1. Write an R program for different types of data structures in R.

```
# Vectors
my_vector <- c(1, 2, 3, 4, 5)
print("Numeric Vector")
print(my_vector)

cv<-c("apple", "banana", "cherry")
print("Character Vector")  print(cv)

# Matrices
my_matrix <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3,
byrow=TRUE)
print("Matrix")
print(my_matrix)

# Lists
my_list <- list(name = "John", age = 25, scores = c(80, 90, 95))
print("List")
print(my_list)


# Data Frames
my_data <- data.frame(name = c("mohan", "Jameel",
        "Mehran","Ganesh"), age = c(25, 30, 35, 7), scores = c(80,
        90, 85,90))
print("Data Frame")
print(my_data)
class(my_vector)
```

Output:

```
[1] "Numeric Vector"
[1] 1 2 3 4 5
[1] "Character Vector"
[1] "apple"  "banana" "cherry"
[1] "Matrix"
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[1] "List"
$name
```

```
[1] "Mohan"

$age
[1] 25

$scores
[1] 80 90 95

[1] "Data Frame"
    name age scores
1   Mohan   25   80
2   Jameel  30   90
3   Mehran  35   85
4   Ganesh   7   90
```

2. Write an R program that includes variables, constants, data types.

```
# Variables
name <- "John"
age <- 25
score <- 95.5

# Constants
PI <- 3.14159
GRAVITY<-9.81

# Data Types
is_student <- TRUE
grades <- c(80, 90, 85)
student_info <- list(name = "Hasan", age = 25, scores = grades)

# Printing the values
cat("Data type of 'name':",(typeof(name)),"\n")
cat("Data type of 'age':",(typeof(age)),"\n")
cat("Data type of 'score':",(typeof(score)),"\n")
cat("Data type of 'is_Student':",(typeof(is_student)),"\n")
cat("Data type of 'grades':",(typeof(grades)),"\n")
cat("Data type of 'Student_Info':",(typeof(student_info)),"\n")
cat("Constant Value of 'PI':",PI,"\n")
cat("Constant Value of 'GRAVITY':",GRAVITY,"\n")
```

Output:

```
Data type of 'name': character
Data type of 'age': double
Data type of 'score': double
Data type of 'is_Student': logical
Data type of 'grades': double
Data type of 'Student_Info': list
Constant Value of 'PI': 3.14159
Constant Value of 'GRAVITY': 9.81
```

3. Write an R program that include different operators, control structures, default values for arguments, returning complex objects.

```
#Function with default values for arguments
calculate_area <-function(radius=5,
shape='circle'){ if(shape=='circle'){
area<-pi*radius*2
}else
if(shape=='square'){ area
<-radius*2
}else{
cat("Unsupported shape", shape, "\n")
"return(NULL)
}

#Conditional Operator
msg<-ifelse(area>10,"Large Area", "Small Area")

#Returning complex object
return(list(
shape=shape,
radius=radius,
area=area
Msg=msg
))
}

#usage
circle_result<-calculate_area() #using default values
square_result<-calculate_area(radius=4, shape='square')

#display result
cat("Circle Result","\n")
print(circle_result)

cat("Square Result","\n")
print(square_result)
```

Output:

```
Circle Result
$shape
[1] "circle"

$radius
```

[1] 5

```
$area
[1] 31.41593

$Msg
[1] "Large Area"

Square Result
$shape
[1] "square"

$radius
[1] 4

$area
[1] 8

$Msg
[1] "Small Area"
```

4. Write an R program for quick sort implementation, binary search tree.

```
# Function to perform Quick Sort
quickSort <- function(arr) {
if (length(arr) <= 1)
{ return(arr)
}
pivot <- arr[1]
smaller <- arr[arr < pivot]
equal <- arr[arr == pivot]
greater <- arr[arr > pivot]
return(c(quickSort(smaller), equal, quickSort(greater)))
}

# Example usage Quick Sort
print("Quick Sort")
print("Before Sort")
my_array <- c(5, 2, 8, 3, 1, 9)
print(my_array)
sorted_array <- quickSort(my_array)
print("After Sort") print(sorted_array)
#Binary Search Tree:

# Define a Node structure for Binary Search Tree
Node <- function(value) {
list(
value = value,
left = NULL,
right = NULL
)
}
# Function to insert a value into Binary Search Tree
insert <- function(root, value) {
if (is.null(root))
{ return(Node(value))
}
if (value < root$value) {
root$left <- insert(root$left, value)
} else if (value > root$value)
{ root$right <- insert(root$right, value)
}
```

```
return(root)
}

# Function to perform Inorder traversal of Binary Search Tree
inorder <- function(root) {
if (!is.null(root))
{ inorder(root$left)
print(root$value)
inorder(root$right)
}
}

# Example usage BST
print("Binary Search Tree")
my_tree <- NULL
keys<-c(5,2,8,3,1,9)
for(key in keys){ my_tree <- insert(my_tree, key)
}
inorder(my_tree)
```

