PCET's

PIMPRI CHINCHWAD UNIVERSITY

Department of CSE - Artificial Intelligence & Data Science

PCET's Pimpri Chinchwad University

Learn | Grow | Achieve

LAB MANUAL

UBTDS323 - Data Science and Analytics Laboratory

Academic Year 2025–2026

Prepared By:

Prof. Tushar R. Mahore
Assistant Professor
Department of CSE (AI & DS)
Pimpri Chinchwad University

Contents

Practical 1: Introduction to R and RStudio

Aim

To install R and RStudio, explore the interface, and perform basic operations using R.

Objective

- Understand the purpose of R and its importance in data science.
- Familiarize with the RStudio IDE (console, script editor, environment, plots).
- Learn about different data types and variable assignments.
- Execute basic arithmetic and relational operations.
- Load and explore an inbuilt dataset in R.

Software Required

- R (version 4.0 or above) – Open-source language for statistical computing and graphics.
- RStudio IDE – Integrated development environment for R.
- Packages: Base R functions are sufficient for this practical.

Theory

Introduction to R

R is a programming language and environment designed for statistical analysis, data visualization, and machine learning. It is widely used in academia, research, and industry because of its strong data handling and graphical capabilities.

Why R?

- Open-source and free.
- Rich collection of packages (CRAN, Bioconductor).
- Strong support for statistical modeling and visualization.
- Easy integration with Python, SQL, Excel, and cloud tools.

Installation of R and RStudio (Windows)

Follow these steps to install R and RStudio on a Windows system.

