TCET	R Programming	SY-AIDS-34
Name: ujwal sahu	Batch: B	
Subject: R Programming	Experiment No: 2	SY-BVOC SEM IV

Title: Implement data structures in R on datasets.

Tools: R studio

Theory: A data structure is a particular way of organizing data in a computer so that it can be used effectively. The idea is to reduce the space and time complexities of different tasks. Data structures in R programming are tools for holding multiple values.

R's base data structures are often organized by their dimensionality (1D, 2D, or nD) and whether they're homogeneous (all elements must be of the identical type) or heterogeneous (the elements are often of various types). This gives rise to the six data types which are most frequently utilized in data analysis.

ration with other programming languages like Python, C, and Java.

The most essential data structures used in R include:

- Vectors
- Lists
- Dataframes
- Matrices
- Arrays
- Factors
- Tibbles

1) From the mtcars dataset, create a vector from the mpg column. Replace values greater than the 75th percentile with "High" and less than the 25th percentile with "Low"leaving the rest as "Medium".

   Code:

```
data(mtcars)
mpg_vector <- mtcars$mpg
lower_percentile <- quantile(mpg_vector, 0.25)
upper_percentile <- quantile(mpg_vector, 0.75)
mpg_category <- ifelse(mpg_vector > upper_percentile, "High",
                       ifelse(mpg_vector < lower_percentile, "Low", "Medium"))
print(mpg_category)
```

Output:

```
data(mtcars)
mpg_vector <- mtcars$mpg
lower_percentile <- quantile(mpg_vector, 0.25)
upper_percentile <- quantile(mpg_vector, 0.75)
mpg_category <- ifelse(mpg_vector > upper_percentile, "High",
                       ifelse(mpg_vector < lower_percentile, "Low", "Mediu
print(mpg_category)
```

```
> data(mtcars)
> mpg_vector <- mtcars$mpg
> lower_percentile <- quantile(mpg_vector, 0.25)
> upper_percentile <- quantile(mpg_vector, 0.75)
> mpg_category <- ifelse(mpg_vector > upper_percentile, "High",
+                        ifelse(mpg_vector < lower_percentile, "Low", "Medium"))
> print(mpg_category)
 [1] "Medium" "Medium" "Medium" "Medium" "Medium" "Medium" "Low"    "High"
 [9] "Medium" "Medium" "Medium" "Medium" "Medium" "Low"    "Low"    "Low"
[17] "Low"    "High"   "High"   "High"   "Medium" "Medium" "Low"    "Low"
[25] "Medium" "High"   "High"   "High"   "Medium" "Medium" "Low"    "Medium"
>
```

2) Create two vectors from the iris dataset (Sepal.Length and Sepal.Width). Calculate the element-wise difference and product. Find the sum of only the positive differences.

Code:

```
data(iris)
sepal_length <- iris$Sepal.Length
sepal_width <- iris$Sepal.Width
difference <- sepal_length - sepal_width
product <- sepal_length * sepal_width
positive_difference_sum <- sum(difference[difference > 0])
print(paste("Sum of positive differences: ", positive_difference_sum))
print("Product of Sepal.Length and Sepal.Width: ")
print(product)
```

Output:

```
> data(iris)
> sepal_length <- iris$Sepal.Length
> sepal_width <- iris$Sepal.Width
> difference <- sepal_length - sepal_width
> product <- sepal_length * sepal_width
> positive_difference_sum <- sum(difference[difference > 0])
> print(paste("Sum of positive differences: ", positive_difference_sum))
[1] "Sum of positive differences:  417.9"
> print("Product of Sepal.Length and Sepal.Width: ")
[1] "Product of Sepal.Length and Sepal.Width: "
> print(product)
  [1] 17.85 14.70 15.04 14.26 18.00 21.06 15.64 17.00 12.76 15.19 19.98 16.32
 [13] 14.40 12.90 23.20 25.08 21.06 17.85 21.66 19.38 18.36 18.87 16.56 16.83
 [25] 16.32 15.00 17.00 18.20 17.68 15.04 14.88 18.36 21.32 23.10 15.19 16.00
```

```
 [37] 19.25 17.64 13.20 17.34 17.50 10.35 14.08 17.50 19.38 14.40 19.38 14.72
 [49] 19.61 16.50 22.40 20.48 21.39 12.65 18.20 15.96 20.79 11.76 19.14 14.04
 [61] 10.00 17.70 13.20 17.69 16.24 20.77 16.80 15.66 13.64 14.00 18.88 17.08
 [73] 15.75 17.08 18.56 19.80 19.04 20.10 17.40 14.82 13.20 13.20 15.66 16.20
 [85] 16.20 20.40 20.77 14.49 16.80 13.75 14.30 18.30 15.08 11.50 15.12 17.10
 [97] 16.53 17.98 12.75 15.96 20.79 15.66 21.30 18.27 19.50 22.80 12.25 21.17
[109] 16.75 25.92 20.80 17.28 20.40 14.25 16.24 20.48 19.50 29.26 20.02 13.20
[121] 22.08 15.68 21.56 17.01 22.11 23.04 17.36 18.30 17.92 21.60 20.72 30.02
[133] 17.92 17.64 15.86 23.10 21.42 19.84 18.00 21.39 20.77 21.39 15.66 21.76
[145] 22.11 20.10 15.75 19.50 21.08 17.70
```

3) From the mtcars dataset, extract rows where mpg is greater than the median and hp is below the median. Find the correlation between mpg and hp for this subset.

Code:

```
data(mtcars)
median_mpg <- median(mtcars$mpg)
median_hp <- median(mtcars$hp)
subset_data <- mtcars[mtcars$mpg > median_mpg & mtcars$hp < median_hp, ]
correlation <- cor(subset_data$mpg, subset_data$hp)
print(correlation)
```

Output:

```
> data(mtcars)
> median_mpg <- median(mtcars$mpg)
> median_hp <- median(mtcars$hp)
> subset_data <- mtcars[mtcars$mpg > median_mpg & mtcars$hp < median_hp, ]
> correlation <- cor(subset_data$mpg, subset_data$hp)
> print(correlation)
[1] -0.6537747
```

4) Create a list from the mtcars dataset containing mpg, hp, and cyl. Update the list to include the row names as a new element. Then filter out the cars where mpg is less than 20.

Code:

```
data(mtcars)
car_list <- list(
  mpg = mtcars$mpg,
  hp = mtcars$hp,
  cyl = mtcars$cyl
)
car_list$row_names <- rownames(mtcars)
car_df <- data.frame(car_list)
filtered_cars <- car_df[car_df$mpg >= 20, ]
print(filtered_cars)
```

Output:

```
> data(mtcars)
> car_list <- list(
+   mpg = mtcars$mpg,
+   hp = mtcars$hp,
+   cyl = mtcars$cyl
+ )
> car_list$row_names <- rownames(mtcars)
> car_df <- data.frame(car_list)
> filtered_cars <- car_df[car_df$mpg >= 20, ]
> print(filtered_cars)
    mpg  hp cyl      row_names
1  21.0 110   6      Mazda RX4
2  21.0 110   6  Mazda RX4 Wag
3  22.8  93   4     Datsun 710
4  21.4 110   6 Hornet 4 Drive
8  24.4  62   4      Merc 240D
9  22.8  95   4       Merc 230
18 32.4  66   4       Fiat 128
19 30.4  52   4    Honda Civic
20 33.9  65   4 Toyota Corolla
21 21.5  97   4  Toyota Corona
26 27.3  66   4      Fiat X1-9
27 26.0  91   4  Porsche 914-2
28 30.4 113   4   Lotus Europa
32 21.4 109   4     Volvo 142E
```

5) From the iris dataset, create a list containing Sepal.Length, Sepal.Width, and Species.Merge this list with a new list containing a summary of Sepal.Length using the summary() function.

