3 <- "marks"
> text4 <- "in Mathematics"
>
> result <- paste(text1, text2, text3, text4)
>
> print(result)
[1] "Ram has scored 89 marks in Mathematics"
>
```

13. Consider the following data frame given below:

course	id	class	marks
1	11	1	56
2	12	2	75
3	13	1	48
4	14	2	69
5	15	1	84
6	16	2	53

i. Create a subset of course less than 3 by using [ ] brackets and demonstrate the output.

ii. Create a subset where the course column is less than 3 or the class equals to 2 by using subset () function and demonstrate the output.

```
df <- data.frame(
  course = c(1,2,3,4,5,6),
  id= c(11,12,13,14,15,16),
  class = c(1,2,1,2,1,2),
  marks = c(56,75,48,69,84,53)
)

print(df)

# Create a Subset of course Less than 3 Using Square Brackets []

course_subset = df[df$course < 3, ]
print(course_subset)

# Create a subset where the course column is less than 3 or the class
equals to 2 by using subset () function and demonstrate the output.

subset_course_class <- subset(df, course < 3 | class == 2)
print(subset_course_class)
```

```
> print(df)
  course id class marks
1      1 11     1    56
2      2 12     2    75
3      3 13     1    48
4      4 14     2    69
5      5 15     1    84
6      6 16     2    53
>
> # Create a Subset of course Less than 3 Using Square Brackets []
>
> course_subset = df[df$course < 3, ]
> print(course_subset)
  course id class marks
1      1 11     1    56
2      2 12     2    75
>
> # Create a subset where the course column is less than 3 or the class equals $
>
> subset_course_class <- subset(df, course < 3 | class == 2)
> print(subset_course_class)
  course id class marks
1      1 11     1    56
2      2 12     2    75
4      4 14     2    69
6      6 16     2    53
>
```

i. Create a data frame from the following 4 vectors and demonstrate the output:

emp_id = c (1:5)
emp_name = c("Rick","Dan","Michelle","Ryan","Gary")
start_date = c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11", "2015-03-27")
salary = c(60000, 45000, 75000, 84000, 20000)

ii. Display structure and summary of the above data frame.

iii. Extract the emp_name and salary columns from the above data frame.

iv. Extract the employee details whose salary is less than or equal to 60000.

```
df = data.frame(
  emp_id = c(1:5),
  emp_name = c("Rick", "Dan", "Michelle", "Ryan", "Gary"),
  start_date = c("2012-01-01", "2013-09-23", "2014-11-15",
"2014-05-11",
         "2015-03-27"),
  salary = c(60000, 45000, 75000, 84000, 20000)
)

# display structure and summary

print(df)
summary(df)

# extract emp_name and salary column

emp_salary_data <- df[, c("emp_name", "salary")]
print("Employee Name and Salary columns:")
print(emp_salary_data)

salary_emp <- subset(df, salary <= 60000)
print("Employees with salary less than or equal to 60000:")
print(salary_emp)
```

```
> # display structure and summary
>
> print(df)
  emp_id emp_name start_date salary
1      1     Rick 2012-01-01  60000
2      2      Dan 2013-09-23  45000
3      3 Michelle 2014-11-15  75000
4      4     Ryan 2014-05-11  84000
5      5     Gary 2015-03-27  20000
> summary(df)
     emp_id    emp_name          start_date            salary
 Min.   :1   Length:5           Length:5           Min.   :20000
 1st Qu.:2   Class :character   Class :character   1st Qu.:45000
 Median :3   Mode  :character   Mode  :character   Median :60000
 Mean   :3                                         Mean   :56800
 3rd Qu.:4                                         3rd Qu.:75000
 Max.   :5                                         Max.   :84000
>
> # extract emp_name and salary column
>
> emp_salary_data <- df[, c("emp_name", "salary")]
> print("Employee Name and Salary columns:")
[1] "Employee Name and Salary columns:"
> print(emp_salary_data)
  emp_name salary
1     Rick  60000
2      Dan  45000
3 Michelle  75000
4     Ryan  84000
5     Gary  20000
>

> salary_emp <- subset(df, salary <= 60000)
> print("Employees with salary less than or equal to 60000:")
[1] "Employees with salary less than or equal to 60000:"
> print(salary_emp)
  emp_id emp_name start_date salary
1      1     Rick 2012-01-01  60000
2      2      Dan 2013-09-23  45000
5      5     Gary 2015-03-27  20000
>
>
```

15

ii. Suppose you have two datasets A and B.
Dataset A has the following data: 6 7 8 9.
Dataset B has the following data: 1 2 4 5.
Which function is used to combine the data from both datasets into dataset C.
Demonstrate the function with the input values and write the output.

```
A <- c(6, 7, 8, 9)
B <- c(1, 2, 4, 5)

C <- c(A, B)

print("Combined Dataset C:")
print(C)
```

```
> print("Combined Dataset C:")
[1] "Combined Dataset C:"
> print(C)
[1] 6 7 8 9 1 2 4 5
> 
```

Implement Bloom Filter using R Programming.