1. To install R, go to `https://cran.r-project.org/`

Figure 1: CRAN R Project website

2. Select "Download R for Windows".

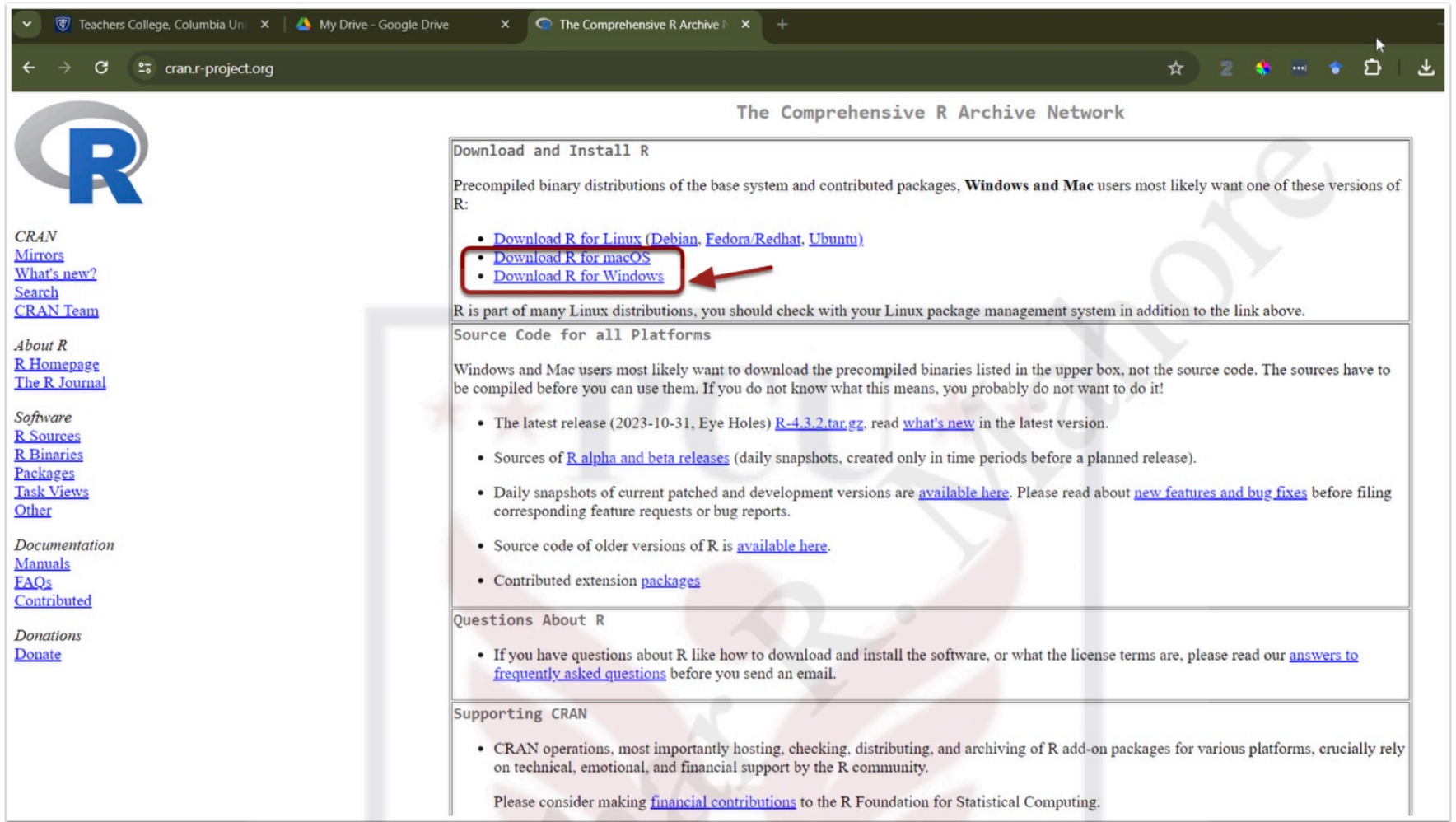

Figure 2: Download R for Windows link

3. Install R Click on install R for the first time.

Figure 3: Install R for first time

4. Click Download R for Windows. Open the downloaded file.

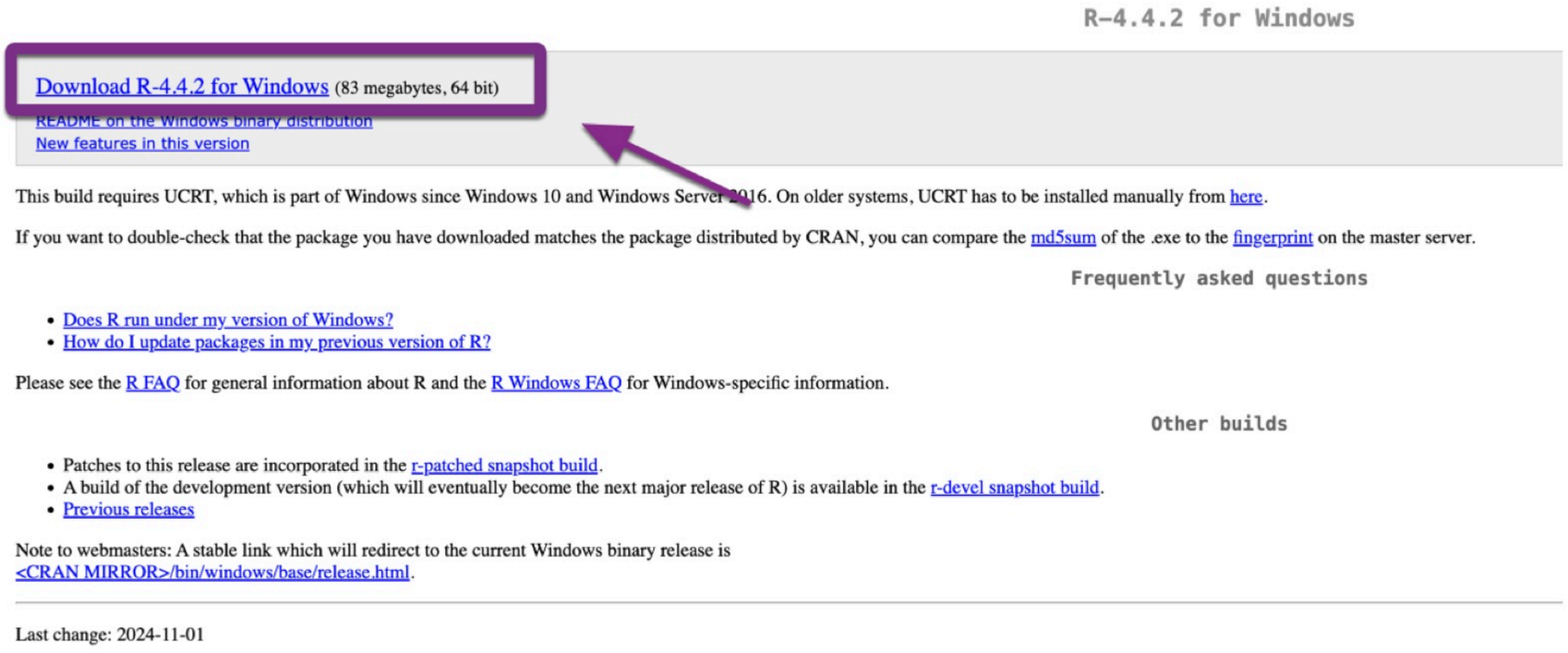

Figure 4: Latest R version download link

5. Run the installer once downloaded and select language.

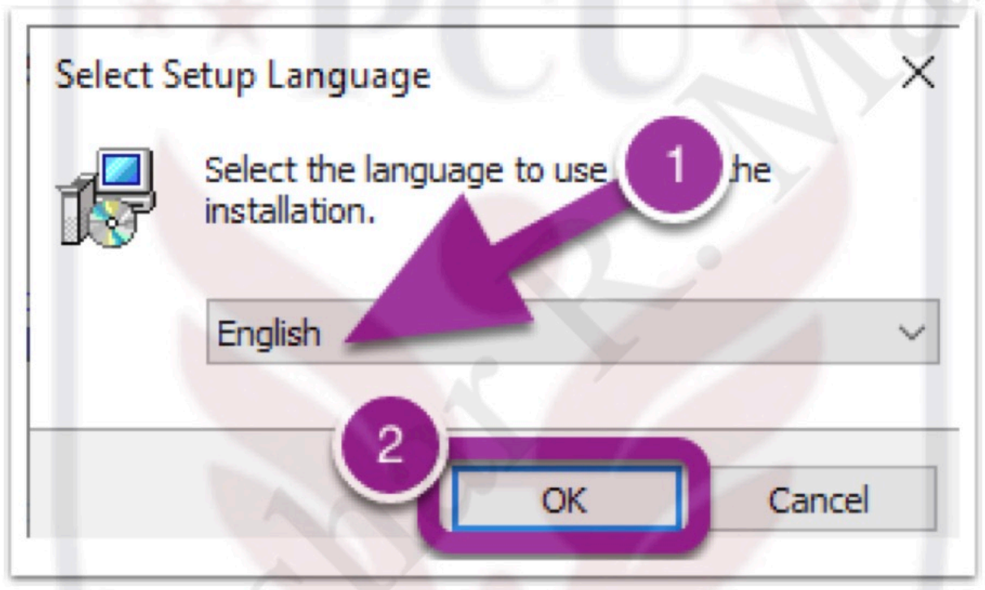