Code:

```
data(iris)
iris_list <- list(
  Sepal.Length = iris$Sepal.Length,
  Sepal.Width = iris$Sepal.Width,
  Species = iris$Species
)
sepal_length_summary <- summary(iris$Sepal.Length)
merged_list <- c(iris_list, Sepal.Length.Summary = sepal_length_summary)
print(merged_list)
```

Output:

```
> data(iris)
> iris_list <- list(
+   Sepal.Length = iris$Sepal.Length,
+   Sepal.Width = iris$Sepal.Width,
+   Species = iris$Species
+ )
> sepal_length_summary <- summary(iris$Sepal.Length)
> merged_list <- c(iris_list, Sepal.Length.Summary = sepal_length_summary)
> print(merged_list)
$Sepal.Length
  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4 5.1
 [19] 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5 4.9 5.0
 [37] 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0 6.4 6.9 5.5
 [55] 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8 6.2 5.6 5.9 6.1
```

6) Create a matrix from the mtcars dataset containing the first 10 rows of mpg, hp, and cyl. Add a new row representing a car with mpg = 18, hp = 150, and cyl = 6, and a new column showing the sum of all rows.

Code:

```
data(mtcars)
car_matrix <- as.matrix(mtcars[1:10, c("mpg", "hp", "cyl")])
new_car <- c(18, 150, 6)
car_matrix <- rbind(car_matrix, new_car)
rownames(car_matrix)[nrow(car_matrix)] <- "New_Car"
car_matrix <- cbind(car_matrix, Sum = rowSums(car_matrix))
print(car_matrix)
```

Output:

```
> data(mtcars)
> car_matrix <- as.matrix(mtcars[1:10, c("mpg", "hp", "cyl")])
> new_car <- c(18, 150, 6)
> car_matrix <- rbind(car_matrix, new_car)
> rownames(car_matrix)[nrow(car_matrix)] <- "New_Car"
> car_matrix <- cbind(car_matrix, Sum = rowSums(car_matrix))
> print(car_matrix)
                   mpg  hp cyl   Sum
Mazda RX4         21.0 110   6 137.0
Mazda RX4 Wag     21.0 110   6 137.0
Datsun 710        22.8  93   4 119.8
Hornet 4 Drive    21.4 110   6 137.4
Hornet Sportabout 18.7 175   8 201.7
Valiant           18.1 105   6 129.1
Duster 360        14.3 245   8 267.3
```

7) From a matrix created from mtcars (mpg and hp), write a function that accepts row and column indices and returns the value at that position. If the indices are out of bounds, return "Invalid".

Code:

```
data(mtcars)
car_matrix <- as.matrix(mtcars[, c("mpg", "hp")])
get_value <- function(matrix, row_index, col_index) {
  if (row_index < 1 || row_index > nrow(matrix) || col_index < 1 || col_index >
ncol(matrix)) {
    return("Invalid")
  } else {
    return(matrix[row_index, col_index])
  }
}
```

```
value1 <- get_value(car_matrix, 5, 1)
print(value1)
value2 <- get_value(car_matrix, 10, 2)
print(value2)
invalid_value <- get_value(car_matrix, 15, 1) # Out of bounds
print(invalid_value)
```

Output:

```
> data(mtcars)
> car_matrix <- as.matrix(mtcars[, c("mpg", "hp")])
> get_value <- function(matrix, row_index, col_index) {
+   if (row_index < 1 || row_index > nrow(matrix) || col_index < 1 || col_index >
ncol(matrix)) {
+     return("Invalid")
+   } else {
+     return(matrix[row_index, col_index])
+   }
+ }
> value1 <- get_value(car_matrix, 5, 1)
> print(value1)
[1] 18.7
> value2 <- get_value(car_matrix, 10, 2)
> print(value2)
[1] 123
> invalid_value <- get_value(car_matrix, 15, 1)  # Out of bounds
> print(invalid_value)
[1] 10.4
```

8) Create a 3D array from the iris dataset using Sepal.Length, Sepal.Width, and Petal.Length. Find the row-wise mean for each matrix level and store the result in a new matrix.

Code:

```
data(iris)
iris_array <- array(c(iris$Sepal.Length[1:10],
                      iris$Sepal.Width[1:10],
                    iris$Petal.Length[1:10]),
                    dim = c(10, 3, 1))
dimnames(iris_array) <- list(paste("Row", 1:10),
                             c("Sepal.Length", "Sepal.Width", "Petal.Length"),
                             "Level 1")
print("3D Array:")
print(iris_array)
row_means <- apply(iris_array, c(1, 3), mean)
print("Row-wise Means:")
print(row_means)
```

Output:

```
> data(iris)
> iris_array <- array(c(iris$Sepal.Length[1:10],
+                       iris$Sepal.Width[1:10],
+                       iris$Petal.Length[1:10]),
+                     dim = c(10, 3, 1))
> dimnames(iris_array) <- list(paste("Row", 1:10),
+                              c("Sepal.Length", "Sepal.Width", "Petal.Length"),
+                              "Level 1")
> print("3D Array:")
[1] "3D Array:"
> print(iris_array)
, , Level 1

       Sepal.Length Sepal.Width Petal.Length
Row 1           5.1         3.5          1.4
Row 2           4.9         3.0          1.4
Row 3           4.7         3.2          1.3
Row 4           4.6         3.1          1.5
Row 5           5.0         3.6          1.4
Row 6           5.4         3.9          1.7
Row 7           4.6         3.4          1.4
Row 8           5.0         3.4          1.5
Row 9           4.4         2.9          1.4
Row 10          4.9         3.1          1.5

> row_means <- apply(iris_array, c(1, 3), mean)
> print("Row-wise Means:")
[1] "Row-wise Means:"
> print(row_means)
        Level 1
Row 1  3.333333
Row 2  3.100000
Row 3  3.066667
Row 4  3.066667
Row 5  3.333333
Row 6  3.666667
```

9) From an array created using mtcars (mpg, hp, cyl), create a function that calculates the mean and median for a specific matrix level. Use apply() for the calculation.

