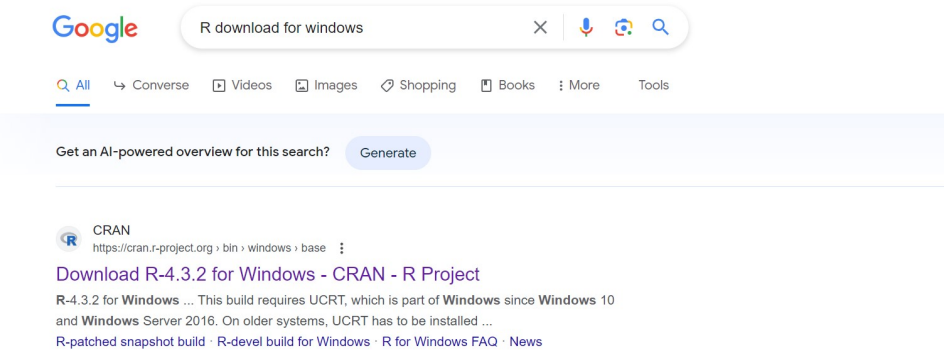
1 Experiment 0

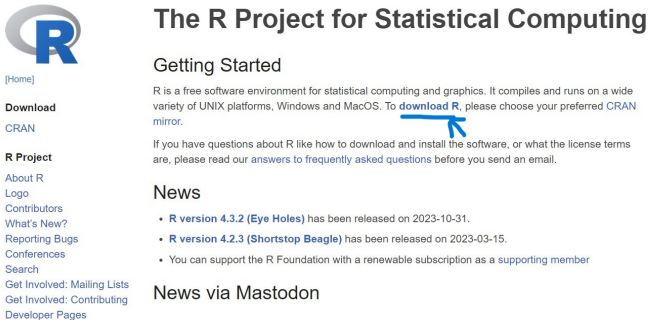
Demonstrate the steps for installation of R and R Studio. Perform the following:

1. **Installation of R on windows**

   - **Step 1**: Google RStudio Website

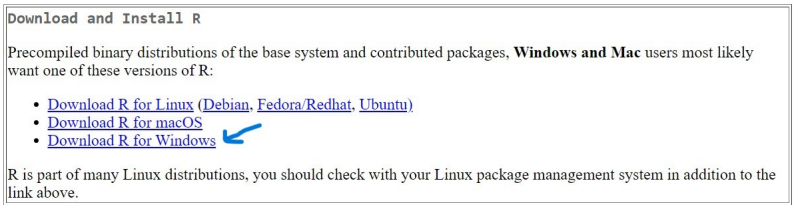


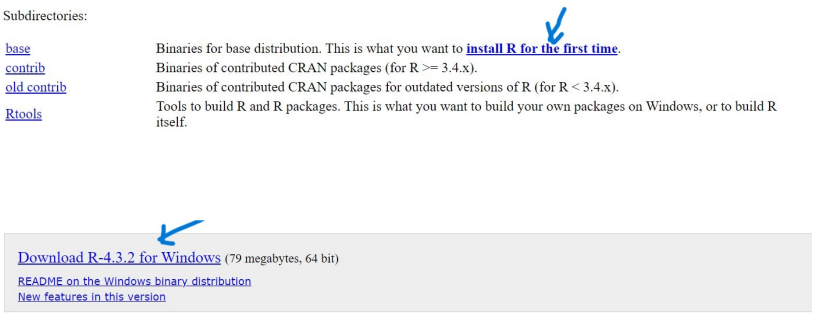
   - **Step 2**: Click on Download R.



   - **Step 3**: Click on the link under India as show below.

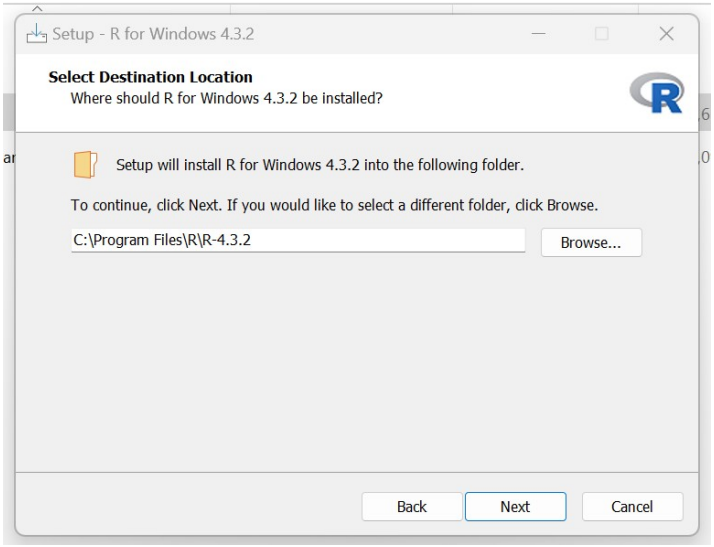
   - **Step 4**: Click on the below link as shown in figure below. Download R for Windows.

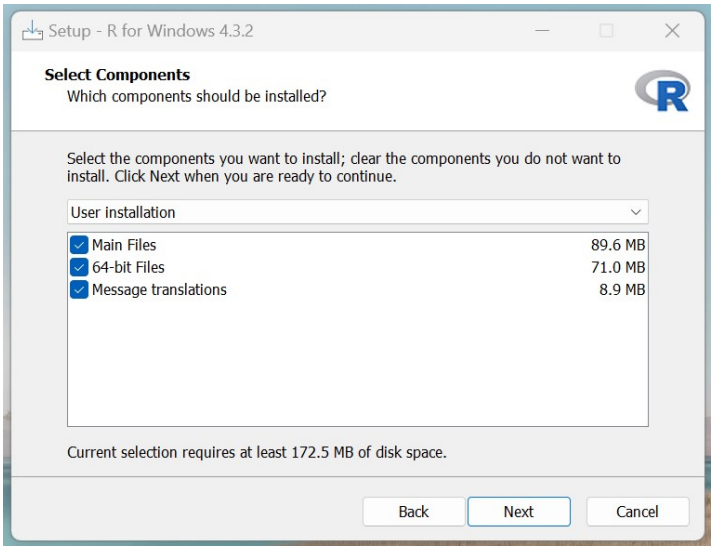


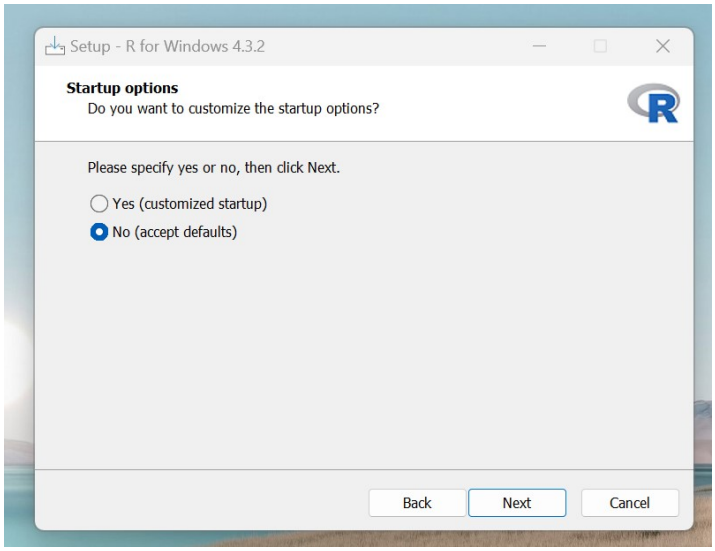


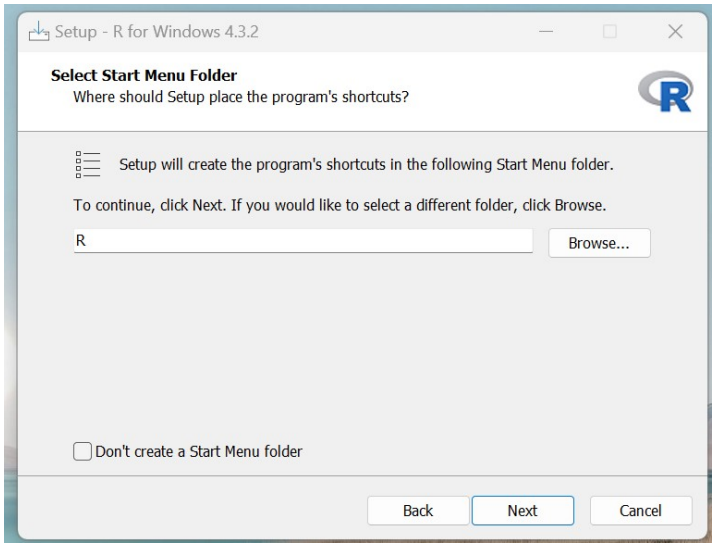
- **Step 5**: Now double click and Install the R binary file, that is downloaded.

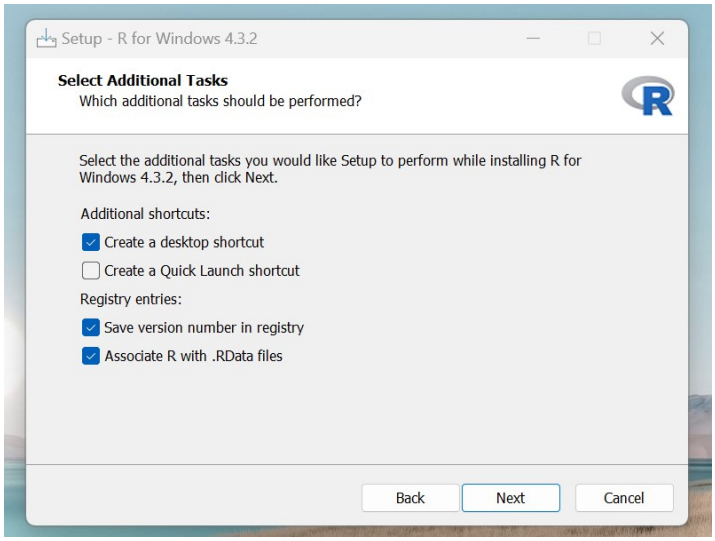
- **Step 6**: Save the binary in the location shown in the figure.