Figure 5: R installer file and language selection

6. Step 6: Click Next.

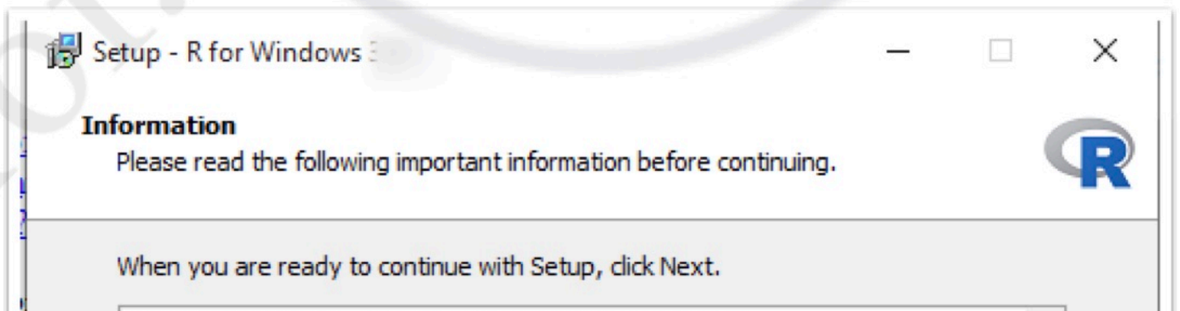

Figure 6: Click Next

7. Select where you would like R to be installed. It will default to your Program Files on your C Drive. Click Next.

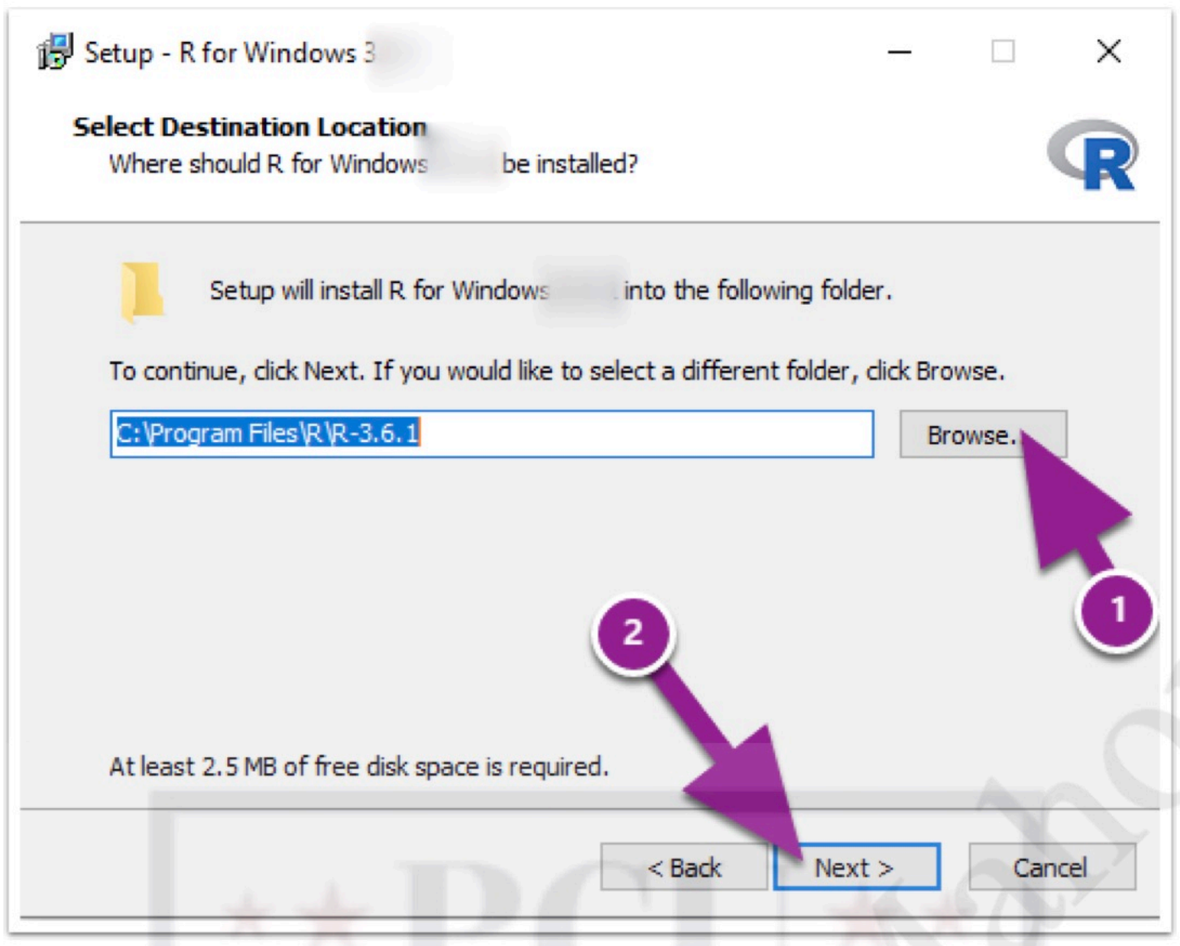

Figure 7: Select Destination

8. Step 8: You can then choose which installation you would like.

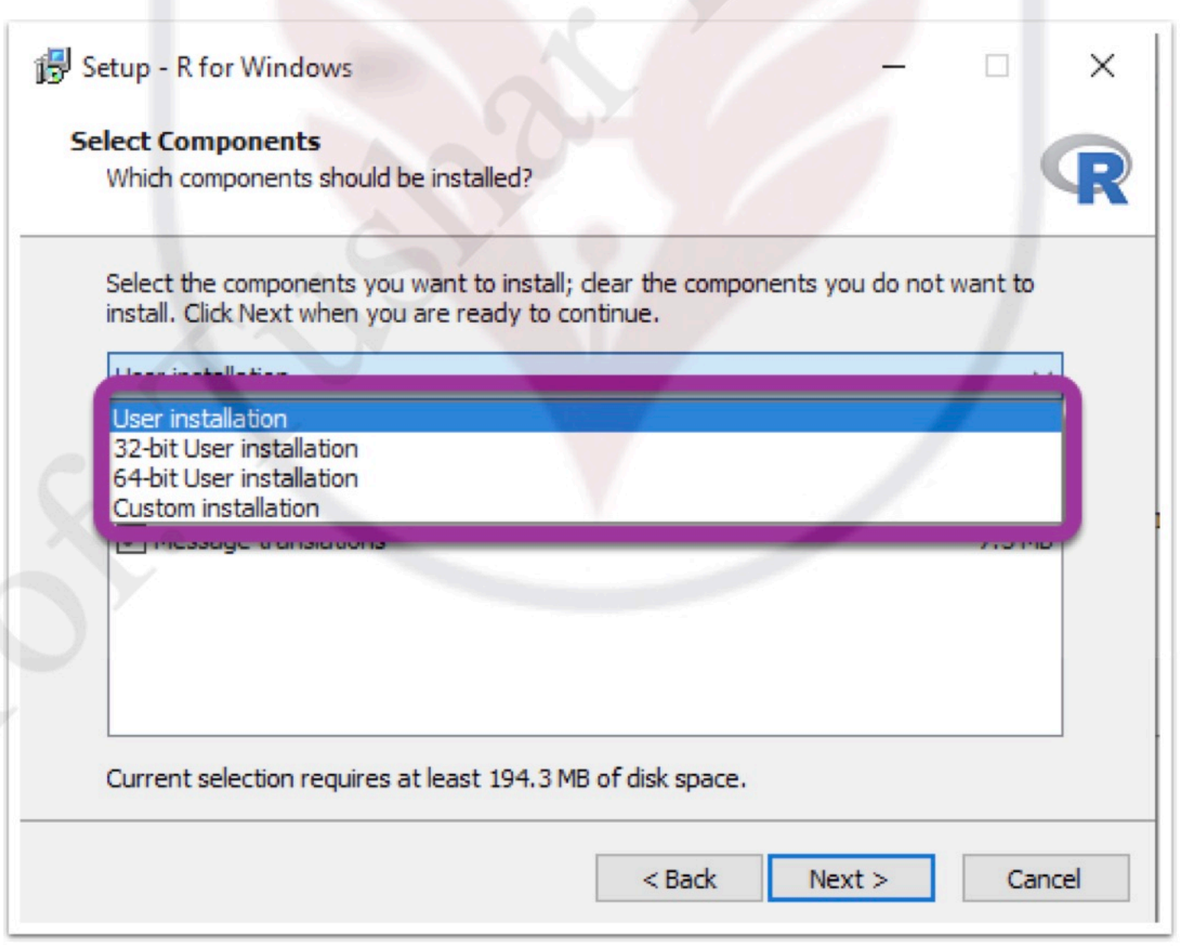

Figure 8: Choose installation type

9. (Optional) If your computer is a 64-bit, you can choose the 64-bit User Installation. Then click Next.