Code:

```
data(mtcars)
mtcars_array <- array(c(mtcars$mpg, mtcars$hp, mtcars$cyl),
                      dim = c(nrow(mtcars), 3, 1))
dimnames(mtcars_array) <- list(rownames(mtcars),
                               c("mpg", "hp", "cyl"),
                               "Level 1")
print("3D Array:")
print(mtcars_array)
calculate_stats <- function(array, level) {
  if (level < 1 || level > dim(array)[3]) {
  return("Invalid level")
```

```
  }
  means <- apply(array[, , level], 1, mean)
  medians <- apply(array[, , level], 1, median)
  result <- data.frame(Mean = means, Median = medians)
  return(result)
}
stats_level_1 <- calculate_stats(mtcars_array, 1)
print("Mean and Median for Level 1:")
print(stats_level_1)
```

Output:

```
> data(mtcars)
> mtcars_array <- array(c(mtcars$mpg, mtcars$hp, mtcars$cyl),
+                       dim = c(nrow(mtcars), 3, 1))
> dimnames(mtcars_array) <- list(rownames(mtcars),
+                                c("mpg", "hp", "cyl"),
+                                "Level 1")
> print("3D Array:")
[1] "3D Array:"
> print(mtcars_array)
, , Level 1

                   mpg  hp cyl
Mazda RX4         21.0 110   6
Mazda RX4 Wag     21.0 110   6
Datsun 710        22.8  93   4
Hornet 4 Drive    21.4 110   6
Hornet Sportabout 18.7 175   8
Valiant           18.1 105   6
Duster 360        14.3 245   8
Merc 240D         24.4  62   4
Merc 230          22.8  95   4
Merc 280          19.2 123   6
Merc 280C         17.8 123   6
Merc 450SE        16.4 180   8
Merc 450SL        17.3 180   8
Merc 450SLC       15.2 180   8
Cadillac Fleetwood  10.4 205   8
Lincoln Continental 10.4 215   8
```

```
> calculate_stats <- function(array, level) {
+   if (level < 1 || level > dim(array)[3]) {
+     return("Invalid level")
+   }
+   means <- apply(array[, , level], 1, mean)
+   medians <- apply(array[, , level], 1, median)
+   result <- data.frame(Mean = means, Median = medians)
+   return(result)
+ }
> stats_level_1 <- calculate_stats(mtcars_array, 1)
> print("Mean and Median for Level 1:")
[1] "Mean and Median for Level 1:"
> print(stats_level_1)
                     Mean Median
Mazda RX4        45.66667   21.0
Mazda RX4 Wag    45.66667   21.0
Datsun 710       39.93333   22.8
Hornet 4 Drive   45.80000   21.4
Hornet Sportabout 67.23333  18.7
Valiant          43.03333   18.1
Duster 360       89.10000   14.3
Merc 240D        30.13333   24.4
Merc 230         40.60000   22.8
Merc 280         49.40000   19.2
Merc 280C        48.93333   17.8
Merc 450SE       68.13333   16.4
Merc 450SL       68.43333   17.3
Merc 450SLC      67.73333   15.2
```

10) Create a data frame from the iris dataset containing Sepal.Length, Sepal.Width, and Species. Create a function that takes a species name as input and returns the average Sepal.Length for that species.

Code:

```
data(iris)
iris_df <- iris[, c("Sepal.Length", "Sepal.Width", "Species")]
print("Iris Data Frame:")
print(head(iris_df))
average_sepal_length <- function(species_name) {
 filtered_data <- iris_df[iris_df$Species == species_name, ]
 if (nrow(filtered_data) == 0) {
  return("Species not found")
 }
  avg_length <- mean(filtered_data$Sepal.Length)
 return(avg_length)
}
species_to_check <- "setosa"
avg_length_setosa <- average_sepal_length(species_to_check)
print(paste("Average Sepal.Length for", species_to_check, ":",
avg_length_setosa))
species_not_found <- "unknown_species"
avg_length_unknown <- average_
```

Output:

```
> data(iris)
> iris_df <- iris[, c("Sepal.Length", "Sepal.Width", "Species")]
> print("Iris Data Frame:")
[1] "Iris Data Frame:"
> print(head(iris_df))
  Sepal.Length Sepal.Width Species
1          5.1         3.5  setosa
2          4.9         3.0  setosa
3          4.7         3.2  setosa
4          4.6         3.1  setosa
5          5.0         3.6  setosa
6          5.4         3.9  setosa
> average_sepal_length <- function(species_name) {
+   filtered_data <- iris_df[iris_df$Species == species_name, ]
+   if (nrow(filtered_data) == 0) {
+     return("Species not found")
+   }
+     avg_length <- mean(filtered_data$Sepal.Length)
+   return(avg_length)
+ }
> species_to_check <- "setosa"
> avg_length_setosa <- average_sepal_length(species_to_check)
> print(paste("Average Sepal.Length for", species_to_check, ":", avg_length_setos
a))
[1] "Average Sepal.Length for setosa : 5.006"
> species_not_found <- "unknown_species"
> avg_length_unknown <- average_sepal_length(species_not_found)
> print(avg_length_unknown)
[1] "Species not found"
```

11) From the mtcars dataset, create a new column power_to_weight as the ratio of hp to wt.Then find the car with the highest and lowest power_to_weight ratio.

Code:

```
data(mtcars)
mtcars$power_to_weight <- mtcars$hp / mtcars$wt
print("Updated mtcars dataset with power_to_weight:")
print(head(mtcars))
highest_power_to_weight <- mtcars[which.max(mtcars$power_to_weight), ]
lowest_power_to_weight <- mtcars[which.min(mtcars$power_to_weight), ]
print("Car with the highest power_to_weight ratio:")
print(highest_power_to_weight)
print("Car with the lowest power_to_weight ratio:")
print(lowest_power_to_weight)
```

Output:

```
> data(mtcars)
> mtcars$power_to_weight <- mtcars$hp / mtcars$wt
> print("Updated mtcars dataset with power_to_weight:")
[1] "Updated mtcars dataset with power_to_weight:"
> print(head(mtcars))
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
                  power_to_weight
Mazda RX4                41.98473
Mazda RX4 Wag            38.26087
Datsun 710               40.08621
Hornet 4 Drive           34.21462
Hornet Sportabout        50.87209
Valiant                  30.34682
> highest_power_to_weight <- mtcars[which.max(mtcars$power_to_weight), ]
> lowest_power_to_weight <- mtcars[which.min(mtcars$power_to_weight), ]
> print("Car with the highest power_to_weight ratio:")
[1] "Car with the highest power_to_weight ratio:"
> print(highest_power_to_weight)
              mpg cyl disp  hp drat   wt qsec vs am gear carb power_to_weight
Maserati Bora  15   8  301 335 3.54 3.57 14.6  0  1    5    8        93.83754
> print("Car with the lowest power_to_weight ratio:")
[1] "Car with the lowest power_to_weight ratio:"
> print(lowest_power_to_weight)
           mpg cyl  disp hp drat   wt qsec vs am gear carb power_to_weight
Merc 240D 24.4   4 146.7 62 3.69 3.19   20  1  0    4    2        19.43574
```

12) From the iris dataset, create a new data frame where the Sepal.Length is greater than 5.5. Then calculate the correlation matrix for the numeric columns.

Code:

```
data(iris)
iris_filtered <- iris[iris$Sepal.Length > 5.5, ]
print("Filtered Iris Data Frame (Sepal.Length > 5.5):")
print(head(iris_filtered))
correlation_matrix <- cor(iris_filtered[, sapply(iris_filtered, is.numeric)])
print("Correlation Matrix for Numeric Columns:")
print(correlation_matrix)
```

Output:

```
> data(iris)
> iris_filtered <- iris[iris$Sepal.Length > 5.5, ]
> print("Filtered Iris Data Frame (Sepal.Length > 5.5):")
[1] "Filtered Iris Data Frame (Sepal.Length > 5.5):"
> print(head(iris_filtered))
   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
15          5.8         4.0          1.2         0.2     setosa
16          5.7         4.4          1.5         0.4     setosa
19          5.7         3.8          1.7         0.3     setosa
51          7.0         3.2          4.7         1.4 versicolor
52          6.4         3.2          4.5         1.5 versicolor
53          6.9         3.1          4.9         1.5 versicolor
> correlation_matrix <- cor(iris_filtered[, sapply(iris_filtered, is.numeric)])
> print("Correlation Matrix for Numeric Columns:")
[1] "Correlation Matrix for Numeric Columns:"
> print(correlation_matrix)
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  0.21495443    0.7121662  0.52271575
Sepal.Width     0.2149544  1.00000000   -0.1377127  0.01925592
Petal.Length    0.7121662 -0.13771270    1.0000000  0.85330666
Petal.Width     0.5227157  0.01925592    0.8533067  1.00000000
```

14) Create a factor from the Species column of iris. Create a contingency table showing the frequency of each species combined with the Sepal.Width above and below the median.

