**Project of R Lab**

**Exploratory Data Analysis (EDA) Using R: A case study on the mtcars Dataset**

Subject:-SATAISTICAL TECHNIQUES USING R LAB (24CAP-614)

[Submitted by](https://www.google.com/search?sca_esv=fa4253f7b1e1cfcc&rlz=1C1ONGR_enIN1115IN1122&sxsrf=ADLYWIKbkgQ9fqoKeV4KBN8U_vxUR7BSTA:1729943473185&q=submitted+by&spell=1&sa=X&ved=2ahUKEwiLmM3R_auJAxXFSmwGHQpWA0cQkeECKAB6BAgKEAE):-Anuj and Mannat Mahajan(24MCI10020,24MCI10032)

SECTION\GROUP:24MAM1(A) BRANCH:MCA(AIML)

Github:- <https://github.com/anujdhiman28/Exploratory-Data-Analysis-EDA-Using-R-> A-case-study-on-the-mtcars-Dataset.git

**Introduction**

Exploratory Data Analysis (EDA) is an essential step in data analysis that helps analysts understand the structure, patterns, relationships, and characteristics of a dataset. EDA can be visual and quantitative, often using graphs, summary statistics, and tables to discover insights about the data. In this report, we perform an exploratory data analysis on the **mtcars dataset**, which is available in R by default. The **mtcars** dataset contains various attributes of different car models from the 1970s, including **miles per gallon (mpg)**, **number of cylinders (cyl)**, **horsepower (hp)**, and more.

This report covers the following steps:

1. **Installation of R and Required Packages**.
2. **Downloading the dataset and exploring its dimensions**.
3. **Applying arithmetic operations and mathematical functions**.
4. **Data frame creation and manipulation**.
5. **Looping in R**.
6. **Creating visualizations: Histograms and Boxplots**.
7. **Exploring relationships between variables**.
8. **Conclusion and insights**.

**1. Installation of R and Required Packages**

Before starting the data analysis, it is important to ensure that R is properly installed along with the required packages. R can be downloaded from CRAN (Comprehensive R Archive Network). RStudio is a popular Integrated Development Environment (IDE) for R that simplifies working with R scripts and packages.

**Required Packages:**

We will use the following packages for our analysis:

* dplyr: For data manipulation tasks.
* ggplot2: For creating visualizations.
* tibble: For improved data frames.

To install and load these packages, use the following commands in R:

install.packages(c("dplyr", "ggplot2", "tibble"))

library(dplyr)

library(ggplot2)

library(tibble)

Once installed, we can begin working with the **mtcars** dataset.

**2. Loading and Exploring the mtcars Dataset**

The **mtcars** dataset is built into R, so we do not need to download it. It can be accessed directly using the data() function.

# Load the mtcars dataset

data(mtcars)

# Display the first few rows of the dataset

head(mtcars)

The dataset contains the following columns:

* **mpg**: Miles per gallon.
* **cyl**: Number of cylinders.
* **hp**: Horsepower.
* **drat**: Rear axle ratio.
* **wt**: Weight (in 1000 lbs).
* **qsec**: Quarter mile time.
* **vs**: Engine (0 = V/S, 1 = V).
* **am**: Transmission (0 = automatic, 1 = manual).
* **gear**: Number of forward gears.
* **carb**: Number of carburetors.

We can check the dimensions of the dataset and explore its structure using the dim() and glimpse() functions:

# Get the dimensions of the dataset

dim(mtcars)

**3. Arithmetic Operations and Miscellaneous Mathematical Functions**

R supports a wide range of arithmetic operations and mathematical functions. Let's start by performing some basic operations and applying mathematical functions to the **mtcars** dataset.

**Basic Arithmetic Operations:**

# Arithmetic operations on variables

x <- 10

y <- 5

# Perform sum, difference, product, quotient, and modulus

sum\_result <- x + y

diff\_result <- x - y

prod\_result <- x \* y

quot\_result <- x / y

mod\_result <- x %% y

# Print results

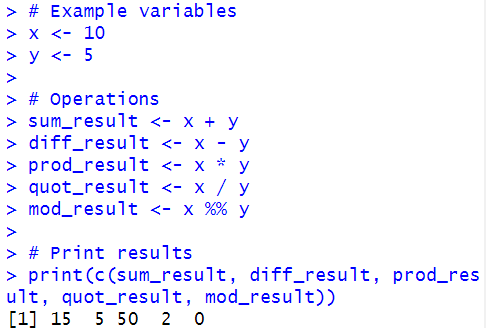
print(sum\_result)

print(diff\_result)

print(prod\_result)

print(quot\_result)

print(mod\_result)



**Mathematical Functions:**

We can use several mathematical functions in R like sqrt(), log(), and exp():

# Apply miscellaneous mathematical functions

sqrt\_val <- sqrt(16)

log\_val <- log(100)

exp\_val <- exp(1)

# Print the results

print(sqrt\_val)

print(log\_val)

print(exp\_val)

A close-up of a computer code

Description automatically generated

These functions are useful in various statistical analyses and calculations that we may perform later.