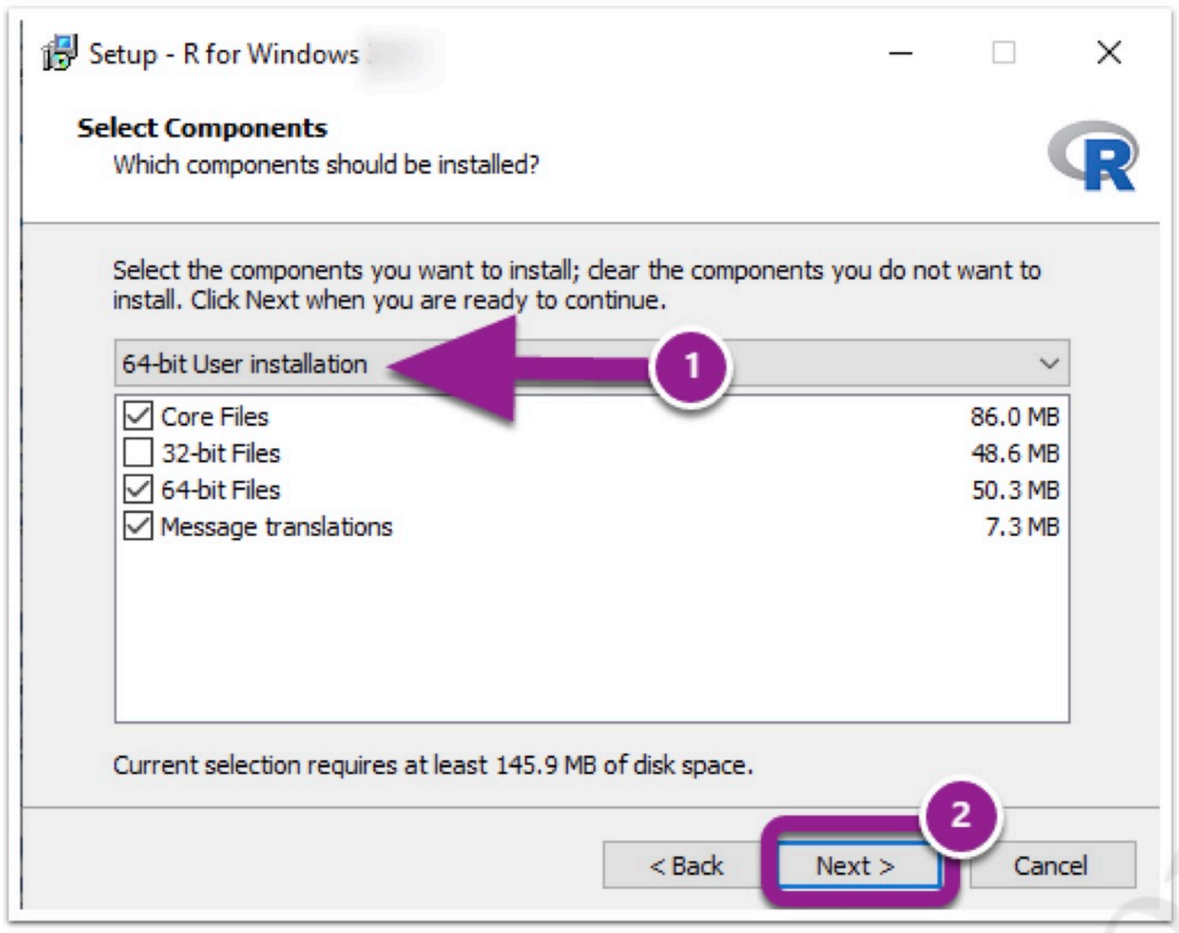

Figure 9: Select components

10. Then specify if you want to customized your startup or just use the defaults. Then click Next.

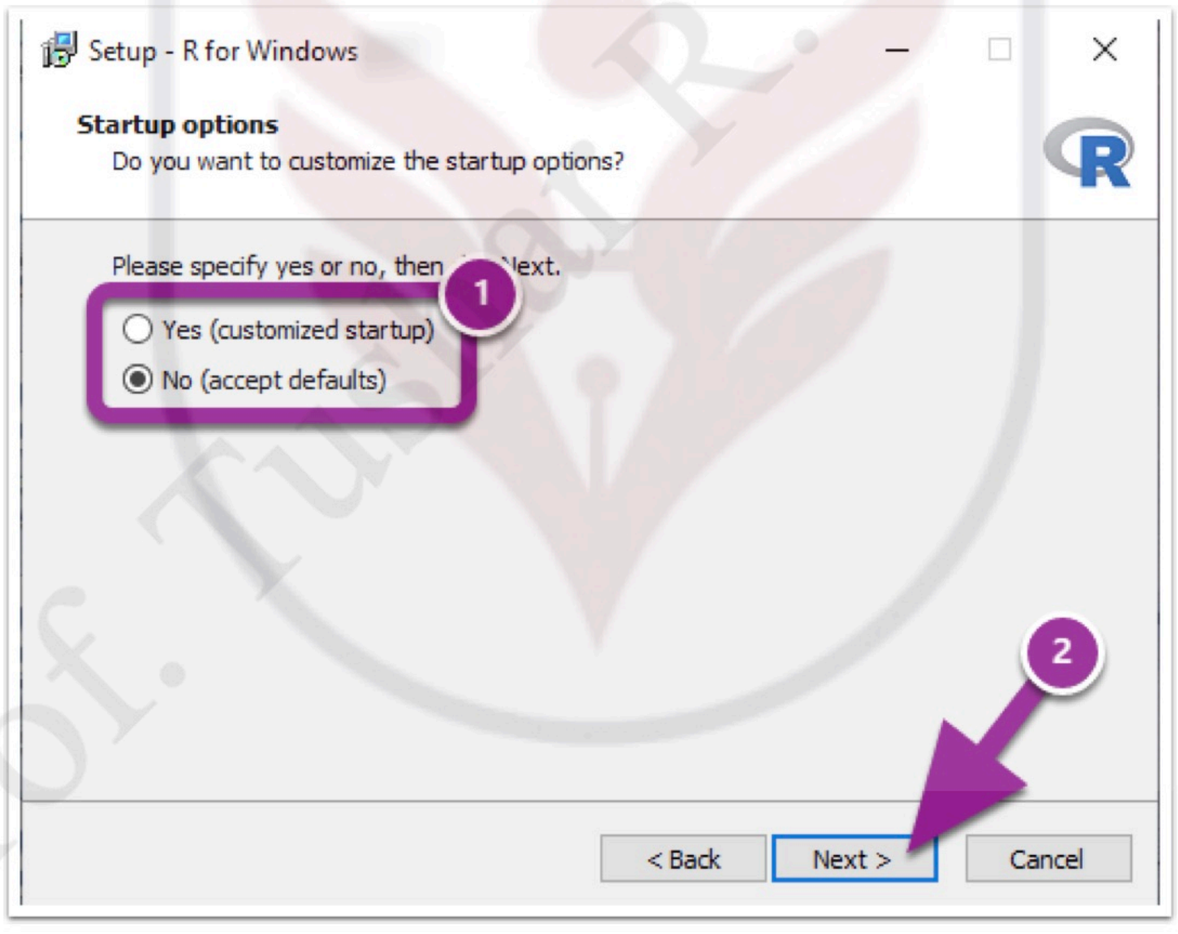

Figure 10: Startup options

11. Then you can choose the folder that you want R to be saved within or the default if the R folder that was created. Once you have finished, click Next.