Code:

```
data(iris)
iris$Species <- as.factor(iris$Species)
median_sepal_width <- median(iris$Sepal.Width)
iris$Width_Category <- ifelse(iris$Sepal.Width > median_sepal_width, "Above Median", "Below Median")
contingency_table <- table(iris$Species, iris$Width_Category)
print("Contingency Table of Species and Sepal.Width Category:")
print(contingency_table)
```

Output:

```
> data(iris)
> iris$Species <- as.factor(iris$Species)
> median_sepal_width <- median(iris$Sepal.Width)
> iris$Width_Category <- ifelse(iris$Sepal.Width > median_sepal_width, "Above Med
ian", "Below Median")
> contingency_table <- table(iris$Species, iris$Width_Category)
> print("Contingency Table of Species and Sepal.Width Category:")
[1] "Contingency Table of Species and Sepal.Width Category:"
> print(contingency_table)

            Above Median Below Median
  setosa              42            8
  versicolor           8           42
  virginica           17           33
```

15) From the mtcars dataset, identify rows with missing values. Replace missing values in mpg and hp with the median of their respective columns.

Code:

```
data(mtcars)
missing_values <- is.na(mtcars)
rows_with_missing <- mtcars[apply(missing_values, 1, any), ]
print("Rows with missing values:")
print(rows_with_missing)
median_mpg <- median(mtcars$mpg, na.rm = TRUE)
median_hp <- median(mtcars$hp, na.rm = TRUE)
mtcars$mpg[is.na(mtcars$mpg)] <- median_mpg
mtcars$hp[is.na(mtcars$hp)] <- median_hp
print("Updated mtcars dataset after replacing missing values:")
print(head(mtcars))
```

Output:

```
> data(mtcars)
> missing_values <- is.na(mtcars)
> rows_with_missing <- mtcars[apply(missing_values, 1, any), ]
> print("Rows with missing values:")
[1] "Rows with missing values:"
> print(rows_with_missing)
 [1] mpg  cyl  disp hp   drat wt   qsec vs   am   gear carb
<0 rows> (or 0-length row.names)
> median_mpg <- median(mtcars$mpg, na.rm = TRUE)
> median_hp <- median(mtcars$hp, na.rm = TRUE)
> mtcars$mpg[is.na(mtcars$mpg)] <- median_mpg
> mtcars$hp[is.na(mtcars$hp)] <- median_hp
> print("Updated mtcars dataset after replacing missing values:")
[1] "Updated mtcars dataset after replacing missing values:"
> print(head(mtcars))
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

17) Create a function that takes a data frame and a column name as input, and converts continuous numeric data into three categories: "Low", "Medium", "High", based on quantiles. Test it on mtcars$hp.

Code:

```
categorize_continuous <- function(data, column_name) {
  if (!column_name %in% names(data)) {
    stop("Column not found in the data frame.")
  }
  column_data <- data[[column_name]]
  quantiles <- quantile(column_data, probs = c(0, 1/3, 2/3, 1), na.rm = TRUE)
  categories <- cut(column_data,
                    breaks = quantiles,
                    labels = c("Low", "Medium", "High"),
                    include.lowest = TRUE)
  return(categories)
}
mtcars$hp_category <- categorize_continuous(mtcars, "hp")
print("Updated mtcars dataset with hp categories:")
print(head(mtcars))
```

Output:

```
> categorize_continuous <- function(data, column_name) {
+   if (!column_name %in% names(data)) {
+     stop("Column not found in the data frame.")
+   }
+   column_data <- data[[column_name]]
+   quantiles <- quantile(column_data, probs = c(0, 1/3, 2/3, 1), na.rm = TRUE)
+   categories <- cut(column_data,
+                     breaks = quantiles,
+                     labels = c("Low", "Medium", "High"),
+                     include.lowest = TRUE)
+
+   return(categories)
+ }
> mtcars$hp_category <- categorize_continuous(mtcars, "hp")
> print("Updated mtcars dataset with hp categories:")
[1] "Updated mtcars dataset with hp categories:"
> print(head(mtcars))
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
                  hp_category
Mazda RX4              Medium
Mazda RX4 Wag          Medium
Datsun 710                Low
Hornet 4 Drive         Medium
Hornet Sportabout      Medium
Valiant                   Low
```

18) From the mtcars dataset, calculate the mean and median mpg for each cyl group using dplyr. Then plot a bar chart showing the differences.

Code:

```
library(dplyr)
library(ggplot2)
data(mtcars)
mpg_summary <- mtcars %>%
  group_by(cyl) %>%
```

```
 summarise(
   Mean_MPG = mean(mpg, na.rm = TRUE),
   Median_MPG = median(mpg, na.rm = TRUE)
 )
print("Mean and Median MPG for each cyl group:")
print(mpg_summary)
```

Output:

```
> library(dplyr)

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union

Warning message:
package 'dplyr' was built under R version 4.4.3
> library(ggplot2)
Warning message:
package 'ggplot2' was built under R version 4.4.3
> data(mtcars)
> mpg_summary <- mtcars %>%
+   group_by(cyl) %>%
+   summarise(
+     Mean_MPG = mean(mpg, na.rm = TRUE),
+     Median_MPG = median(mpg, na.rm = TRUE)
+   )
> print("Mean and Median MPG for each cyl group:")
[1] "Mean and Median MPG for each cyl group:"
> print(mpg_summary)
# A tibble: 3 × 3
    cyl Mean_MPG Median_MPG
  <db l>    <db l>      <db l>
1     4      26.7        26
2     6      19.7        19.7
3     8      15.1        15.2
```

19) Create a function that takes two numeric vectors and returns a list containing the sum, product, difference, and element-wise maximum. Test it on mtcars$mpg and mtcars$hp.

Code: calculate_vector_operations <- function(vec1, vec2) {

```
 if (length(vec1) != length(vec2)) {
   stop("Vectors must be of the same length.")
 }
 result <- list(
   Sum = vec1 + vec2,
   Product = vec1 * vec2,
   Difference = vec1 - vec2,
   Elementwise_Max = pmax(vec1, vec2)
 )
```

```
  return(result)
}
result <- calculate_vector_operations(mtcars$mpg, mtcars$hp)
print("Results of vector operations on mtcars$mpg and mtcars$hp:")
print(result)
```

Output:

```
> calculate_vector_operations <- function(vec1, vec2) {
+   if (length(vec1) != length(vec2)) {
+     stop("Vectors must be of the same length.")
+   }
+   result <- list(
+     Sum = vec1 + vec2,
+     Product = vec1 * vec2,
+     Difference = vec1 - vec2,
+     Elementwise_Max = pmax(vec1, vec2)
+   )
+
+   return(result)
+ }
> result <- calculate_vector_operations(mtcars$mpg, mtcars$hp)
> print("Results of vector operations on mtcars$mpg and mtcars$hp:")
[1] "Results of vector operations on mtcars$mpg and mtcars$hp:"
> print(result)
$Sum
 [1] 131.0 131.0 115.8 131.4 193.7 123.1 259.3  86.4 117.8 142.2 140.8 196.4
[13] 197.3 195.2 215.4 225.4 244.7  98.4  82.4  98.9 118.5 165.5 165.2 258.3
[25] 194.2  93.3 117.0 143.4 279.8 194.7 350.0 130.4