**4. Data Frame Creation and Manipulation**

In R, data frames are the most common structure for holding tabular data. Let’s create a simple data frame and perform some manipulation tasks.

**Creating a Data Frame:**

# Creating a data frame manually

car\_data <- data.frame(

Model = c("Car1", "Car2", "Car3"),

MPG = c(22, 18, 30),

Cylinders = c(4, 6, 4),

Horsepower = c(110, 130, 95)

)

# View the data frame

print(car\_data)

A screenshot of a computer code

Description automatically generated

**5. Looping in R**

R allows you to use loops to automate repetitive tasks. Below is an example of how to loop through the **mtcars** dataset and perform an operation.

**Example Loop:**

We can use a for loop to check whether the cars in the dataset have **mpg** greater than 20 and print those car models.

# Loop through the mpg column and print car names with mpg > 20

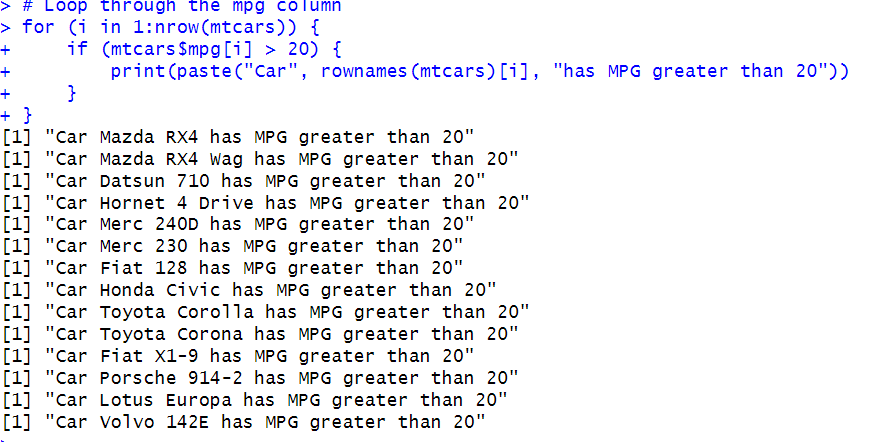
for (i in 1:nrow(mtcars)) {

if (mtcars$mpg[i] > 20) {

print(paste("Car", rownames(mtcars)[i], "has MPG greater than 20"))

}

}



This loop iterates through each row of the **mtcars** dataset, checking the **mpg** value. If the value is greater than 20, it prints the corresponding car name.

**6. Creating a Histogram**

Histograms are a powerful way to visualize the distribution of data. Let’s create a histogram to visualize the distribution of **mpg** values in the **mtcars** dataset.

# Create a histogram of the mpg variable

hist(mtcars$mpg,

main = "Histogram of Miles Per Gallon (MPG)",

col = "blue",

xlab = "Miles Per Gallon",

border = "black",

xlim = c(10, 35),

ylim = c(0, 10),

breaks = 10)

**Explanation of Parameters:**

* **main**: Title of the chart.
* **col**: Color of the bars.
* **xlab**: Label for the horizontal axis (MPG).
* **border**: Color of the border of the bars.
* **xlim**: Limits of the x-axis.
* **ylim**: Limits of the y-axis.
* **breaks**: Number of bins or width of the bars.

A graph of a graph of miles per gallon

Description automatically generated

The histogram shows the distribution of **mpg** values across the cars, with most cars having a **mpg** between 15 and 25.

**7. Creating a Boxplot**

Boxplots are useful for visualizing the spread and identifying outliers in the data. Let’s create a boxplot to explore the relationship between **mpg** and **cyl** (number of cylinders).

# Boxplot of mpg by number of cylinders

boxplot(mpg ~ cyl,

data = mtcars,

main = "Boxplot of MPG by Cylinders",

xlab = "Number of Cylinders",

ylab = "Miles Per Gallon",

col = c("lightblue", "lightgreen", "lightcoral"),

border = "darkblue")

A diagram of a graph

Description automatically generated

The boxplot shows how **mpg** varies across different numbers of cylinders. We can observe that cars with 4 cylinders tend to have higher **mpg** compared to cars with 6 or 8 cylinders.

**8. Exploring Relationships Between Variables**

Now, let’s explore the relationships between different numerical variables using a **scatter plot**. We will plot **horsepower (hp)** against **mpg** to visualize how these two variables are related.

# Scatter plot of Horsepower vs. Miles Per Gallon

ggplot(mtcars, aes(x = hp, y = mpg)) +

geom\_point(color = "blue") +

ggtitle("Scatter Plot of HP vs MPG") +

xlab("Horsepower") +

ylab("Miles Per Gallon")

This scatter plot visualizes the negative correlation between **horsepower** and **mpg**: As **horsepower** increases, **mpg** tends to decrease