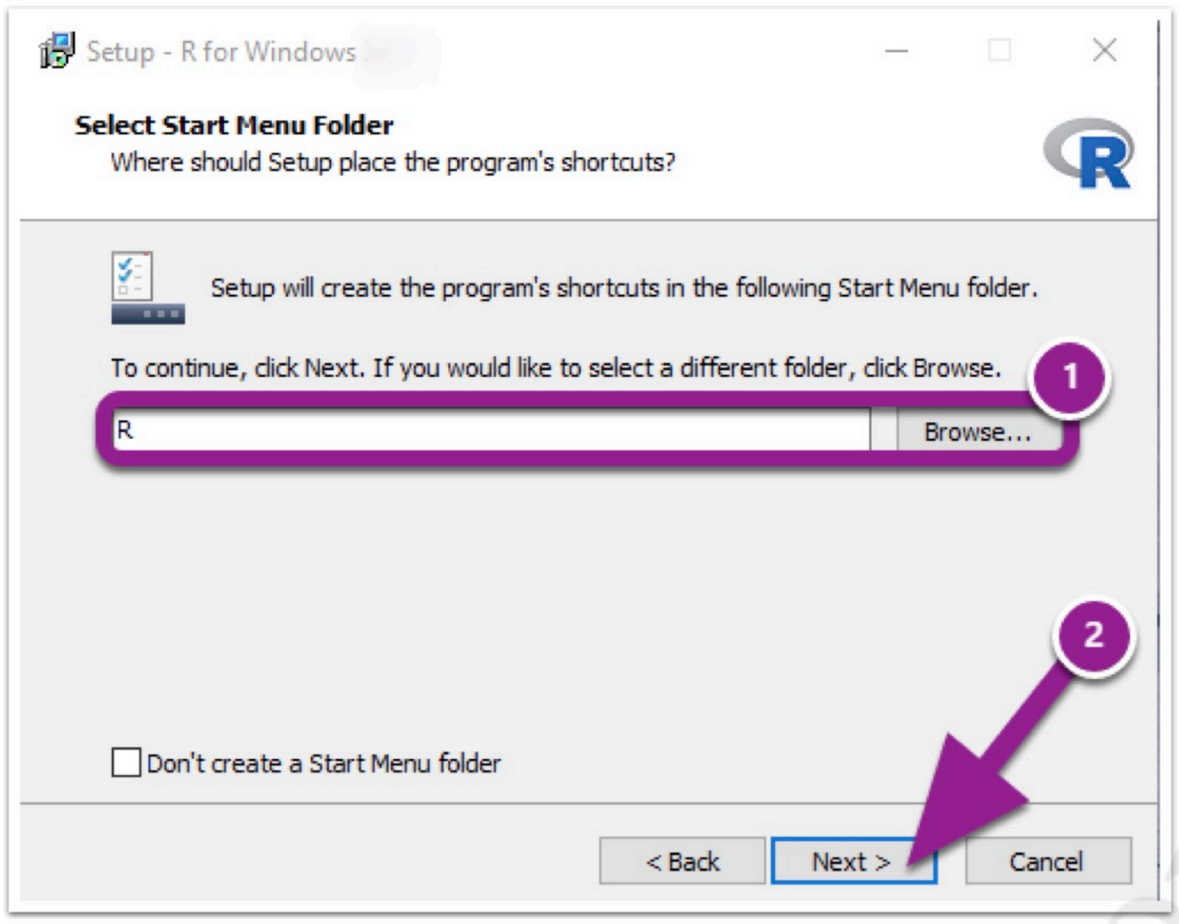

Figure 11: Select start menu folder

12. You can then select additional shortcuts if you would like. Click Next.

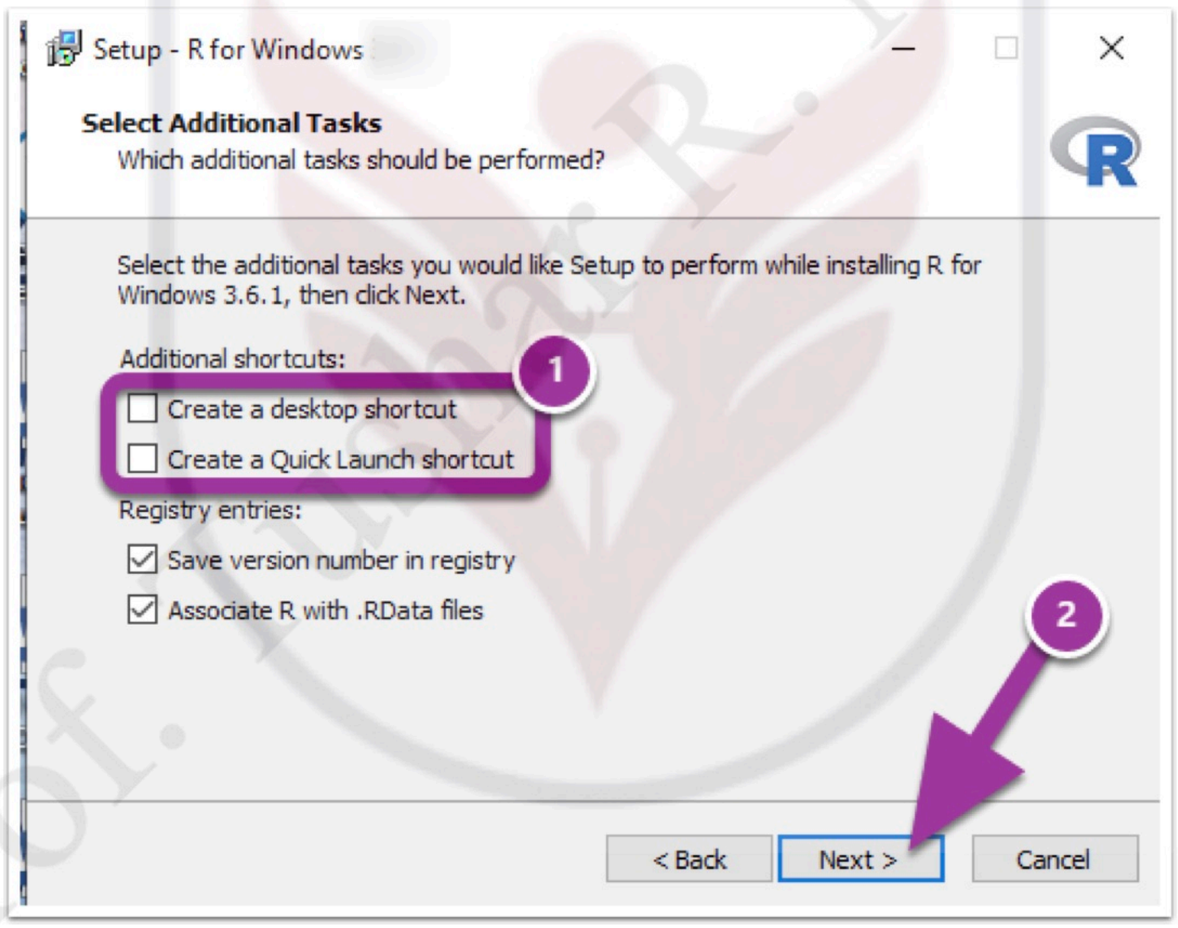

Figure 12: Select Additional Tasks

13. Click Finish.