Output:

```
[1] "Quick Sort"
[1] "Before Sort"
[1] 5 2 8 3 1 9
[1] "After Sort"
[1] 1 2 3 5 8 9
[1] "Binary Search Tree"
[1] 1
[1] 2
[1] 3
[1] 5
[1] 8
[1] 9
```

5. Write an R program for calculating cumulative sums and products, minima, maxima and calculus.

```
# Create a vector of numbers
numbers <- c(2, 4, 1, 8, 5, 7)

# Calculate the cumulative sum
cumulative_sum <- cumsum(numbers)
cat("Cummulative Sum:",cumulative_sum,"\n")

# Calculate the cumulative product
cumulative_product <- cumprod(numbers)
cat("Cummulative Product:",cumulative_product,"\n")

# Find the minimum and maximum values
minimum_value <- min(numbers)
maximum_value <- max(numbers)
cat("Minimum Value", minimum_value, "\n")
cat("Maximum Value", maximum_value, "\n")

# Calculus Differentiation
differentiate<-diff(numbers)
cat("Differentiation (First Difference):", differentiate,"\n")

# Calculus Integration
integrate<-cumsum(numbers)
cat("Integration (Cummulative Sum):",integrate, "\n")
```

Output:

```
Original Vector: 1 2 3 4 5
Cumulative Sums: 1 3 6 10 15
Cumulative Products: 1 2 6 24 120
Minimum Value: 1
Maximum Value: 5
Derivative (Calculus): NA 1 1 1 1

Accessing Results:
Original Vector: 1 2 3 4 5
Cumulative Sums: 1 3 6 10 15
Cumulative Products: 1 2 6 24 120
Minimum Value: 1
Maximum Value: 5
Derivative (Calculus): NA 1 1 1 1
```

6. Write an R program for finding stationary distribution of markanov chains.

```
# Install and load the markovchain package
install.packages("markovchain")
library(markovchain)

# Create a transition matrix for the Markov chain
transition_matrix <- matrix(c(0.7, 0.3, 0.2, 0.8), nrow = 2, byrow = TRUE)

# Create a markovchain object
mc <- new("markovchain", states = c("State1", "State2"), transitionMatrix =
transition_matrix)

# Find the stationary distribution
stationary_dist <- steadyStates(mc)

# Print the stationary distribution
print(stationary_dist)
```

Output:

```
     State1 State2
[1,]    0.4    0.6
```

7. Write an R program that includes linear algebra operations on vectors and matrices

```
# Create vectors

vector1 <- c(1, 2, 3)
vector2 <- c(4, 5, 6)

# Perform vector addition
vector_sum <- vector1 + vector2
cat("Vector Addition:", "\n")
cat(vector_sum)

# Perform vector subtraction
vector_diff <- vector1 - vector2
cat("\n Vector Subtraction:\n")
cat(vector_diff)

# Perform vector multiplication
scalar<-2
vector_scalar_product <- scalar*vector1
cat("\n Vector Scalar Product:\n")
cat(vector_scalar_product)

# Create matrices
matrix1 <- matrix(c(1, 2, 3, 4), nrow = 2, ncol = 2)
matrix2 <- matrix(c(5, 6, 7, 8), nrow = 2, ncol = 2)

# Perform matrix addition
matrix_sum <- matrix1 + matrix2
cat("\n Matrix Addition:\n")
cat(matrix_sum)

# Perform matrix subtraction
matrix_diff <- matrix1 - matrix2
cat("\n Matrix Subtraction:\n")
cat(matrix_diff)

# Perform matrix multiplication
matrix_product <- matrix1 %*%(matrix2)
cat("\n Matrix Multiplication:\n")
cat(matrix_product)

# Calculate matrix determinant
matrix_det <- det(matrix1)
cat("\n Matrix Determinant:\n")
```

```
cat(matrix_det)

#Calculate Matrix Transpose
matrix_transpose<-t(matrix1)
cat("\n Matrix Transpose:\n")
cat(matrix_transpose)

# Calculate matrix inverse
matrix_inv <- solve(matrix1)
print("\n Matrix Inverse:")
print(matrix_inv)
```