$Product
 [1] 2310.0 2310.0 2120.4 2354.0 3272.5 1900.5 3503.5 1512.8 2166.0 2361.6
[11] 2189.4 2952.0 3114.0 2736.0 2132.0 2236.0 3381.0 2138.4 1580.8 2203.5
[21] 2085.5 2325.0 2280.0 3258.5 3360.0 1801.8 2366.0 3435.2 4171.2 3447.5
[31] 5025.0 2332.6

$Difference
 [1]  -89.0  -89.0  -70.2  -88.6 -156.3  -86.9 -230.7  -37.6  -72.2 -103.8
[11] -105.2 -163.6 -162.7 -164.8 -194.6 -204.6 -215.3  -33.6  -21.6  -31.1
[21]  -75.5 -134.5 -134.8 -231.7 -155.8  -38.7  -65.0  -82.6 -248.2 -155.3
[31] -320.0  -87.6

$Elementwise_Max
 [1] 110 110  93 110 175 105 245  62  95 123 123 180 180 180 205 215 230  66
[19]  52  65  97 150 150 245 175  66  91 113 264 175 335 109
```

20) Write a function that takes a matrix and returns the row with the highest sum of values. Test it on a matrix created from mtcars$mpg, mtcars$hp, and mtcars$cyl.

Code:

```
calculate_vector_operations <- function(vec1, vec2) {
  if (length(vec1) != length(vec2)) {
    stop("Vectors must be of the same length.")
  }
  result <- list(
```

```
  Sum = vec1 + vec2,
  Product = vec1 * vec2,
  Difference = vec1 - vec2,
  Elementwise_Max = pmax(vec1, vec2)
 )

 return(result)
}
result <- calculate_vector_operations(mtcars$mpg, mtcars$hp)
print("Results of vector operations on mtcars$mpg and mtcars$hp:")
print(result)
row_with_highest_sum <- function(mat) {
 row_sums <- rowSums(mat)
 max_row_index <- which.max(row_sums)
 return(mat[max_row_index, , drop = FALSE])  # drop = FALSE to keep it as a matrix
}
mtcars_matrix <- as.matrix(mtcars[, c("mpg", "hp", "cyl")])
highest_sum_row <- row_with_highest_sum(mtcars_matrix)
print("Row with the highest sum of values:")
print(highest_sum_row)
```

Output:

```
> calculate_vector_operations <- function(vec1, vec2) {
+   if (length(vec1) != length(vec2)) {
+     stop("Vectors must be of the same length.")
+   }
+   result <- list(
+     Sum = vec1 + vec2,
+     Product = vec1 * vec2,
+     Difference = vec1 - vec2,
+     Elementwise_Max = pmax(vec1, vec2)
+   )
+
+   return(result)
+ }
> result <- calculate_vector_operations(mtcars$mpg, mtcars$hp)
> print("Results of vector operations on mtcars$mpg and mtcars$hp:")
[1] "Results of vector operations on mtcars$mpg and mtcars$hp:"
> print(result)
$Sum
 [1] 131.0 131.0 115.8 131.4 193.7 123.1 259.3  86.4 117.8 142.2 140.8 196.4
[13] 197.3 195.2 215.4 225.4 244.7  98.4  82.4  98.9 118.5 165.5 165.2 258.3
[25] 194.2  93.3 117.0 143.4 279.8 194.7 350.0 130.4

$Product
 [1] 2310.0 2310.0 2120.4 2354.0 3272.5 1900.5 3503.5 1512.8 2166.0 2361.6
[11] 2189.4 2952.0 3114.0 2736.0 2132.0 2236.0 3381.0 2138.4 1580.8 2203.5
[21] 2085.5 2325.0 2280.0 3258.5 3360.0 1801.8 2366.0 3435.2 4171.2 3447.5
[31] 5025.0 2332.6

$Difference
 [1]   -89.0   -89.0   -70.2   -88.6  -156.3   -86.9  -230.7   -37.6   -72.2  -103.8
[11] -105.2  -163.6  -162.7  -164.8  -194.6  -204.6  -215.3   -33.6   -21.6   -31.1
[21]  -75.5  -134.5  -134.8  -231.7  -155.8   -38.7   -65.0   -82.6  -248.2  -155.3
[31] -320.0   -87.6

$Elementwise_Max
 [1] 110 110  93 110 175 105 245  62  95 123 123 180 180 180 205 215 230  66
```

21) Create a data frame from mtcars and iris. Use merge() to combine them based on a common column name. Handle cases where the column names do not match.

Code:

```
library(dplyr)
data(mtcars)
data(iris)
```

```
mtcars$common_col <- substr(rownames(mtcars), 1, 1)  # First letter of car names
iris$common_col <- substr(iris$Species, 1, 1)  # First letter of species names
print("mtcars dataset:")
print(head(mtcars))
print("iris  dataset:")
print(head(iris))
merged_data <- merge(mtcars, iris, by = "common_col", all = TRUE)
print("Merged dataset:")
```

Output:

```
> library(dplyr)
> data(mtcars)
> data(iris)
> mtcars$common_col <- substr(rownames(mtcars), 1, 1)  # First letter of car name
s
> iris$common_col <- substr(iris$Species, 1, 1)  # First letter of species names
> print("mtcars dataset:")
[1] "mtcars dataset:"
> print(head(mtcars))
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
                  common_col
Mazda RX4                  M
Mazda RX4 Wag              M
Datsun 710                 D
Hornet 4 Drive             H
Hornet Sportabout          H
Valiant                    V
> print("iris dataset:")
[1] "iris dataset:"
> print(head(iris))
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species common_col
1          5.1         3.5          1.4         0.2  setosa          s
2          4.9         3.0          1.4         0.2  setosa          s
3          4.7         3.2          1.3         0.2  setosa          s
4          4.6         3.1          1.5         0.2  setosa          s
5          5.0         3.6          1.4         0.2  setosa          s
6          5.4         3.9          1.7         0.4  setosa          s
> merged_data <- merge(mtcars, iris, by = "common_col", all = TRUE)
> print("Merged dataset:")
[1] "Merged dataset:"
```

22) Create a data frame from the mtcars dataset. Find the car with the highest and lowest mpg for each cyl group using dplyr and store the result in a new data frame.

Code:

```
library(dplyr)
data(mtcars)
mpg_summary <- mtcars %>%
  group_by(cyl) %>%
  summarise(
    highest_mpg = max(mpg),
```

```
  lowest_mpg = min(mpg),
  highest_mpg_car = rownames(mtcars[which.max(mpg), ]),
  lowest_mpg_car = rownames(mtcars[which.min(mpg), ])
 )
print("Summary of highest and lowest mpg for each cyl group:")
print(mpg_summary)
```

Output:

```
> library(dplyr)
> data(mtcars)
> mpg_summary <- mtcars %>%
+   group_by(cyl) %>%
+   summarise(
+     highest_mpg = max(mpg),
+     lowest_mpg = min(mpg),
+     highest_mpg_car = rownames(mtcars[which.max(mpg), ]),
+     lowest_mpg_car = rownames(mtcars[which.min(mpg), ])
+   )
> print("Summary of highest and lowest mpg for each cyl group:")
[1] "Summary of highest and lowest mpg for each cyl group:"
> print(mpg_summary)
# A tibble: 3 × 5
    cyl highest_mpg lowest_mpg highest_mpg_car lowest_mpg_car
  <dbl>       <dbl>      <dbl> <chr>           <chr>
1     4        33.9       21.4 Valiant         Merc 280C
2     6        21.4       17.8 Datsun 710      Valiant
3     8        19.2       10.4 Merc 450SE      Valiant
```

23) Create a matrix from the iris dataset. Write a function that accepts a matrix and returns the row and column indices of the minimum and maximum values.

Code:

```
data(iris)
iris_matrix <- as.matrix(iris[, -5]) # Exclude the Species column
find_min_max_indices <- function(mat) {
 min_value <- min(mat)
 min_indices <- which(mat == min_value, arr.ind = TRUE)
 max_value <- max(mat)
 max_indices <- which(mat == max_value, arr.ind = TRUE)
 return(list(
  min_value = min_value,
  min_indices = min_indices,
  max_value = max_value,
  max_indices = max_indices
 ))
}
```

```
result <- find_min_max_indices(iris_matrix)
print("Minimum value and its indices:")
print(result$min_value)
print(result$min_indices)
print("Maximum value and its indices:")
print(result$max_value)
print(result$max_indices)
```

Output:

```
> data(iris)
> iris_matrix <- as.matrix(iris[, -5])  # Exclude the Species column
> find_min_max_indices <- function(mat) {
+   min_value <- min(mat)
+   min_indices <- which(mat == min_value, arr.ind = TRUE)
+   max_value <- max(mat)
+   max_indices <- which(mat == max_value, arr.ind = TRUE)
+   return(list(
+     min_value = min_value,
+     min_indices = min_indices,
+     max_value = max_value,
+     max_indices = max_indices
+   ))
+ }
> result <- find_min_max_indices(iris_matrix)
> print("Minimum value and its indices:")
[1] "Minimum value and its indices:"
> print(result$min_value)
[1] 0.1
> print(result$min_indices)
     row col
[1,]  10   4
[2,]  13   4
[3,]  14   4
[4,]  33   4
[5,]  38   4
> print("Maximum value and its indices:")
[1] "Maximum value and its indices:"
> print(result$max_value)
[1] 7.9
> print(result$max_indices)
     row col
[1,] 132   1
```

24) Create a correlation matrix for mtcars using cor(). Identify the two columns with the highest and lowest correlation and display the result in a structured format.

Code:

```
data(mtcars)
correlation_matrix <- cor(mtcars)
print("Correlation Matrix:")
print(correlation_matrix)
upper_tri <- correlation_matrix[upper.tri(correlation_matrix)]
highest_correlation <- max(upper_tri)
```

```
lowest_correlation <- min(upper_tri)
highest_indices <- which(correlation_matrix == highest_correlation, arr.ind = TRUE)
lowest_indices <- which(correlation_matrix == lowest_correlation, arr.ind = TRUE)
cat("Highest Correlation:\n")
cat("Columns:", colnames(mtcars)[highest_indices[1, 1]], "and",
colnames(mtcars)[highest_indices[1, 2]], "\n")
cat("Correlation Value:", highest_correlation, "\n\n")
cat("Lowest Correlation:\n")
cat("Columns:", colnames(mtcars)[lowest_indices[1, 1]], "and",
colnames(mtcars)[lowest_indices[1, 2]], "\n")
cat("Correlation Value:", lowest_correlation, "\n")
```

Output:

```
> data(iris)
> iris_matrix <- as.matrix(iris[, -5])  # Exclude the Species column
> find_min_max_indices <- function(mat) {
+   min_value <- min(mat)
+   min_indices <- which(mat == min_value, arr.ind = TRUE)
+   max_value <- max(mat)
+   max_indices <- which(mat == max_value, arr.ind = TRUE)
+   return(list(
+     min_value = min_value,
+     min_indices = min_indices,
+     max_value = max_value,
+     max_indices = max_indices
+   ))
+ }
> result <- find_min_max_indices(iris_matrix)
> print("Minimum value and its indices:")
[1] "Minimum value and its indices:"
> print(result$min_value)
[1] 0.1
> print(result$min_indices)
     row col
[1,]  10   4
[2,]  13   4
[3,]  14   4
[4,]  33   4
[5,]  38   4
> print("Maximum value and its indices:")
[1] "Maximum value and its indices:"
> print(result$max_value)
[1] 7.9
> print(result$max_indices)
     row col
[1,] 132   1
> data(mtcars)
> correlation_matrix <- cor(mtcars)
> print("Correlation Matrix:")
> print(correlation_matrix)
            mpg        cyl       disp         hp       drat         wt
mpg   1.0000000 -0.8521620 -0.8475514 -0.7761684  0.68117191 -0.8676594
cyl  -0.8521620  1.0000000  0.9020329  0.8324475 -0.69993811  0.7824958
disp -0.8475514  0.9020329  1.0000000  0.7909486 -0.71021393  0.8879799
hp   -0.7761684  0.8324475  0.7909486  1.0000000 -0.44875912  0.6587479
drat  0.6811719 -0.6999381 -0.7102139 -0.4487591  1.00000000 -0.7124406
wt   -0.8676594  0.7824958  0.8879799  0.6587479 -0.71244065  1.0000000
qsec  0.4186840 -0.5912421 -0.4336979 -0.7082234  0.09120476 -0.1747159
vs    0.6640389 -0.8108118 -0.7104159 -0.7230967  0.44027846 -0.5549157
am    0.5998324 -0.5226070 -0.5912270 -0.2432043  0.71271113 -0.6924953
gear  0.4802848 -0.4926866 -0.5555692 -0.1257043  0.69961013 -0.5832870
carb -0.5509251  0.5269883  0.3949769  0.7498125 -0.09078980  0.4276059
             qsec         vs         am       gear        carb
mpg   0.41868403  0.6640389  0.5998324  0.4802848 -0.5509251
cyl  -0.59124207 -0.8108118 -0.5226070 -0.4926866  0.5269883
disp -0.43369788 -0.7104159 -0.5912270 -0.5555692  0.3949769
hp   -0.70822339 -0.7230967 -0.2432043 -0.1257043  0.7498125
drat  0.09120476  0.4402785  0.7127111  0.6996101 -0.0907898
wt   -0.17471588 -0.5549157 -0.6924953 -0.5832870  0.4276059
qsec  1.00000000  0.7445354 -0.2298609 -0.2126822 -0.6562492
vs    0.74453544  1.0000000  0.1683451  0.2060233 -0.5696071
am   -0.22986086  0.1683451  1.0000000  0.7940588  0.0575343
gear -0.21268223  0.2060233  0.7940588  1.0000000  0.2740728
carb -0.65624923 -0.5696071  0.0575343  0.2740728  1.0000000
> upper_tri <- correlation_matrix[upper.tri(correlation_matrix)]
> highest_correlation <- max(upper_tri)
> lowest_correlation <- min(upper_tri)
> highest_indices <- which(correlation_matrix == highest_correlation, arr.ind = TRUE)
> lowest_indices <- which(correlation_matrix == lowest_correlation, arr.ind = TRUE)
> cat("Highest Correlation:\n")
Highest Correlation:
> cat("Columns:", colnames(mtcars)[highest_indices[1, 1]], "and", colnames(mtcars)[highest_indices[1, 2]], "\n")
Columns: disp and cyl
> cat("Correlation Value:", highest_correlation, "\n\n")
Correlation Value: 0.9020329