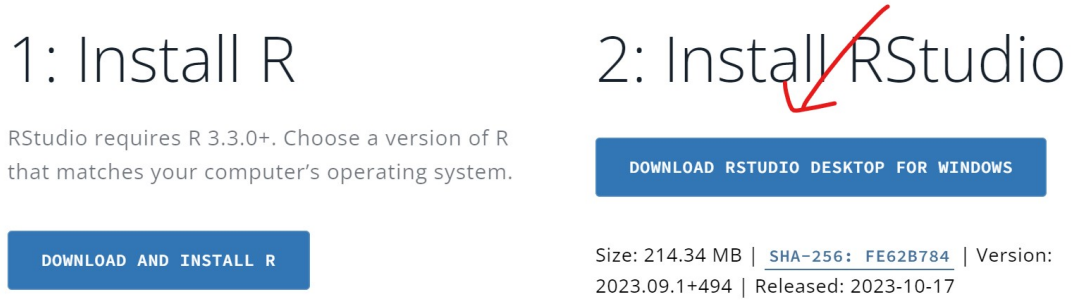




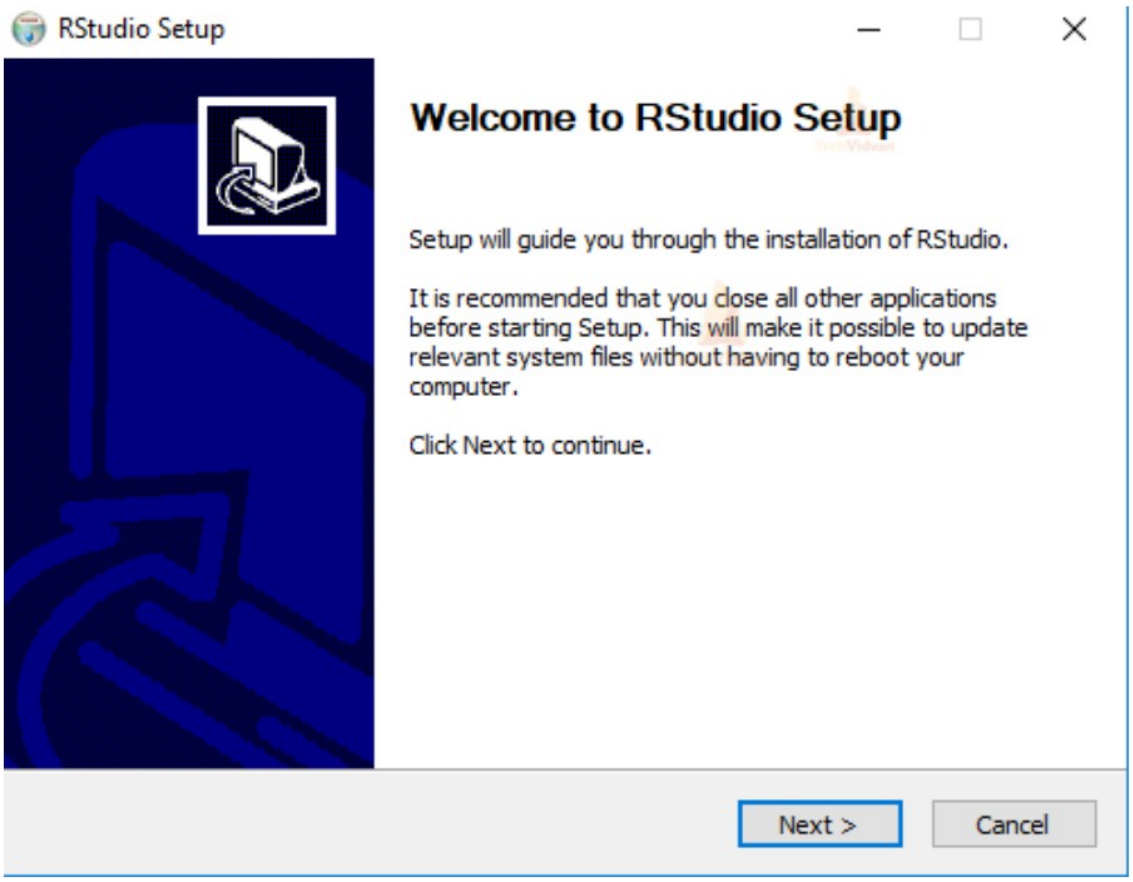
- **Step 7**: Now Click on Finish. Installation is complete.

2. **Installation of R Studio on windows**

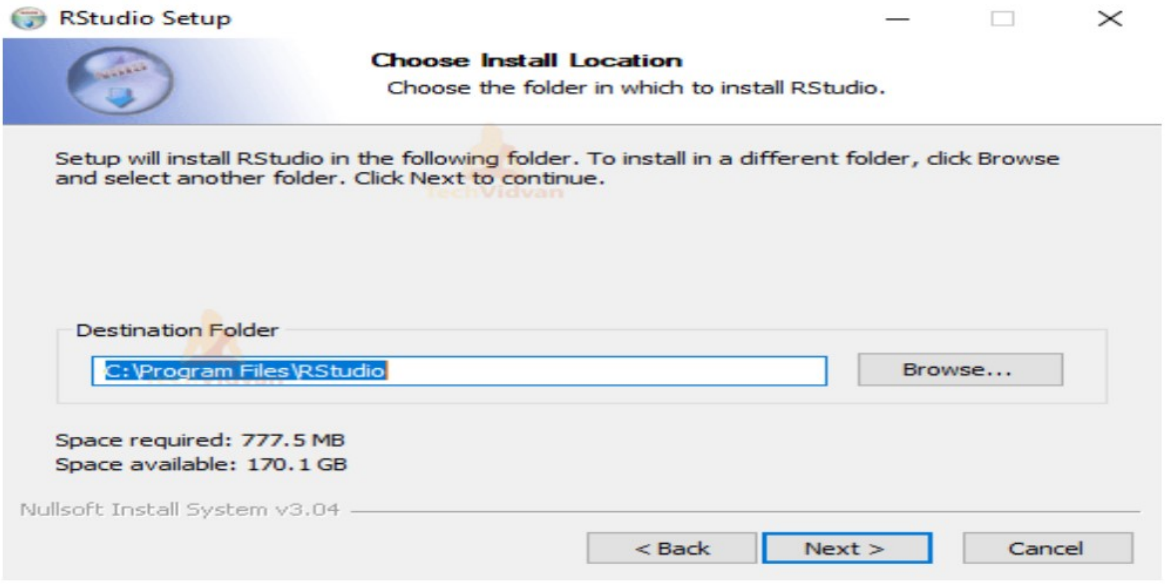
- **Step 1**:With R-base installed, let's move on to installing RStudio. To begin, go to download RStudio and click on the download button for RStudio desktop. Use the below link: https://posit.co/download/rstudio-desktop/



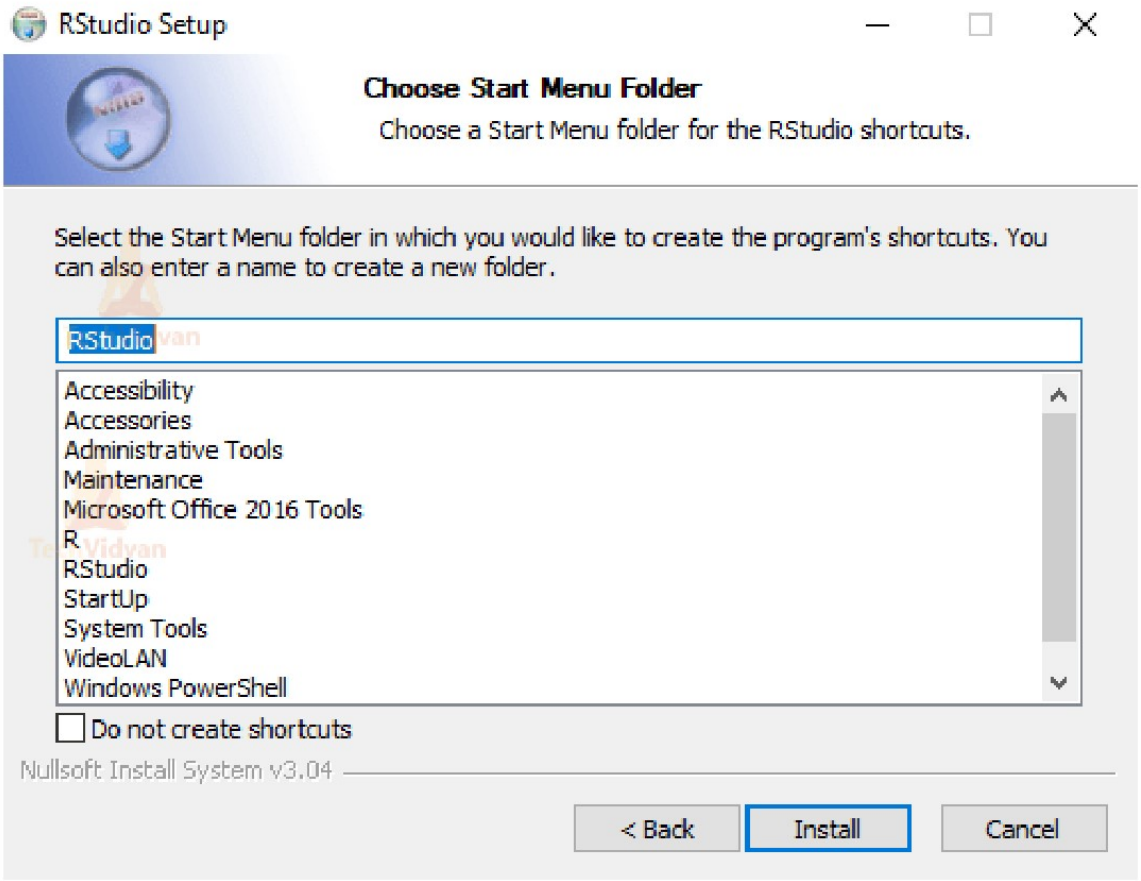
- **Step 2**: Click on the link for the windows version of RStudio and save the .exe file.
- **Step 3**:Run the .exe and follow the installation instructions.
  (a) Click Next on the welcome window.