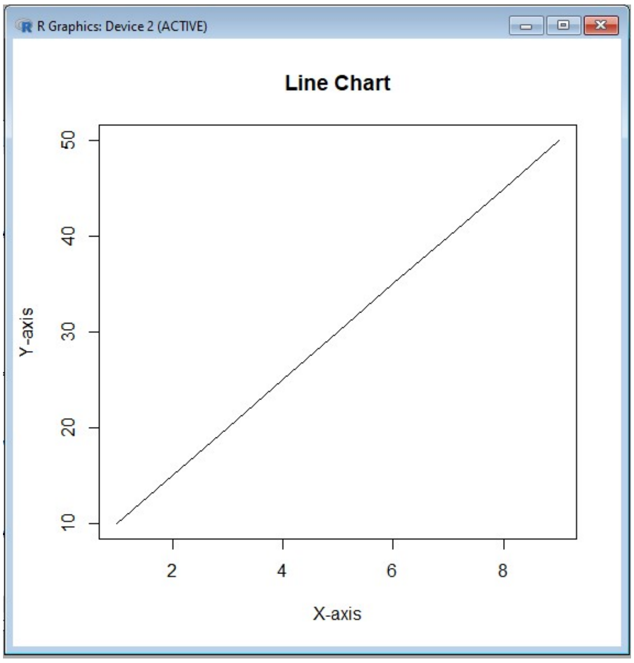
Output:

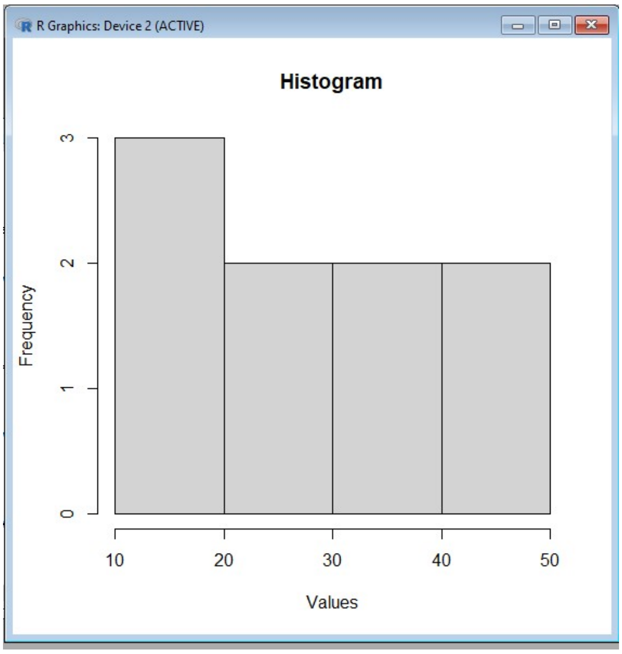
```
Vector Addition:
5 7 9
 Vector Subtraction:
-3 -3 -3
 Vector Scalar Product:
2 4 6
 Matrix Addition:
6  8
10 12
 Matrix Subtraction:
-4 -4
-4 -4
 Matrix Multiplication:
23 34
31 46
 Matrix Determinant:
-2
 Matrix Transpose:
1 3 2 4
[1]  Matrix Inverse:"
     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
```

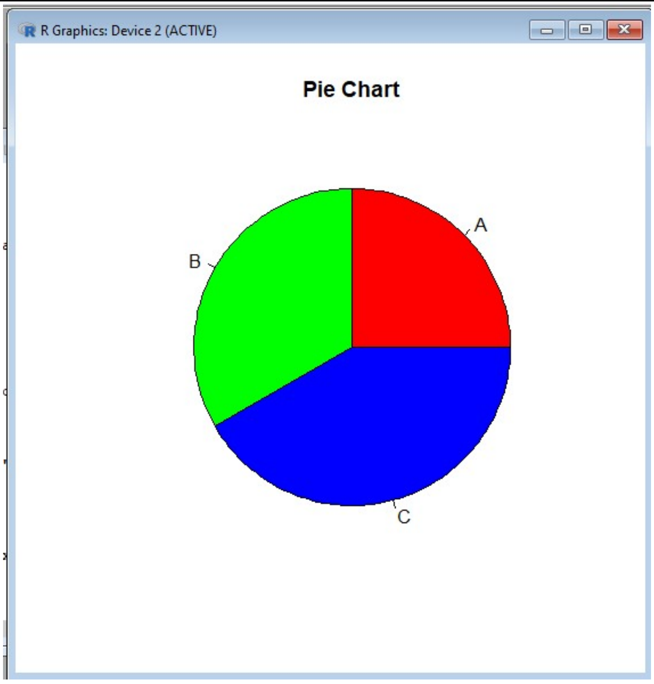
8. Write a R program for any visual representation of an object with creating graphs using graphic functions: Plot(), Hist(), Linechart(), Pie(), Boxplot(), Scatter plots().