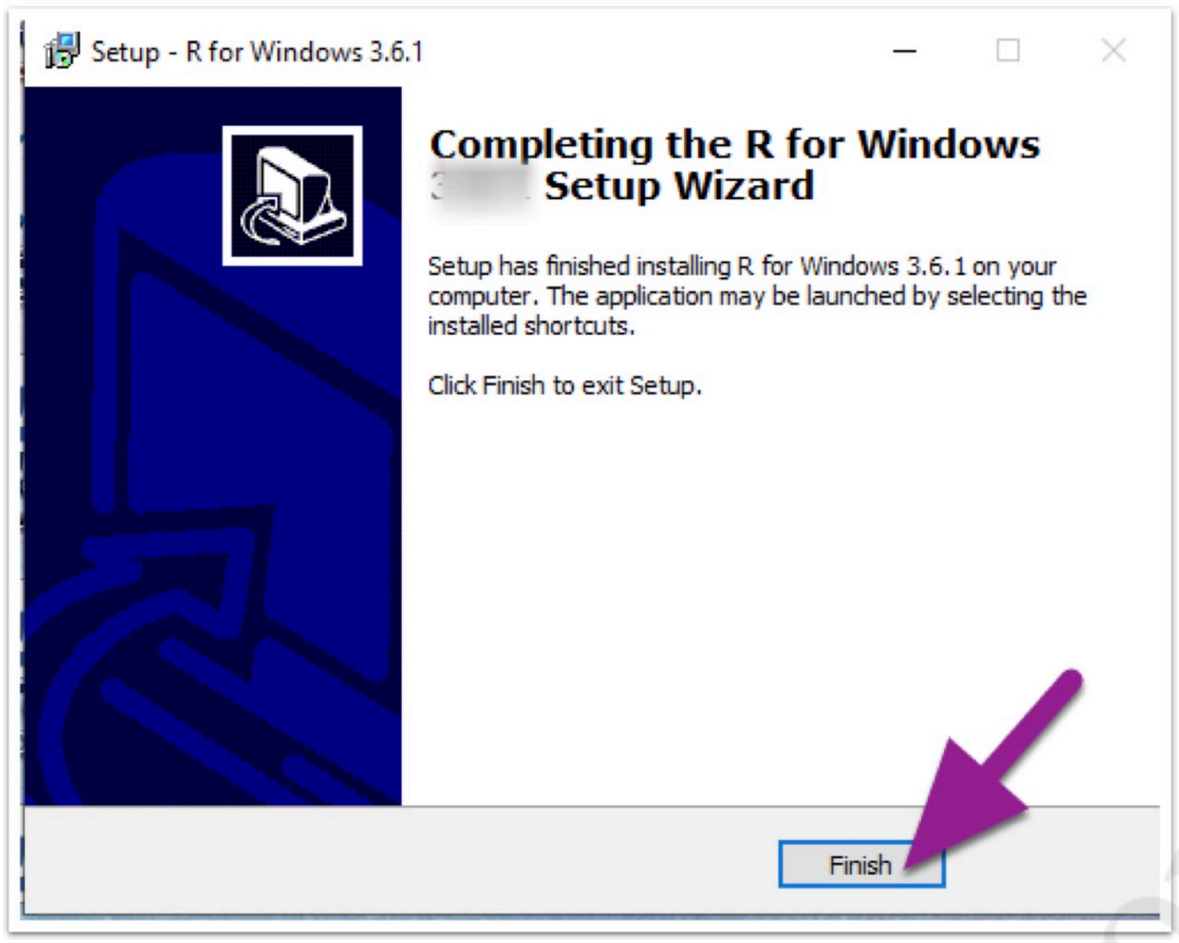

Figure 13: Completing R Setup

14. Go to the RStudio download page: `https://posit.co/download/rstudio-desktop/`.

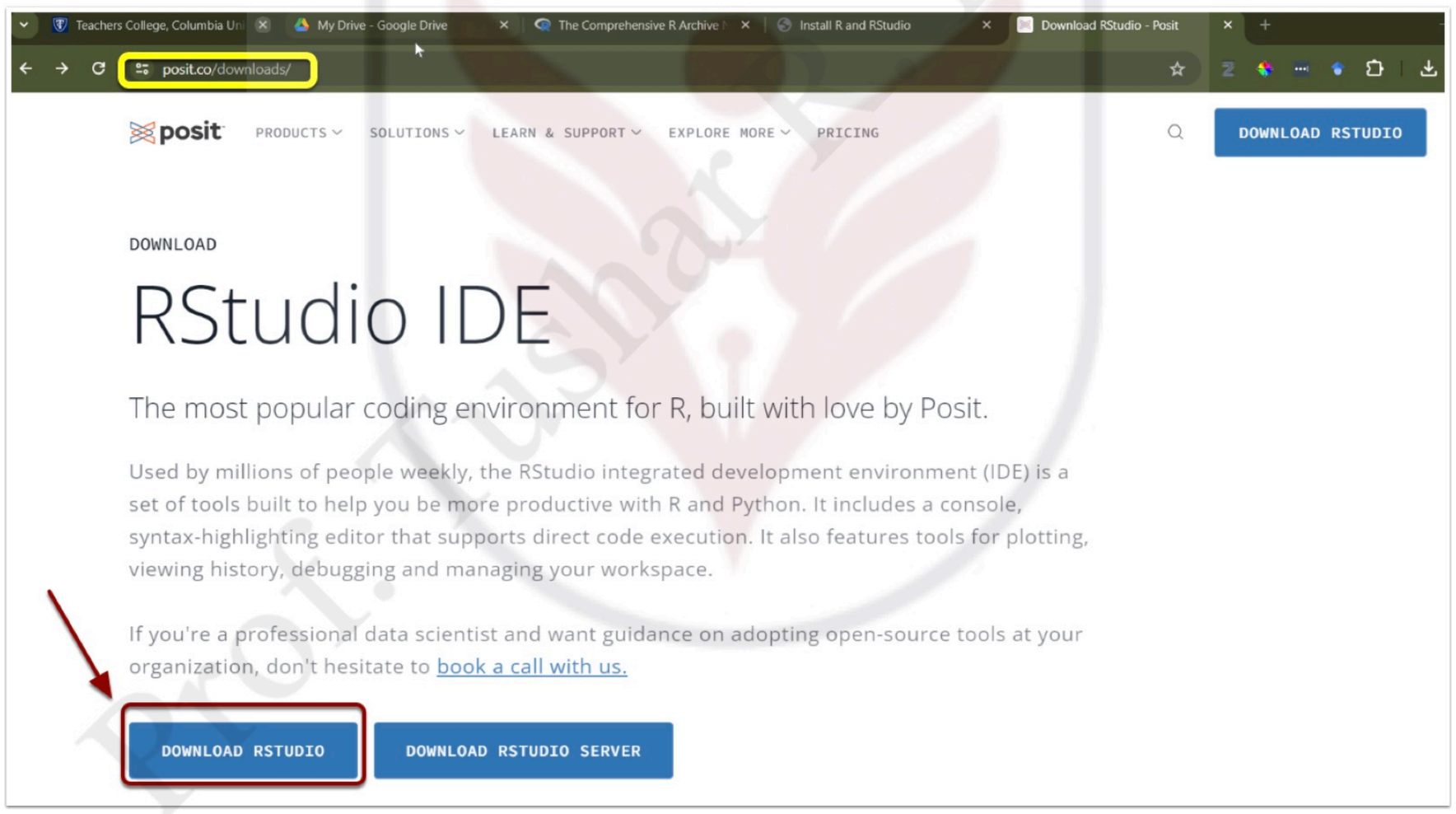

Figure 14: RStudio Desktop download page