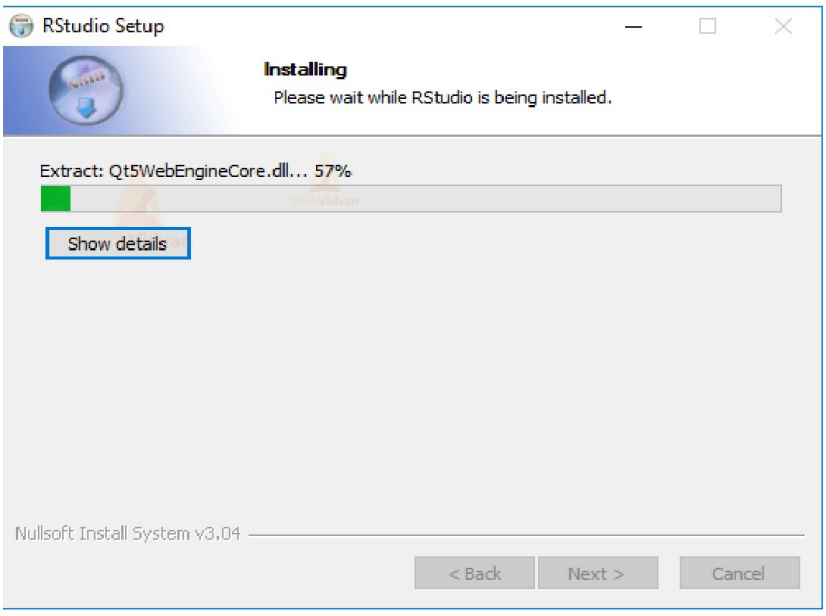
(b) Enter/browse the path to the installation folder and click Next to proceed.



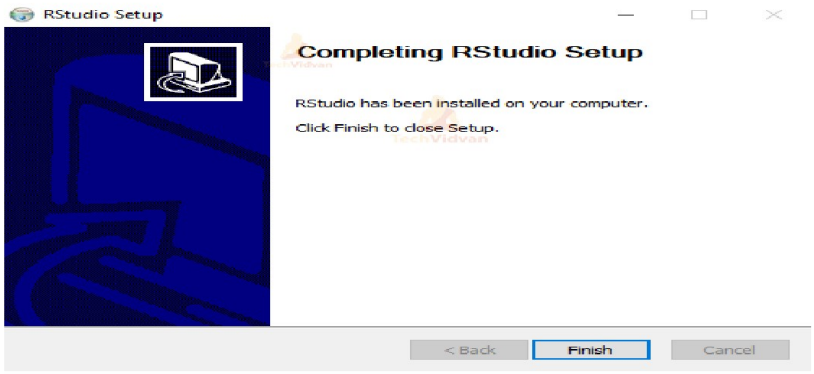
(c) Select the folder for the start menu shortcut or click on do not create shortcuts and then click Next.



(d) Wait for the installation process to complete.



(e) Click Finish to end the installation.



2 Experiment 1

1. **Assign different type of values to variables and display the type of variable. Assign different types such as Double, Integer, Logical, Complex and Character and understand the difference between each data type.**

Basic Data Types	Values	Examples
Numeric	Set of all real numbers	"numeric_value <- 3.14"
Integer	Set of all integers, Z	"integer_value <- 42L"
Logical	TRUE and FALSE	"logical_value <- TRUE"
Complex	Set of complex numbers	"complex_value <- 1 + 2i"
Character	"a", "b", "c", . . . , "@", "1", "2", . . . etc	"character_value <- "Hello Geeks"

(a) **Numeric data type in R (Double, Decimal)**
Decimal values are called numeric in R. It is the default R data type for numbers in R. If you assign a decimal value to a variable x as follows, x will be of numeric type. Real numbers with a decimal point are represented using this data type in R. it uses a format for double-precision floating-point numbers to represent numerical values.

```
# Assign an integer value to y
y = 5
# print the type of variable
print(typeof(y))
[1] "double"

# Assign a decimal value to x
x = 5.6
# print the type of variable
print(typeof(x))
[1] "double"

# Assign a integer value to y
y = 5
# is y an integer?
print(is.integer(y))
[1] FALSE
```

When R stores a number in a variable, it converts the number into a "double" value or a decimal type with at least two decimal places. This means that a value such as "5" here, is stored as 5.00 with a type of double and a class of numeric. And also y is not an integer here can be confirmed with the is.integer() function.

(b) **Integer data type in R**
R supports integer data types which are the set of all integers. You can create as well as convert a value into an integer type using the as.integer() function. You can also use the capital 'L' notation as a suffix to denote that a particular value is of the integer R data type.

```
# Create an integer value
x = as.integer(5)
# print the type of x
print(typeof(x))
[1] "integer"
# Declare an integer by appending an L suffix.
y = 5L
# print the type of y
print(typeof(y))
[1] "integer"
```

(c) **Logical Data type in R**

R has logical data types that take either a value of true or false. A logical value is often created via a comparison between variables. Boolean values, which have two possible values, are represented by this R data type: FALSE or TRUE

```
x = 4
y = 3
# Comparing two values
z = x > y
# print the logical value
print(z)
[1] TRUE"
# print the type of z
print(typeof(z))
[1] "logical"
```

(d) **Complex Data type in R**

R supports complex data types that are set of all the complex numbers. The complex data type is to store numbers with an imaginary component.

```
# Assign a complex value to x
x = 4 + 3i
# print the type of x
print(typeof(x))
[1] "complex"
```

(e) **Character Data type in R**

R supports character data types where you have all the alphabets and special characters. It stores character values or strings. Strings in R can contain alphabets, numbers, and symbols. The easiest way to denote that a value is of character type in R data type is to wrap the value inside single or double inverted commas.

```
# Assign a character value to char
char = "ATME Collge of Engineering"
# print the type of char
print(typeof(char))
[1] "character"
```

2. **Demonstrate Arithmetic and Logical Operations with simple examples.**

   (a) **Arithmetic Operators in R**
   Arithmetic operations in R simulate various math operations, like addition, subtraction, multiplication, division, and modulo using the specified operator between operands, which may be either scalar values, complex numbers, or vectors. The R operators are performed element-wise at the corresponding positions of the vectors.

   i. **Addition Operator**

```
a <- 1
b<- 4
print(a+b)
[1] 5
```

   ii. **Subtraction Operator**

```
a <- 6
b <- 8.4
print (a-b)
[1] -2.4
```

   iii. **Multiplication Operator**

```
a <- 6
b <- 8.4
print (a*b)
[1] 50.4
```

   iv. **Division Operator**

```
a<-20
b<-5
print(a/b)
[1] 4
```

   v. **Power Operator**

```
a<-4
b<-3
print(a^b)
[1] 64
```

   vi. **Modulo Operator**

```
a<-10
b<-3
```

```
print(a%%b)
[1] 1
```

(b) **Logical Operators in R**

Logical operations in R simulate element-wise decision operations, based on the specified operator between the operands, which are then evaluated to either a True or False boolean value. Any non-zero integer value is considered as a TRUE value, be it a complex or real number.

i. **Element-wise Logical AND operator**

Returns True if both the operands are True.

```
a <- 1.35
b<- 5+3i
print(a & b)
[1] True
```

Any non zero integer value is considered as a TRUE value, be it complex or real number.

ii. **Element-wise Logical OR operator**

Returns True if either of the operands is True.

```
a <- 1.35
b<- 5+3i
print(a | b)
[1] True
```

iii. **Not Operator**

A unary operator that negates the status of the elements of the operand.

```
a <- TRUE
print(!a)
[1] FALSE
```

iv. **Logical AND operator**

Returns True if both the first elements of the operands are True.

```
a<-20
b<-5
print(a&&b)
[1] TRUE
```

v. **Logical OR operator**

```
a<-4
b<-0
print(a||b)
[1] TRUE
```

3. **Demonstrate generation of sequences and creation of vectors.**

(a) **Sequence generation and creation of vectors**

```
# Sequence generation
seq1 <- 1:10
seq2 <- seq(from = 5, to = 50, by = 5)

# Vector creation
num_vector <- c(10, 20, 30)
char_vector <- c("apple", "banana", "cherry")
log_vector <- c(TRUE, FALSE, FALSE, TRUE)
mixed_vector <- c(1, "Hello", TRUE)

# Print the results
print(seq1)
print(seq2)
print(num_vector)
print(char_vector)
print(log_vector)
print(mixed_vector)
```