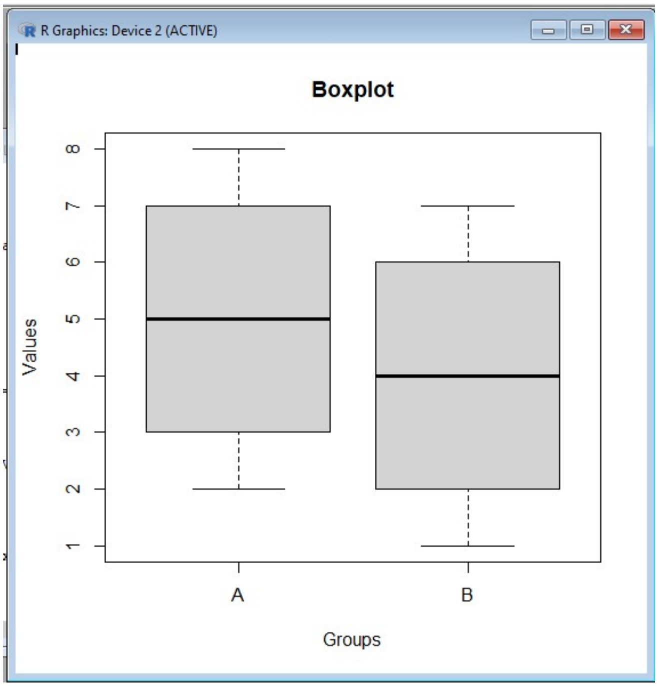
```
# Creating a vector of data
data <- c(10, 15, 20, 25, 30, 35, 40, 45, 50)

# Plotting a line chart
plot(data, type = "l", main = "Line Chart", xlab = "X-axis", ylab = "Y-axis")

# Creating a histogram
hist(data, main = "Histogram", xlab = "Values", ylab = "Frequency")

# Creating a pie chart
categories<-c("A", "B", "C")
values<-c(30, 40, 50)
pie(values, labels=categories, main = "Pie Chart", col=c("red", "green",
"blue"))

# Creating a boxplot
data<-list(A=c(2, 4, 6, 8), B=c(1, 3, 5, 7))
boxplot(data, main = "Boxplot", xlab="Groups", ylab = "Values")

# Creating a scatter plot
x <- c(1, 2, 3, 4, 5)
y <- c(10, 20, 30, 20, 50)
plot(x, y, pch=19, col="blue", main = "Scatter Plot", xlab = "X-axis", ylab =
"Y-axis")
```

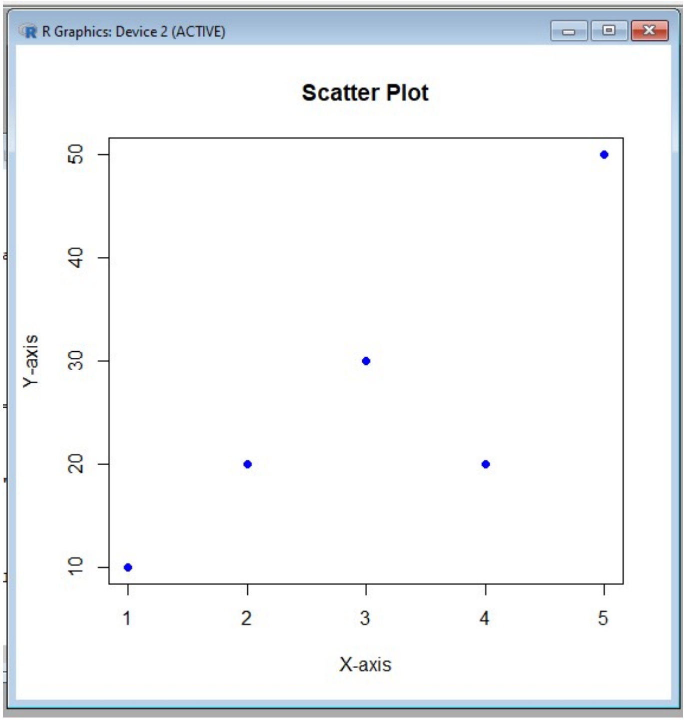
Output:


R Graphics: Device 2 (ACTIVE)
Line Chart
Y-axis
10
20
30
40
50
2
4
6
8
X-axis


R Graphics: Device 2 (ACTIVE)
Histogram
Frequency
0
1
2
3
10
20
30
40
50
Values


R Graphics: Device 2 (ACTIVE)
Pie Chart
A
B
C


R Graphics: Device 2 (ACTIVE)
Boxplot
Values
1
2
3
4
5
6
7
8
A
B
Groups


R Graphics: Device 2 (ACTIVE)
Scatter Plot
Y-axis
10
20
30
40
50
1
2
3
4
5
X-axis

9. Write an R program for with any dataset containing data frame objects, indexing and sub-setting data frames and employ manipulating and analyzing data.

```
#Create a sample employee dataset as a data frame
emp_data<-data.frame(
EmployeeId=c(1,2,3,4,5),
FirstName=c("Talooth","Alia","Bobby","Carol","David"),
LastName=c("maulim","Bhat","deol","dias","Warner"),
Age=c(30,25,28,35,32),
Department=c("HR","Marketing","Finance","HR","IT"),
Salary=c(50000,55000,60000,52000,70000)
)
#Print the entire employee dataset
cat("Employee Data:\n")
print(emp_data)
#Sunset and index the data frame
cat("\nSubset and Indexing")
#Select employees in the HR department
hr_emp<-emp_data[emp_data$Department=="HR",]
cat("HR Employees:\n")
print(hr_emp)

#Select employees aged 30 or older
old_emp<-emp_data[emp_data$Age>=30,]
cat("Employee aged 30 or Older:\n")
print(old_emp)
#Select employees with salary greater than $55000
high_sal_emp<-emp_data[emp_data$Salary>=55000,]
cat("Employee aged 30 or Older:\n")
print(high_sal_emp)
```

```
#Manipulate and analyze the data
cat("\nData Manipulation and Analysis:\n")
#Calculate the average salary
avg_sal<-mean(emp_data$Salary)
cat("Average Salary:",avg_sal,"\n")
#Calculate the Maximum age
max_age<-max(emp_data$Age)
cat("\nMaximum Age:",max_age,"\n")

#Calculate the no. of employees in each department
dept_count<-table(emp_data$Department)
cat("\nNumber of Employees in each Department")
print(dept_count)
#Calculate the total Payroll for each department
dept_payroll<-tapply(emp_data$Salary, emp_data$Department, sum)
cat("Total Payroll in Each Department:\n")
print(dept_payroll)
```

Output:

Employee Data:

```
  EmployeeId FirstName LastName Age Department Salary
1          1   Talooth   Maulim  30         HR  50000
2          2      Alia     Bhat  25  Marketing  55000
3          3     Bobby     Deol  28    Finance  60000
4          4     Carol     dias  35         HR  52000
5          5     David   warner  32         IT  70000
```

Subset and Indexing HR Employees:

```
  EmployeeId FirstName LastName Age Department Salary
1          1   Talooth   maulim  30         HR  50000
4          4     Carol     dias  35         HR  52000
```

Employee aged 30 or Older:

	EmployeeId	FirstName	LastName	Age	Department	Salary
1	1	Jhon	Smith	30	HR	50000
4	4	Carol	Smith	35	HR	52000
5	5	David	Davis	32	IT	70000

Employee aged 30 or Older:

	EmployeeId	FirstName	LastName	Age	Department	Salary
2	2	Alia	bhat	25	Marketing	55000
3	3	Bobby	deol	28	Finance	60000
5	5	David	warner	32	IT	70000

Data Manipulation and Analysis:
Average Salary: 57400

Maximum Age: 35

Number of Employees in each Department

Finance	HR	IT	Marketing
1	2	1	1

Total Payroll in Each Department:

Finance	HR	IT	Marketing
60000	102000	70000	55000

10. Write a program to create any application of Linear Regression in multivariate context for predictive purpose.

```
#Sample dataset: Salary, Years of Experience. Education Level
data<-
data.frame( Salary=c(50000,60000,75000,80000,95000,110000,1200
00,130000),
Experience=c(1,2,3,4,5,6,7,8),
Education=c(12,14,16,16,18,20,20,22)
)
#Perform multivariate linear regression
model<-lm(Salary~Experience+Education,data=data)
model
#Predict salaries for new data
new_data<-
data.frame( Experience=c(9,1
0), Education=c(22,24)
)
predicted_salaries<-predict(model,newdata=new_data)
#Print the predicted salaries
cat("Predicted Salaries:\n")
print(predicted_salaries)
```

Output:

```
Predicted Salaries:
       1        2
139333.3 152500.0
```