> cat("Lowest Correlation:\n")
Lowest Correlation:
> cat("Columns:", colnames(mtcars)[lowest_indices[1, 1]], "and", colnames(mtcars)[lowest_indices[1, 2]], "\n")
Columns: wt and mpg
> cat("Correlation Value:", lowest_correlation, "\n")
Correlation Value: -0.8676594
```

25) From the iris dataset, create a bar chart showing the average Sepal.Length for each species using ggplot2. Add error bars showing the standard deviation.

Code:

```
library(ggplot2)
library(dplyr)
data(iris)
summary_data <- iris %>%
  group_by(Species) %>%
  summarise(
    avg_sepal_length = mean(Sepal.Length),
    sd_sepal_length = sd(Sepal.Length)
  )
ggplot(summary_data, aes(x = Species, y = avg_sepal_length)) +
  geom_bar(stat = "identity", fill = "skyblue", width = 0.7) +
  geom_errorbar(aes(ymin = avg_sepal_length - sd_sepal_length,
                    ymax = avg_sepal_length + sd_sepal_length),
                width = 0.2) +
  labs(title = "Average Sepal Length by Species",
       x = "Species",
       y = "Average Sepal Length") +
  theme_minimal()
```

Output:

```
> library(ggplot2)
> library(dplyr)
> data(iris)
> summary_data <- iris %>%
+   group_by(Species) %>%
+   summarise(
+     avg_sepal_length = mean(Sepal.Length),
+     sd_sepal_length = sd(Sepal.Length)
+   )
> ggplot(summary_data, aes(x = Species, y = avg_sepal_length)) +
+   geom_bar(stat = "identity", fill = "skyblue", width = 0.7) +
+   geom_errorbar(aes(ymin = avg_sepal_length - sd_sepal_length,
+                     ymax = avg_sepal_length + sd_sepal_length),
+                 width = 0.2) +
+   labs(title = "Average Sepal Length by Species",
+        x = "Species",
+        y = "Average Sepal Length") +
+   theme_minimal()
```

26) Create a matrix from the mtcars dataset. Write a function that accepts a matrix and a row index, and returns the sum of that row. Use apply() for implementation.

Code:

```
data(mtcars)
mtcars_matrix <- as.matrix(mtcars)
row_sum <- function(mat, row_index) {
  if (row_index < 1 || row_index > nrow(mat)) {
    stop("Invalid row index.")
  }
  row_sum_value <- apply(mat[row_index, , drop = FALSE], 1, sum)

  return(row_sum_value)
}
row_index <- 1  # Specify the row index you want to sum
result <- row_sum(mtcars_matrix, row_index)
cat("Sum of row", row_index, "in the mtcars matrix:", result, "\n")
```

Output:

```
> library(ggplot2)
> library(dplyr)
> data(iris)
> summary_data <- iris %>%
+   group_by(Species) %>%
+   summarise(
+     avg_sepal_length = mean(Sepal.Length),
+     sd_sepal_length = sd(Sepal.Length)
+   )
> ggplot(summary_data, aes(x = Species, y = avg_sepal_length)) +
+   geom_bar(stat = "identity", fill = "skyblue", width = 0.7) +
+   geom_errorbar(aes(ymin = avg_sepal_length - sd_sepal_length,
+                     ymax = avg_sepal_length + sd_sepal_length),
+                 width = 0.2) +
+   labs(title = "Average Sepal Length by Species",
+        x = "Species",
+        y = "Average Sepal Length") +
+   theme_minimal()
> data(mtcars)
> mtcars_matrix <- as.matrix(mtcars)
> row_sum <- function(mat, row_index) {
+   if (row_index < 1 || row_index > nrow(mat)) {
+     stop("Invalid row index.")
+   }
+   row_sum_value <- apply(mat[row_index, , drop = FALSE], 1, sum)
+
+   return(row_sum_value)
+ }
> row_index <- 1  # Specify the row index you want to sum
> result <- row_sum(mtcars_matrix, row_index)
> cat("Sum of row", row_index, "in the mtcars matrix:", result, "\n")
Sum of row 1 in the mtcars matrix: 328.98
```

27) Create a 3D array using iris$Sepal.Length, iris$Sepal.Width, and iris$Petal.Length. Write a function that returns the mean for each layer and each row in the array.

Code:

```
data(iris)
iris_array <- array(c(iris$Sepal.Length, iris$Sepal.Width, iris$Petal.Length),
                    dim = c(3, 50, 3),  # 3 rows (for Sepal.Length, Sepal.Width, Petal.Length),
50 samples, 3 species
                    dimnames = list(c("Sepal.Length", "Sepal.Width", "Petal.Length"),
                                    NULL,
                                    levels(iris$Species)))
mean_layer_row <- function(arr) {
  layer_means <- apply(arr, 3, mean)  # Mean across the third dimension (layers)
```

```
  row_means <- apply(arr, 1, mean)  # Mean across the first dimension (rows)

  return(list(layer_means = layer_means, row_means = row_means))
}
result <- mean_layer_row(iris_array)
cat("Mean for each layer (species):\n")
print(result$layer_means)
cat("\nMean for each row (measurements):\n")
print(result$row_means)
```

Output:

```
> data(iris)
> iris_array <- array(c(iris$Sepal.Length, iris$Sepal.Width, iris$Petal.Length),
+                     dim = c(3, 50, 3),  # 3 rows (for Sepal.Length, Sepal.Widt
h, Petal.Length), 50 samples, 3 species
+                     dimnames = list(c("Sepal.Length", "Sepal.Width", "Petal.Len
gth"),
+                                     NULL,
+                                     levels(iris$Species)))
> mean_layer_row <- function(arr) {
+   layer_means <- apply(arr, 3, mean)  # Mean across the third dimension (layer
s)
+   row_means <- apply(arr, 1, mean)  # Mean across the first dimension (rows)
+
+   return(list(layer_means = layer_means, row_means = row_means))
+ }
> result <- mean_layer_row(iris_array)
> cat("Mean for each layer (species):\n")
Mean for each layer (species):
> print(result$layer_means)
    setosa versicolor  virginica
  5.843333   3.057333   3.758000
> cat("\nMean for each row (measurements):\n")