4. **Demonstrate Creation of Matrices**

(a) **Sequence generation and creation of vectors**

```
a <- matrix(1:6, 2,3)
print(a)
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

5. **Demonstrate the Creation of Matrices from Vectors using Binding Function**

Creating matrices from vectors in R can be efficiently done using the rbind() and cbind() functions, which stand for row-binding and column-binding, respectively.

(a)

```
# Define vectors
vector1 <- c(1, 2, 3)
vector2 <- c(4, 5, 6)
vector3 <- c(7, 8, 9)
# Create matrix by row binding
matrix_by_row <- rbind(vector1, vector2, vector3)
# Create matrix by column binding
matrix_by_column <- cbind(vector1, vector2, vector3)
# Print results
print("Matrix by Row Binding:")
[1] "Matrix by Row Binding:"
```

```
print(matrix_by_row)
        [,1] [,2] [,3]
vector1    1    2    3
vector2    4    5    6
vector3    7    8    9
print("Matrix by Column Binding:")
[1] "Matrix by Column Binding:"
print(matrix_by_column)
     vector1 vector2 vector3
[1,]       1       4       7
[2,]       2       5       8
[3,]       3       6       9
```

6. **Demonstrate element extraction from vectors, matrices and arrays**

   (a) **Extracting Elements from Vectors**

```
# Define a vector
vector <- c("a", "b", "c", "d", "e")
print(vector)
[1] "a" "b" "c" "d" "e"
# Extract elements
# Extracts the first element
element1 <- vector[1]
# Extracts the second to fourth elements
element2 <- vector[2:4]
# Extracts all elements except the first
element3 <- vector[-1]

print(element1)
[1] "a"
print(element2)
[1] "b" "c" "d"
print(element3)
[1] "b" "c" "d" "e"
```

   (b) **Extracting Elements from Matrices**

```
# Define a matrix
matrix1 <- matrix(1:9, nrow = 3)
print(matrix1)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

# Extract elements
# Extracts element from the
#first row, second column
```

```
element4 <- matrix1[1, 2]
print(element4)
[1] 4
# Extracts the first row
row1 <- matrix1[1, ]
print(row1)
[1] 1 4 7
# Extracts the second column
column2 <- matrix1[, 2]
print(column2)
[1] 4 5 6
# Extracts a sub-matrix
sub_matrix <- matrix1[1:2, 2:3]
print(sub_matrix)
     [,1] [,2]
[1,]    4    7
[2,]    5    8
```

(c) **Extracting Elements from Array**

```
# Define an array
array1 <- array(1:12, dim = c(3, 2, 2))
print(array1)
, , 1

     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6

, , 2

     [,1] [,2]
[1,]    7   10
[2,]    8   11
[3,]    9   12

# Extracts element from the first row,
# second column, first depth layer
element5 <- array1[1, 2, 1]
print(element5)
[1] 4
# Extracts a sub-array
sub_array <- array1[1:2, , 2]
print(sub_array)

     [,1] [,2]
[1,]    7   10
```

```
[2 ,]    8    11
[3 ,]    9    12
```

3 Experiment 2

Assess the Financial Statement of an Organization being supplied with 2 vectors of data: Monthly Revenue and Monthly Expenses for the Financial Year. You can create your own sample data vector for this experiment. Calculate the following financial metrics:

- **Creating two vectors: one for monthly revenue and another for monthly expenses. Creating a sample data set for a financial year(span of 12 months).**

```
# Step 1: Create Sample Data
 monthly_revenue <-
c(12000, 15000, 13000, 14000,
16000, 17000, 15500, 14500,
13500, 16500, 17500, 18000)
 print(monthly_revenue)
[1] 12000 15000 13000 14000 16000
17000 15500 14500 13500 16500 17500 18000
 monthly_expenses <-
c(10000, 11000, 9500, 12000,
13000,14000, 12000, 11000,
10500, 13000, 14000, 15000)
 print(monthly_expenses)
[1] 10000 11000  9500 12000 13000
14000 12000 11000 10500 13000 14000 15000
```

- **Calculate the following financial metrics:**

1. **Profit for each month**

```
# Calculate Monthly Profit
monthly_profit <- monthly_revenue - monthly_expenses
print(monthly_profit)
[1] 2000 4000 3500 2000 3000 3000
       3500 3500 3000 3500 3500 3000
```

2. **Profit after tax for each month (Tax Rate is 30%)**

```
# Define Tax Rate
 tax_rate <- 0.30

# Calculate Tax for Each Month
 monthly_tax <- monthly_profit * tax_rate
print(monthly_tax)
[1]  600 1200 1050  600  900  900 1050 1050  900 1050 1050
900
 # Calculate Profit After Tax for Each Month
 profit_after_tax <- monthly_profit - monthly_tax
```

```
print(profit_after_tax)
[1] 1400 2800 2450 1400 2100 2100 2450
2450 2100 2450 2450 2100
```

3. **Profit margin for each month equals to profit after tax divided by revenue**

```
# Calculate Profit Margin for Each Month
profit_margin_percentage<-(profit_after_tax / monthly_revenue)
# Round Profit Margin to No Decimal Point
profit_margin_rounded <- round(profit_margin_percentage, 0)
# Print Profit Margin for Each Month in Rounded Percentage
print("Monthly Profit Margin (%):")
[1] "Monthly Profit Margin (%):"
print(profit_margin_rounded)
[1] 12 19 19 10 13 12 16 17 16 15 14 12
```

4. **Good Months – where the profit after tax was greater than the mean for the year**

```
# Calculate Mean Profit After Tax for the Year
mean_profit_after_tax <- mean(profit_after_tax)
# Identify Good Months
good_months <- profit_after_tax > mean_profit_after_tax
# Print Good Months
print("Good Months (Profit After Tax Greater
Than Yearly Average):")
[1] "Good Months (Profit After Tax Greater Than
Yearly Average):"
print(good_months)
[1] FALSE TRUE TRUE FALSE FALSE FALSE TRUE TRUE FALSE
TRUE TRUE FALSE
```

5. **Bad Months – where the profit after tax was less than the mean for the year**

```
# Calculate Mean Profit After Tax for the Year
mean_profit_after_tax <- mean(profit_after_tax)
# Identify Bad Months
bad_months <- profit_after_tax < mean_profit_after_tax
# Print Bad Months
print("Bad Months (Profit After Tax Less Than
Yearly Average):")
[1] "Bad Months (Profit After Tax Less Than Yearly Average):"
print(bad_months)
[1] TRUE FALSE FALSE TRUE TRUE TRUE FALSE FALSE
TRUE FALSE FALSE TRUE
```

6. **The best month – where the profit after tax was max for the year**

```
# Find the Best Month (Maximum Profit After Tax)
best_month_index <- which.max(profit_after_tax)
best_month_profit <- profit_after_tax[best_month_index]

# Print Best Month
print(paste("The Best Month is:", best_month_index,
"with a Profit After Tax of:", best_month_profit))
[1] "The Best Month is: 2 with a Profit After Tax of: 2800
```

7. **The worst month – where the profit after tax was min for the year**

```
# Find the Worst Month (Minimum Profit After Tax)
worst_month_index <- which.min(profit_after_tax)
worst_month_profit <- profit_after_tax[worst_month_index]
# Print Worst Month
print(paste("The Worst Month is:",
worst_month_index, "with a Profit
After Tax of:", worst_month_profit))
[1] "The Worst Month is: 1 with a Profit After Tax of: 1400"
```

Note:

- All Results need to be presented as vectors
- Results for Dollar values need to be calculated with $0.01 precision, but need to be presented in Units of $1000 (i.e 1k) with no decimal points
- Results for the profit margin ratio need to be presented in units of % with no decimal point.
- It is okay for tax to be negative for any given month (deferred tax asset)
- Generate CSV file for the data.

```
# Sample Data
monthly_revenue <- c(12000, 15000, 13000,
14000, 16000, 17000, 15500, 14500, 13500,
16500, 17500, 18000)
monthly_expenses <- c(10000, 11000, 9500,
12000, 13000, 14000, 12000, 11000, 10500,
13000, 14000, 15000)
# Create a Data Frame
financial_data <- data.frame(
    Month = 1:12,
    Revenue = monthly_revenue,
    Expenses = monthly_expenses
)

# Write to CSV File
write.csv(financial_data, file = "SampleFinancialData.csv",
row.names = FALSE)
```

4 Experiment 3

Develop a program to create two 3 X 3 matrices A and B and perform the following operations a) Transpose of the matrix b) addition c) subtraction d) multiplication

```
# Create Matrix A
   A <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3, ncol = 3)

 # Create Matrix B
   B <- matrix(c(9, 8, 7, 6, 5, 4, 3, 2, 1), nrow = 3, ncol = 3)

# Print Matrix A
print("Matrix A:")
[1] "Matrix A:"
print(A)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

# Print Matrix B
print("Matrix B:")
[1] "Matrix B:"
  print(B)
     [,1] [,2] [,3]
[1,]    9    6    3
[2,]    8    5    2
[3,]    7    4    1
```

1. **Transpose of the matrix**

```
# Transpose Matrix A
transpose_A <- t(A)

# Transpose Matrix B
transpose_B <- t(B)

# Print Transpose of Matrix A
print("Transpose of Matrix A:")
[1] "Transpose of Matrix A:"
print(transpose_A)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9

# Print Transpose of Matrix B
```

```
print("Transpose of Matrix B:")
[1] "Transpose of Matrix B:"
print(transpose_B)
     [,1] [,2] [,3]
[1,]    9    8    7
[2,]    6    5    4
[3,]    3    2    1
```

2. **Addition of two matrix**