15. Download the RStudio installer for Windows.

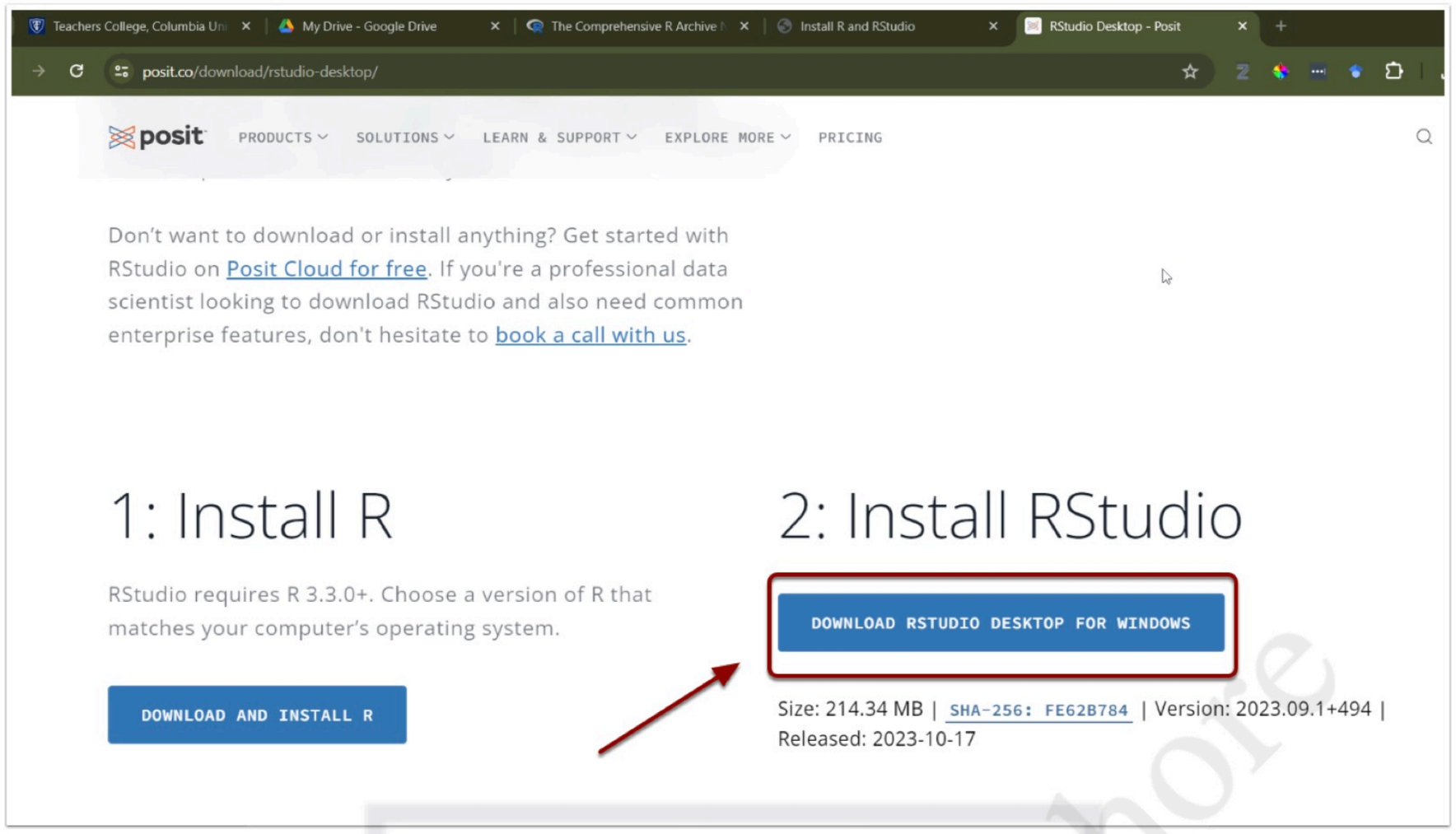

Figure 15: RStudio installer download

16. Run the RStudio installer and follow the setup steps.

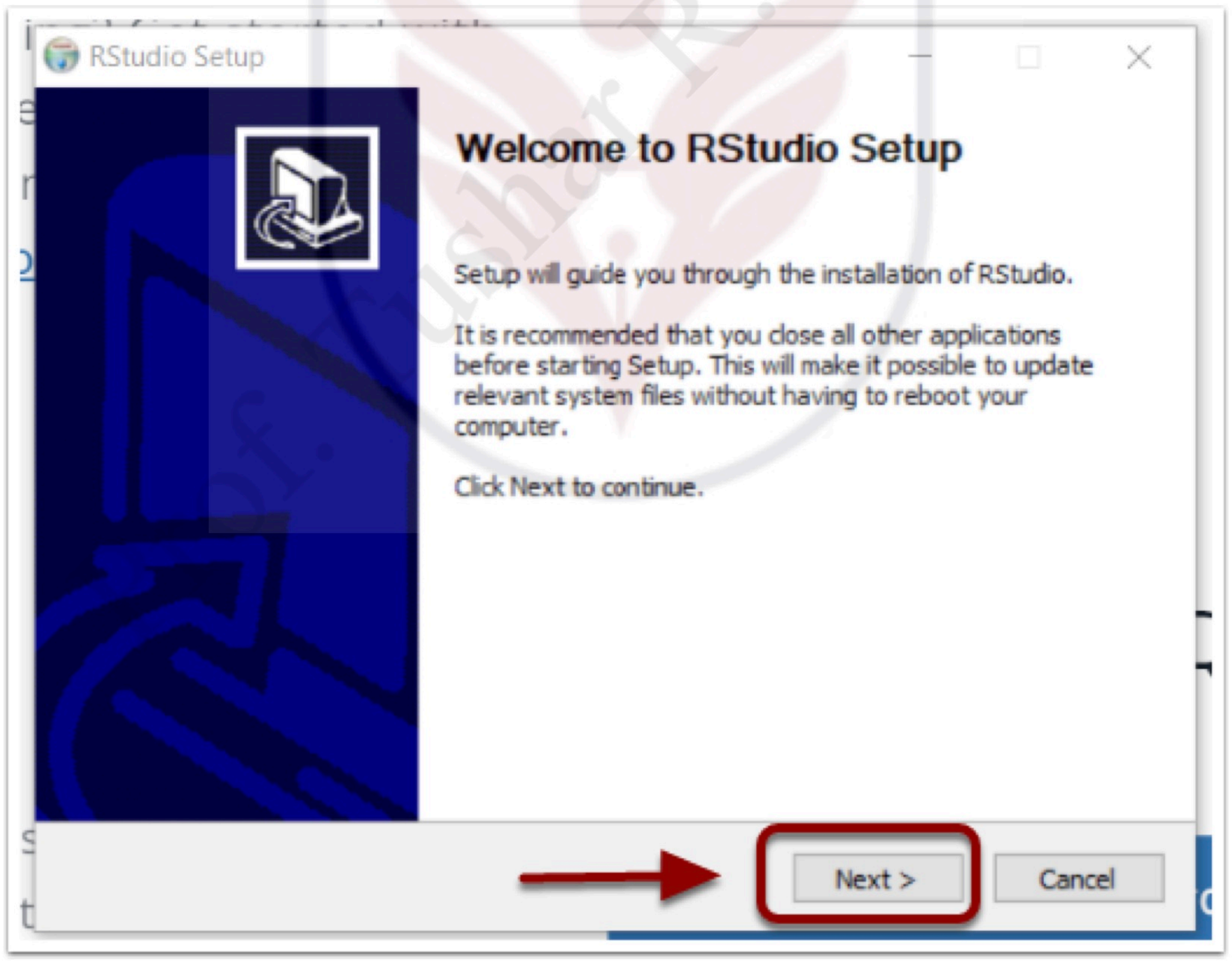

Figure 16: RStudio installation wizard