```r
dataset <- c("orange", "apple", "watermelon")
m <- 10
bitarray <- rep(0, m)

hash_func1 <- function(x,m) {
    return(sum(as.integer(charToRaw(x))) %% 5))
}

hash_func2 <- function(x,m) {
    return((2*sum((as.integer(charToRaw(x))) +3) %% 5))
}

for(data in dataset) {
    h1 <- hash_func1(data, m)
    h2 <- hash_func2(data, m)

    bitarray[h1 + 1] <- 1
    bitarray[h2 + 1] <- 1

    cat("Hash 1: ", h1, "Hash 2: ", h2, "\n")
}

print(bitarray)

input_text = "orange"
input_h1 <- hash_func1(input_text, m)
input_h2 <- hash_func2(input_text, m)

cat("Hash1 of input text: ", input_h1, "\n")
cat("Hash2 of input text: ", input_h2, "\n")

if(bitarray[input_h1 + 1] == 1 & bitarray[input_h2 + 1] == 1) {
    cat("The element may be present!")
} else{
    cat("The element is not present!")
}
```

```
> dataset <- c("orange", "apple", "watermelon")
> m <- 10
> bitarray <- rep(0, m)
>
> hash_func1 <- function(x,m) {
+ return(abs(digest::digest2int(x) %% 5))
+ }
>
> hash_func2 <- function(x,m) {
+ return(abs((2 * digest::digest2int(x)) %% m))
+ }
>
> for(data in dataset) {
+ h1 <- hash_func1(data, m)
+ h2 <- hash_func2(data, m)
+
+ bitarray[h1 + 1] <- 1
+ bitarray[h2 + 1] <- 1
+
+ cat("Hash 1: ", h1, "Hash 2: ", h2, "\n")
+ }
```

```
+
+ cat("Hash 1: ", h1, "Hash 2: ", h2, "\n")
+ }
Hash 1:  4 Hash 2:  8
Hash 1:  0 Hash 2:  0
Hash 1:  3 Hash 2:  6
>
> print(bitarray)
 [1] 1 0 0 1 1 0 1 0 1 0
>
> input_text = "orange"
> input_h1 <- hash_func1(input_text, m)
> input_h2 <- hash_func2(input_text, m)
>
> cat("Hash1 of input text: ", input_h1, "\n")
Hash1 of input text:  4
> cat("Hash2 of input text: ", input_h2, "\n")
Hash2 of input text:  8
>
> if(bitarray[input_h1 + 1] == 1 & bitarray[input_h2 + 1] == 1) {
+ cat("The element may be present!")
+ } else{
+ cat("The element is not present!")
+ }
The element may be present!>
```

7 Implement FM algorithm using R Programming.

```
x <- c(3, 1, 4, 1, 5, 9, 2, 6, 5)

# hashing function

hash_array <- sapply(x, function (i) (2*i+1) %% 32)
cat("hashed function: ", hash_array, "\n")

# convert to binary

int_to_binary <- function(n) {
    binary <- paste(rev(as.integer(intToBits(n))), collapse="")
    sub("^0+", "", binary)
}

hash_binary <- sapply(hash_array, int_to_binary)
cat("binary array ", hash_binary, "\n")

# tail length

trailing_zeros <- sapply(hash_binary, function(i) nchar(i) -
nchar(sub("0+$", "", i)))
cat("trailing zeros ", trailing_zeros, "\n")

# Find the maximum number of trailing zeros

max_trailing_zeros <- max(trailing_zeros)
cat("maximum trailing zeros ", max_trailing_zeros, "\n")

# Estimate the number of unique elements

unique_estimate <- 2^max_trailing_zeros
print(paste("number of unique elements: ", unique_estimate ))
```

```
>
> # hashing function
>
> hash_array <- sapply(x, function (i) (2*i+1) %% 32)
> cat("hashed function: ", hash_array, "\n")
hashed function:  7 3 9 3 11 19 5 13 11
>
> # convert to binary
>
> int_to_binary <- function(n) {
+ binary <- paste(rev(as.integer(intToBits(n))), collapse="")
+ sub("^0+", "", binary)
+ }
>
> hash_binary <- sapply(hash_array, int_to_binary)
> cat("binary array ", hash_binary, "\n")
binary array  111 11 1001 11 1011 10011 101 1101 1011
>
> # tail length
>
> trailing_zeros <- sapply(hash_binary, function(i) nchar(i) - nchar(sub("0+$",$
> cat("trailing zeros ", trailing_zeros, "\n")
trailing zeros  0 0 0 0 0 0 0 0 0
>
> # Find the maximum number of trailing zeros
```

```
+ }
>
> hash_binary <- sapply(hash_array, int_to_binary)
> cat("binary array ", hash_binary, "\n")
binary array  111 11 1001 11 1011 10011 101 1101 1011
>
> # tail length
>
> trailing_zeros <- sapply(hash_binary, function(i) nchar(i) - nchar(sub("0+$",$
> cat("trailing zeros ", trailing_zeros, "\n")
trailing zeros  0 0 0 0 0 0 0 0 0
>
> # Find the maximum number of trailing zeros
>
> max_trailing_zeros <- max(trailing_zeros)
> cat("maximum trailing zeros ", max_trailing_zeros, "\n")
maximum trailing zeros  0
>
> # Estimate the number of unique elements
>
> unique_estimate <- 2^max_trailing_zeros
> print(paste("number of unique elements: ", unique_estimate ))
[1] "number of unique elements:  1"
>
> 
```