```
# Addition of Matrix A and B
C <- A + B

# Print the Resultant Matrix
print("Addition of Matrix A and B:")
[1] "Addition of Matrix A and B:"
print(C)
     [,1] [,2] [,3]
[1,]   10   10   10
[2,]   10   10   10
[3,]   10   10   10
```

3. **Subtraction of two matrix**

```
# Subtraction of Matrix A and B
C <- A - B

# Print the Resultant Matrix
print("Subtraction of Matrix A and B:")
[1] "Subtraction of Matrix A and B:"
print(C)
     [,1] [,2] [,3]
[1,]   -8   -2    4
[2,]   -6    0    6
[3,]   -4    2    8
```

4. **Multiplication of two matrix**

```
# Matrix Multiplication of A and B
C <- A %*% B

# Print the Resultant Matrix
print("Matrix Multiplication of A and B:")
[1] "Matrix Multiplication of A and B:"
print(C)
     [,1] [,2] [,3]
[1,]   90   54   18
[2,]  114   69   24
[3,]  138   84   30
```

5 Experiment 4

Develop a program to find the factorial of given number using recursive function calls

```
# Define the recursive factorial function
factorial <- function(n)
{
    if (n == 0)
    {
        return(1)
    } else
    {
        return(n * factorial(n - 1))
    }
}

# Example: Calculate the factorial of 5
result <- factorial(5)

# Print the result
print(result)
[1] 120
```

6 Experiment 5

Develop an R Program using functions to find all the prime numbers up to a specified number by the method of Sieve of Eratosthenes

```
# Function to implement Sieve of Eratosthenes
sieve_of_eratosthenes <- function(limit) {
  if (limit >= 2) {
    # Create a vector of numbers from 2 to the specified limit
    numbers <- 2:limit

    # Sieve process
    for (p in numbers) {
      if (p^2 > limit) {
        break
      }
      # Remove multiples of the prime number p
      numbers <- numbers[numbers == p | numbers %% p != 0]
    }

    return(numbers)
  } else {
    return(numeric(0)) # Return empty vector for limit less than 2
  }
}

# Example: Find all prime numbers up to 30
prime_numbers <- sieve_of_eratosthenes(30)

# Print the result
print(prime_numbers)
[1]  2  3  5  7 11 13 17 19 23 29
```

7 Experiment 6

The built-in data set mammals contain data on body weight versus brain weight. Develop R commands to:
a) Find the Pearson and Spearman correlation coefficients. Are they similar?
b) Plot the data using the plot command.
c) Plot the logarithm (log) of each variable and see if that makes a difference.

1. **Load Dataset and Explore Dataset**

```
#Load the Dataset
data("mammals", package = "MASS")

# View the first few rows of the dataset
head(mammals)
```

Output:

	body	brain
Arctic fox	3.385	44.5
Owl monkey	0.480	15.5
Mountain beaver	1.350	8.1
Cow	465.000	423.0
Grey wolf	36.330	119.5
Goat	27.660	115.0

```
# Summary of the dataset
summary(mammals)
```

Output:

	body		brain
Min.	0.005	Min.	0.14
1st Qu.	0.600	1st Qu.	4.25
Median	3.342	Median	17.25
Mean	198.790	Mean	283.13
3rd Qu.	48.202	3rd Qu.	166.00
Max.	6654.000	Max.	5712.00

```
 # Structure of the dataset
str(mammals)
```

```
'data.frame':   62 obs. of  2 variables:
$ body : num  3.38 0.48 1.35 465 36.33 ...
$ brain: num  44.5 15.5 8.1 423 119.5 ...
```

2. **Find the Pearson and Spearman correlation coefficients. Are they similar?**

```
#Calculate Pearson Correlation Coefficient:
```

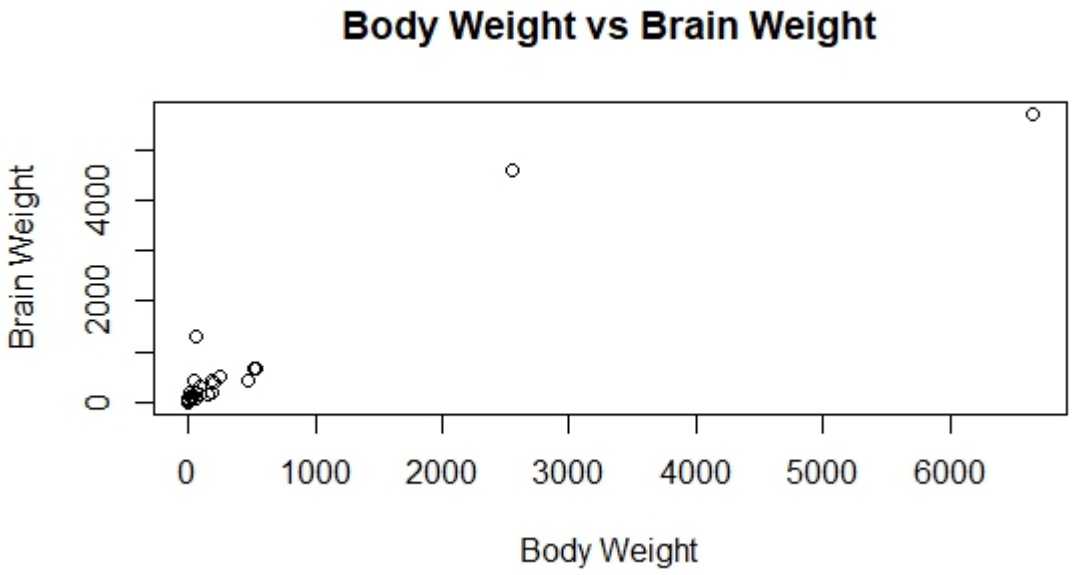
```
cor(mammals$body, mammals$brain, method = "pearson")
Output:
0.9341638

#Calculate Spearman Correlation Coefficient:
cor(mammals$body, mammals$brain, method = "spearman")
Output:
0.9534986

#Compare the Coefficients:
 Check if the Pearson and Spearman coefficients
are similar. They might differ if the relationship is not
linear or if there are outliers.
```

3. **Plot the data using the plot command**

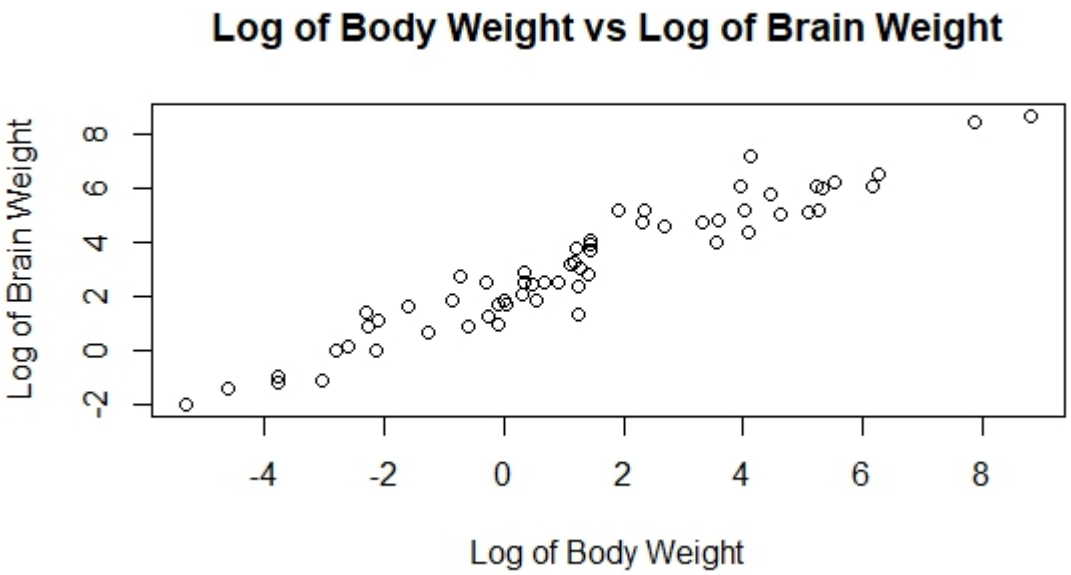
```
plot(mammals$body, mammals$brain, main="Brain vs Body Weight",
xlab="Body Weight", ylab="Brain Weight")
```



4. **Plot the logarithm (log) of each variable and see if that makes a difference.**

```
plot(log(mammals$body), log(mammals$brain), main="Log of Brain
vs Body Weight",
xlab="Log of Body Weight",
ylab="Log of Brain Weight")
```

After plotting the logarithmic values, you can visually inspect the plot to see if the relationship between the variables appears more linear or if the spread of data



points is more uniform. This can affect both the interpretation and statistical analyses.

These steps will help you to calculate and compare the Pearson and Spearman correlation coefficients and to visualize the data using both the original and logarithmic scales. This approach can provide insights into the nature of the relationship between body weight and brain weight in the dataset.

8 Experiment 7

Develop R program to create a DataFrame with following details and do the following operations:

itemCode	itemCategory	itemPrice
1001	Electronics	700
1002	Desktop Supplies	300
1003	Office Supplies	350
1004	USB	400
1005	CD Drive	800

1. **Subset the dataframe and display the details of only those items whose price is greater than or equal to 350.**
2. **Subset the dataframe and display only the items where the category is "Office Supplies" and "Desktop Supplies".**
3. **Create another dataframe called "item-details" with three different fields itemCode, ItemQtyonHand and ItemRecorderLvl and merge the two frame.**

1. **Creating DataFrame**