Mean for each row (measurements):
> print(result$row_means)
Sepal.Length  Sepal.Width Petal.Length
    4.200667     4.228000     4.230000
```

28) From the mtcars dataset, create a scatter plot showing the relationship between mpg and hp using ggplot2. Add a regression line with confidence intervals.

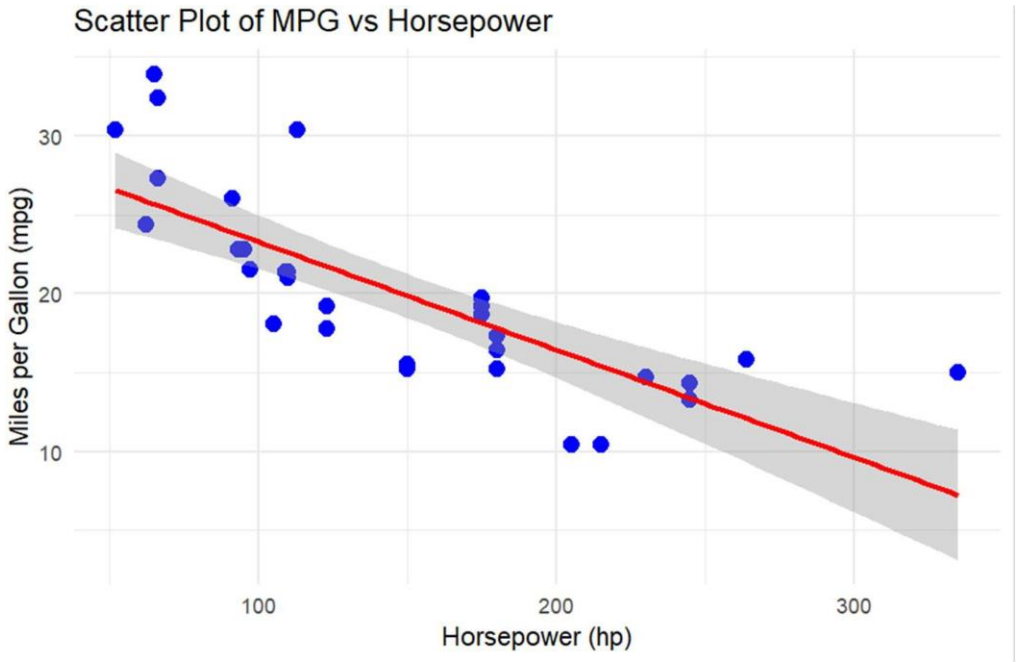
Code:

```
library(ggplot2)
data(mtcars)
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point(color = "blue", size = 3) +
  geom_smooth(method = "lm", se = TRUE, color = "red") +
  labs(title = "Scatter Plot of MPG vs Horsepower",
       x = "Horsepower (hp)",
       y = "Miles per Gallon (mpg)") +
```

theme_minimal()

Output:

```
> library(ggplot2)
> data(mtcars)
> ggplot(mtcars, aes(x = hp, y = mpg)) +
+   geom_point(color = "blue", size = 3) +
+   geom_smooth(method = "lm", se = TRUE, color = "red") +
+   labs(title = "Scatter Plot of MPG vs Horsepower",
+        x = "Horsepower (hp)",
+        y = "Miles per Gallon (mpg)") +
+   theme_minimal()
`geom_smooth()` using formula = 'y ~ x'
```



29) Create a summary report for the mtcars dataset using rmarkdown. Include a table of descriptive statistics and a correlation plot.

Code:

title: "MTCars Summary"

author: "Manasvi Sawant"

date: "2025-03-18"

output: html_document

#Intoduction

This report provides a summary of the mtcars dataset, including descriptive statistics and a correlation plot

data(mtcars)

library(ggplot2)

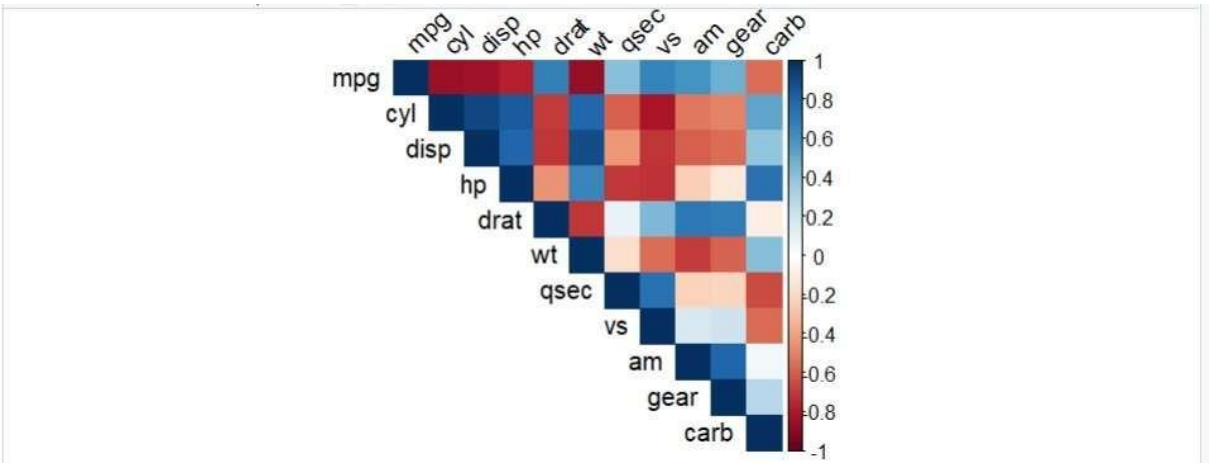
library(corrplot)

```
summary(mtcars)
library(knitr)
kable(summary(mtcars), caption = "Descriptive Statistics of mtcars Dataset")
corr_matrix<-cor(mtcars)
corrplot(corr_matrix, method = "color", type = "upper",
rl.col="black", tl.srt = 45)
```

Output:

Table: Descriptive Statistics of mtcars Dataset

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
	Min. :10.40	Min. :4.000	Min. : 71.1	Min. : 52.0	Min. :2.760	Min. :1.513	Min. :14.50	Min. :0.0000	Min. :0.0000	Min. :3.000	Min. :1.000
	1st Qu.:15.43	1st Qu.:4.000	1st Qu.:120.8	1st Qu.: 96.5	1st Qu.:3.080	1st Qu.:2.581	1st Qu.:16.89	1st Qu.:0.0000	1st Qu.:0.0000	1st Qu.:3.000	1st Qu.:2.000
	Median :19.20	Median :6.000	Median :196.3	Median :123.0	Median :3.695	Median :3.325	Median :17.71	Median :0.0000	Median :0.0000	Median :4.000	Median :2.000
	Mean :20.09	Mean :6.188	Mean :230.7	Mean :146.7	Mean :3.597	Mean :3.217	Mean :17.85	Mean :0.4375	Mean :0.4062	Mean :3.688	Mean :2.812
	3rd Qu.:22.80	3rd Qu.:8.000	3rd Qu.:326.0	3rd Qu.:180.0	3rd Qu.:3.920	3rd Qu.:3.610	3rd Qu.:18.90	3rd Qu.:1.0000	3rd Qu.:1.0000	3rd Qu.:4.000	3rd Qu.:4.000
	Max. :33.90	Max. :8.000	Max. :472.0	Max. :335.0	Max. :4.930	Max. :5.424	Max. :22.90	Max. :1.0000	Max. :1.0000	Max. :5.000	Max. :8.000



30) Create a function that accepts a vector and a value. The function should return the index position of the value if found; otherwise, return "Not Found". Test it on mtcars$mpg.

Code:

```
find_value_index <- function(vec, value) {
  index <- which(vec == value)
  if (length(index) > 0) {
    return(index)
  } else {
    return("Not Found")
  }
}
value_to_find <- 21.0
result <- find_value_index(mtcars$mpg, value_to_find)
cat("Result for value", value_to_find, "in mtcars$mpg:", result, "\n")
```

Output:

```
> find_value_index <- function(vec, value) {
+   index <- which(vec == value)
+   if (length(index) > 0) {
+     return(index)
+   } else {
+     return("Not Found")
+   }
+ }
> value_to_find <- 21.0
> result <- find_value_index(mtcars$mpg, value_to_find)
> cat("Result for value", value_to_find, "in mtcars$mpg:", result, "\n")
Result for value 21 in mtcars$mpg: 1 2
```

Conclusion: In this experiment we learn about implementing the data structures in R on datasets. About Vectors, Lists, Dataframes, Matrices, Arrays, Factors, Tibbles data structres in R programming.

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				