```
itemCode <- c(1001:1005)
itemCategory <- c("Electronics", "Desktop Supplies",
    "Office Supplies", "USB", "CD Drive")
itemPrice <- c(700, 300, 350, 400, 800)

# Creating the data frame using the above vectors
items_df <- data.frame(itemCode, itemCategory, itemPrice)

# Displaying the created data frame
print("Data Frame with Items Information:")
print(items_df)
Output:
              itemCode      itemCategory itemPrice
    1       1001       Electronics        700
    2       1002 Desktop Supplies        300
    3       1003  Office Supplies        350
    4       1004              USB        400
    5       1005         CD Drive        800

# Summary statistics of itemPrice
print(summary(items_df$itemPrice))
Output:
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
300     350     400     510     700     800
```

2. **Subset the dataframe and display the details of only those items whose price is greater than or equal to 350.**

```
# Filtering items with price greater than 350
high_priced_items <- subset(items_df, itemPrice > 350)
print("Items with Price greater than 400:")
print(high_priced_items)
Output:
  itemCode itemCategory itemPrice
1     1001  Electronics       700
4     1004          USB       400
5     1005     CD Drive       800
```

3. **Subset the dataframe and display only the items where the category is "Office Supplies" and "Desktop Supplies".**

```
# Subset the data frame for items with category as
#"Office Supplies" or "Desktop Supplies"
filtered_items <- subset(items_df,
itemCategory %in% c("Office Supplies", "Desktop Supplies"))

 # Display the subsetted data frame
print("Items with 'Office Supplies' or 'Desktop Supplies'
category:")
print(filtered_items)
Output:
  itemCode itemCategory itemPrice
2     1002 DesktopSupplies       300
3     1003 OfficeSupplies       350
```

4. **Create another dataframe called "item-details" with three different fields itemCode, ItemQtyonHand and ItemRecorderLvl and merge the two frame.**

```
# Creating the 'item-details' data frame
itemCode <- c(1001, 1002, 1003, 1004, 1005)
ItemQtyonHand <- c(20, 15, 30, 10, 25)
ItemReorderLvl <- c(5, 10, 8, 3, 7)

# Creating the data frame using the above vectors
item_details <- data.frame(itemCode, ItemQtyonHand,
ItemReorderLvl)

# Displaying the created data frame
print("Data Frame 'item-details':")
print(item_details)
Output:
  itemCode ItemQtyonHand ItemReorderLvl
1     1001           20              5
```

```
2       1002            15              10
3       1003            30              8
4       1004            10              3
5       1005            25              7

# Merging the two data frames based on 'itemCode'
merged_data <- merge(items_df, item_details, by = "itemCode")

# Displaying the merged data frame
print("Merged Data Frame:")
print(merged_data)
```

Output:

itemCode	itemCategory	itemPrice	ItemQtyonHand	ItemOrderLvl
1001	Electronics	700	20	5
1002	Desktop Supplies	300	15	10
1003	Office Supplies	350	30	8
1004	USB	400	10	3
1005	CD Drive	800	25	7

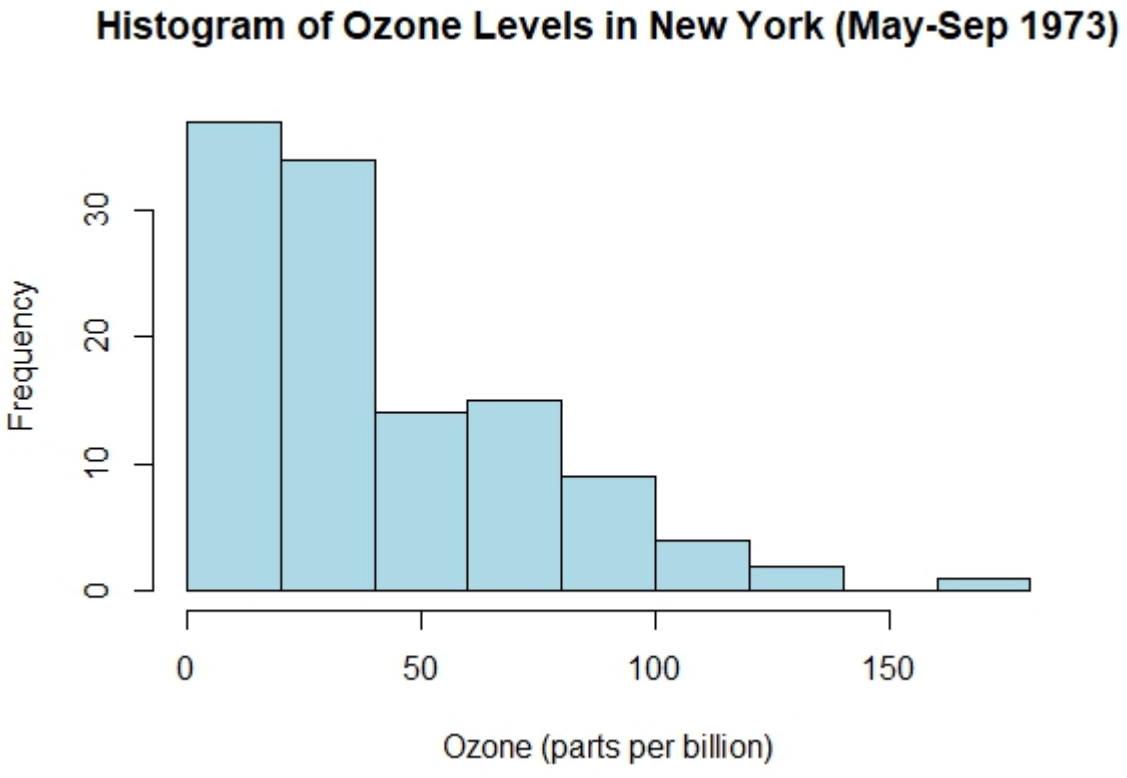
9 Experiment 8

Let us use the built-in dataset air quality which has Daily air quality measurements in New York, May to September 1973. Develop R program to generate histogram by using appropriate arguments for the following statements.
a) Assigning names, using the air quality data set.
b) Change colors of the Histogram
c) Remove Axis and Add labels to Histogram
d) Change Axis limits of a Histogram
e) Add Density curve to the histogram

1. Assigning names, using the air quality data set.

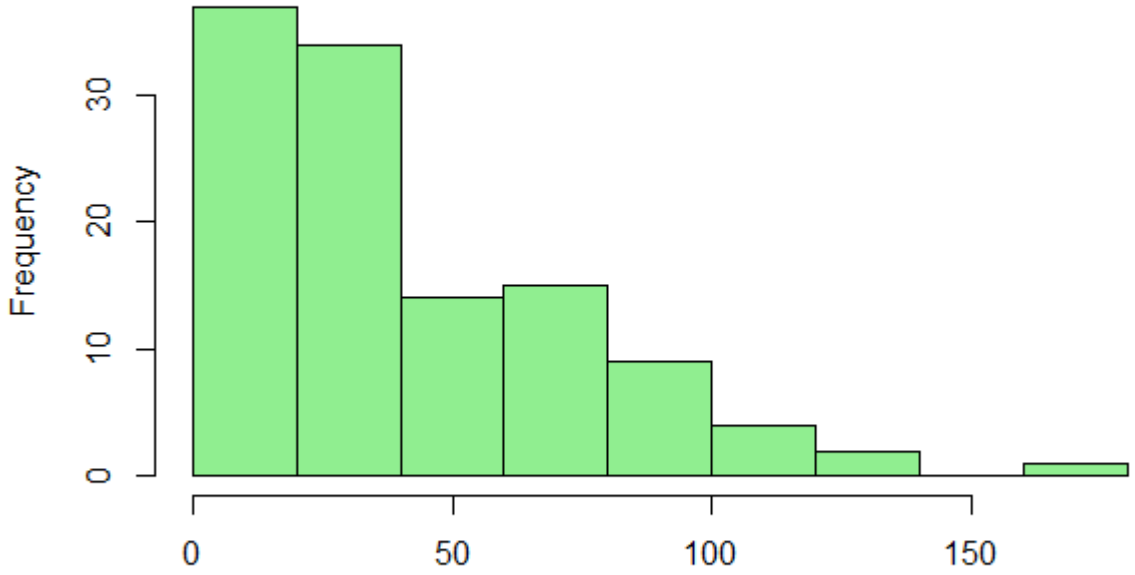
```
# Load the airquality dataset
data(airquality)

# Histogram for Ozone Levels
hist(airquality$Ozone,
main = "Histogram of Ozone Levels",xlab = "Ozone (ppb)",
col = "lightblue", border = "black")
```



2. Change colors of the Histogram.

```
  # Histogram for Ozone Levels
hist(airquality$Ozone, col = "lightgreen", main = "", xlab = ""
, ylab = "Frequency")
```



3. Change Axis limits of a Histogram

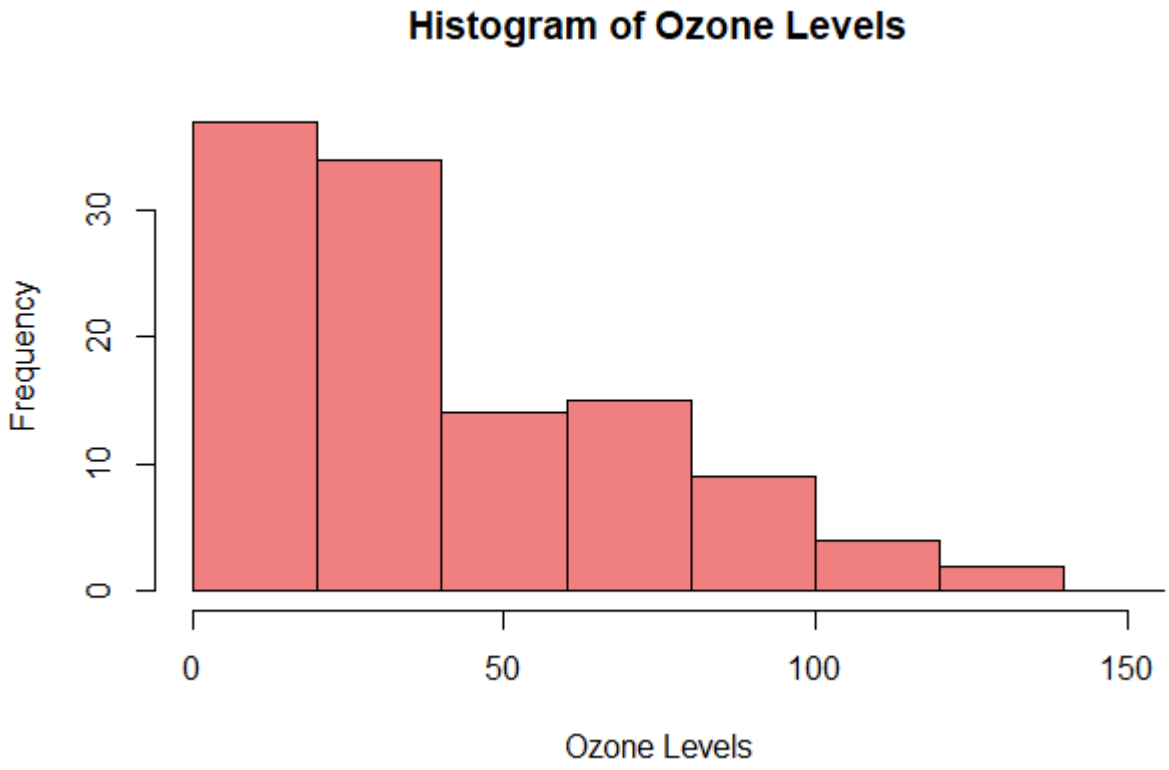
```
hist(airquality$Ozone, col = "lightcoral",
main = "Histogram of Ozone Levels",
xlab = "Ozone Levels", xlim = c(0, 150),ylim = c(0, 40))
```

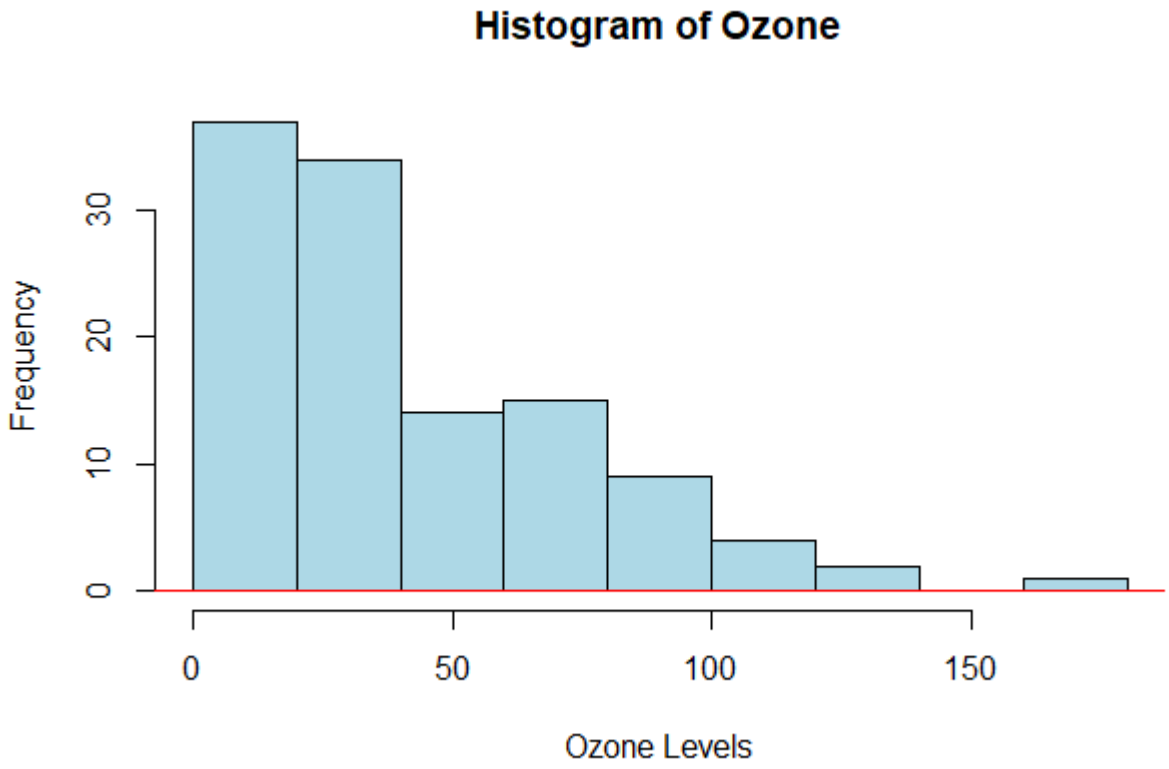
4. Add Density curve to the histogram

```
# Remove missing values in 'Ozone' column
cleaned_data <- na.omit(airquality$Ozone)

# Create a histogram of 'Ozone' column
hist(cleaned_data, col = "lightblue", main = "Histogram of Ozone",
xlab = "Ozone Levels", ylab = "Frequency")

# Add a density curve to the histogram
lines(density(cleaned_data), col = "red")
```


Histogram of Ozone Levels
Frequency
0
10
20
30
0
50
100
150
Ozone Levels


Histogram of Ozone
Frequency
0
10
20
30
0
50
100
150
Ozone Levels

10 Experiment 9

Design a data frame in R for storing about 20 employee details. Create a csv file named "input.csv" that defines all the required information about the employee such as id, name, salary, startdate, dept. Import into R and do the following analysis.
a) Find the number of rows and columns.
b) Find the maximum salary.
c) Retrieve the details of the employee with maximum salary.
d) Retrieve all the employees working in the IT department.
e) Retrieve the employees in IT Department whose salary is greater than 20000 and write these details into another file output.csv

1. **Find the number of rows and columns**

```
# Create a data frame for employee details
employee_data <- data.frame(
  id = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
  16, 17, 18, 19, 20),
  name = c("John", "Jane", "Bob", "Alice", "Charlie",
  "David", "Emily", "Frank", "Grace", "Henry", "Ivy",
  "Jack", "Karen", "Liam", "Mia", "Noah",
  "Olivia", "Peter", "Quinn", "Rachel"),
  salary = c(25000, 30000, 35000, 40000, 45000,
  50000, 55000, 60000, 65000, 70000, 75000,
  80000, 85000, 90000, 95000, 100000, 105000,
  110000, 115000, 120000),
  start_date = as.Date(c("2021-01-01", "2021-02-01",
  "2021-03-01", "2021-04-01", "2021-05-01",
  "2021-06-01", "2021-07-01", "2021-08-01",
  "2021-09-01", "2021-10-01", "2021-11-01",
  "2021-12-01", "2022-01-01", "2022-02-01",
  "2022-03-01", "2022-04-01", "2022-05-01",
  "2022-06-01", "2022-07-01", "2022-08-01")),
  dept = c("IT", "HR", "IT", "Finance", "IT",
  "HR", "IT", "Finance", "IT", "HR", "IT",
  "Finance", "IT", "HR", "IT", "Finance",
  "IT", "HR", "IT", "Finance")
)

# Save the data frame to a CSV file
write.csv(employee_data, file = "input.csv", row.names = FALSE)

# Read the CSV file into R
employee_data <- read.csv("input.csv")

# a) Find the number of rows and columns
num_rows <- nrow(employee_data)
```

```
num_cols <- ncol(employee_data)

cat(" Number of rows:", num_rows, "\n")
cat("    Number of columns:", num_cols, "\n")
```

2. **Find the maximum salary**

```
 max_salary <- max(employee_data$salary)
cat(" Maximum salary:", max_salary, "\n")
```

3. **Retrieve the details of the employee with the maximum salary**

```
employee_max_salary<-employee_data[employee_data$salary == max_salary ,]
cat("Employee with maximum salary:\n")
print(employee_max_salary)
```

4. **Retrieve all the employees working in the IT department**

```
it_department_employees <- subset(employee_data, dept == "IT")
cat("Employees working in the IT Department:\n")
print(it_department_employees)
```

5. **Retrieve the employees in IT Department whose salary is greater than 20000**

```
employees_IT_high_salary <- subset(it_department_employees,
salary > 20000)
cat("Employees in the IT Department with salary > 20000:\n")
print(employees_IT_high_salary)
# Write these employees to a new CSV file
write.csv(employees_IT_high_salary, "output.csv", row.names = FALSE)
```

11 Experiment 10

Using the built in dataset which is a popular dataset consisting of design and fuel consumption pattern of 32 different automobiles. The data was extracted from 1974 Motor trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for automobiles(1973-74 models). format a dataframe on 32 observations on 11 variables.[1] mpg Miles/[US] gallon, [2]cyl Number of cylinders[3] disp displacement(cu.in), [4]hp Gross horsepower, [5] dart rear axle ratio, [6]wt Weight lb/1000, [7] qsec 1/4 mile/time, [8] vs V/S, [9] am Transmission (0=automatic, 1=manual), [10] gear number of forward gears,[11] carb number of carburetors
a) What is the total number of observations and variables in the dataset?
b) Find the car with largest hp and the least hp using suitable functions?
c) Plot histogram/density for each variable and determine whether continuous variables are normally distributed or not. If not, what is their skewness?
d) Average difference and standard deviation between hp for 3 and 4 cylinders
e) Which pair of variable has the highest Pearson correlation?

1. **What is the total number of observations and variables in the dataset?**

```
# Load the mtcars dataset
data(mtcars)

# Create a dataframe with 32 observations and 11 variables
my_dataframe <- mtcars[, c("mpg", "cyl", "disp", "hp", "drat",
"wt", "qsec", "vs", "am", "gear", "carb")]

# Print the dataframe
print(my_dataframe)

# Get the number of observations and variables
num_observations <- nrow(mtcars)
num_variables <- ncol(mtcars)

# Print the results
cat("Total number of observations:", num_observations, "\n")
cat("Total number of variables:", num_variables, "\n")
```

2. **Find the car with largest hp and the least hp using suitable functions?**

```
 # Find the car with the largest horsepower
car_max_hp <- mtcars[which.max(mtcars$hp), ]

# Find the car with the least horsepower
car_min_hp <- mtcars[which.min(mtcars$hp), ]

# Print the results
```

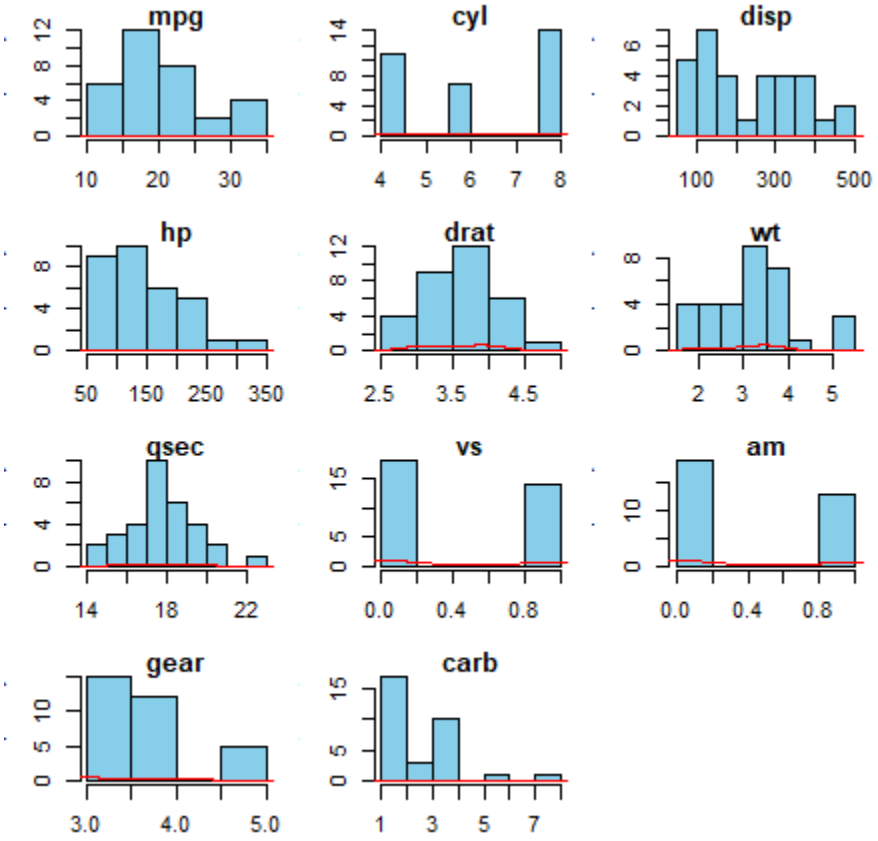
```
print("Car with the largest horsepower:")
print(car_max_hp)

print("Car with the least horsepower:")
print(car_min_hp)
```

3. **Plot histogram/density for each variable and determine whether continuous variables are normally distributed or not. If not, what is their skewness?**

```
par(mfrow = c(4, 3), mar = c(3, 3, 1, 1))  # Adjusting margin size
for (i in 1:ncol(mtcars)) {
  hist(mtcars[, i], main = names(mtcars)[i], xlab = "", col = "skyblue"
  lines(density(mtcars[, i]), col = "red")  # Adding density curve
}

# Calculate skewness
library(e1071)
skew <- sapply(mtcars, skewness)
cat("Skewness of variables:\n")
print(skew)
```



4. **Average difference and standard deviation between hp for 3 and 4 cylinders**

```
# Subset the dataset for 3 and 4 cylinders
hp_3_cyl <- mtcars$hp[mtcars$cyl == 3]
hp_4_cyl <- mtcars$hp[mtcars$cyl == 4]

# Calculate the average difference
avg_difference <- mean(hp_3_cyl) - mean(hp_4_cyl)

# Calculate the standard deviation of the difference
std_dev_difference <- sd(hp_3_cyl - hp_4_cyl)

# Print the results
print(paste("Average difference in hp:", round(avg_difference, 2)))
print(paste("Standard deviation of the difference:",
round(std_dev_difference, 2)))
```

5. **Pair of variables with the highest Pearson correlation**

```
cor_matrix <- cor(mtcars)

# Find the pair with the highest correlation
max_corr <- max(cor_matrix[upper.tri(cor_matrix, diag = FALSE)])

# Find the indices of the pair with the highest correlation
indices <- which(cor_matrix == max_corr, arr.ind = TRUE)

# Get the variable names for the pair
variable1 <- rownames(cor_matrix)[indices[1, 1]]
variable2 <- colnames(cor_matrix)[indices[1, 2]]

# Print the results
print(paste("Pair with the highest Pearson correlation:",
variable1, "and", variable2))
print(paste("Highest Pearson correlation:", round(max_corr, 2)))
```

12 Experiment 11

Demonstrate the progression of salary with years of experience using a suitable data set (You can create your own dataset). Plot the graph visualizing the best fit line on the plot of the given data points. Plot a curve of Actual Values vs. Predicted values to show their correlation and performance of the model. Interpret the meaning of the slope and y-intercept of the line with respect to the given data. Implement using lm function. Save the graphs and coefficients in files. Attach the predicted values of salaries as a new column to the original data set and save the data as a new CSV file.

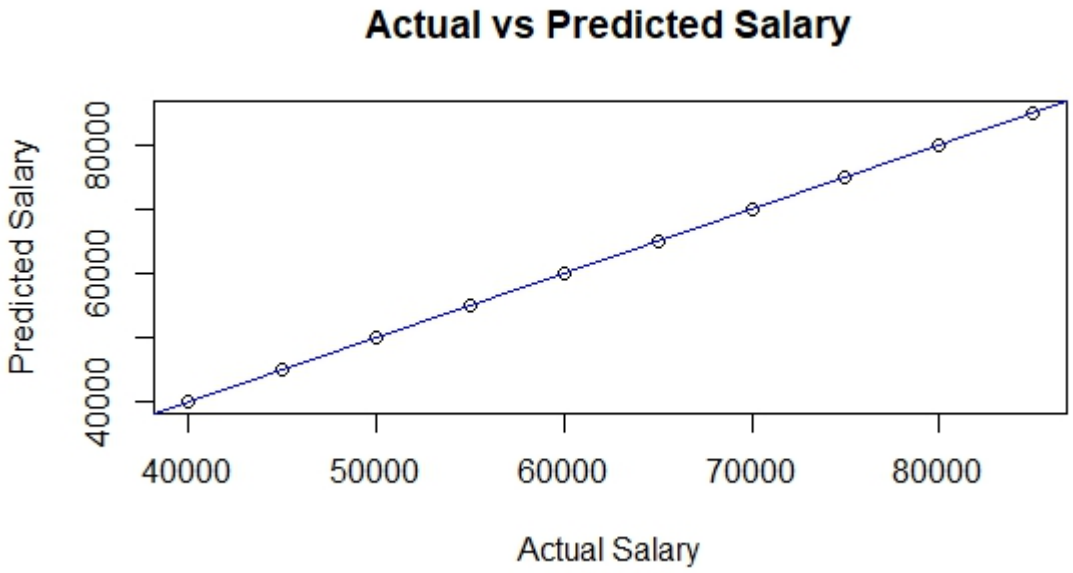
```
Years_Exp <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
Salary <- c(40000, 45000, 50000, 55000, 60000,
65000, 70000, 75000, 80000, 85000)

# Create a data frame
salary_data <- data.frame(Years_Exp, Salary)
# Fit the linear regression model
model <- lm(Salary ~ Years_Exp, data = salary_data)
# Plot the scatter plot
plot(salary_data$Years_Exp, salary_data$Salary,
xlab = "Years of Experience", ylab = "Salary",
main = " Years of Experience vs. Salary")

# Add the best fit line
abline(model, col = "blue", lwd = 2)
# Calculate predicted salaries
predicted_salaries <- predict(model, newdata = salary_data)
# Add predicted salaries to the data frame
salary_data$Predicted_Salary <- predicted_salaries

# Plot Actual vs. Predicted Salaries
plot(salary_data$Salary, salary_data$Predicted_Salary,
xlab = "Actual Salary", ylab = "Predicted Salary",
main = "Actual vs. Predicted Salaries")
# Add a 45-degree reference line
abline(a = 0, b = 1, col = "red", lwd = 2)
# Save the data as a CSV file
write.csv(salary_data, file = "salary_data_with_predictions.csv",
row.names = FALSE)
```


Salary vs Experience
Salary
40000
60000
80000
2
4
6
8
10
Years of Experience


Actual vs Predicted Salary
Predicted Salary
40000
60000
80000
40000
50000
60000
70000
80000
Actual Salary