17. Open RStudio. You should see the main interface with four panes.

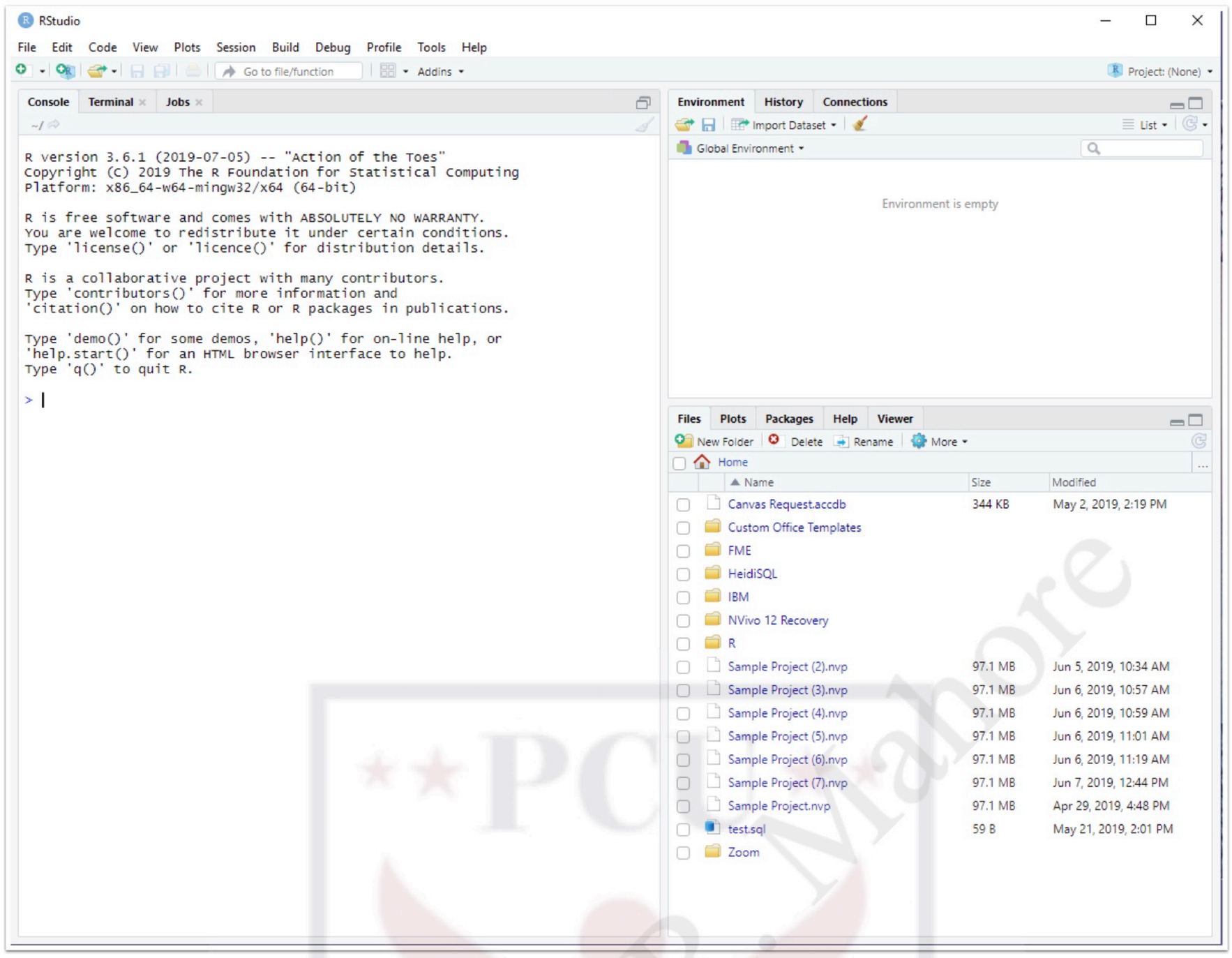

Figure 17: RStudio interface after installation

RStudio Environment

When you open RStudio, you see four main panes:

- Source/Script Editor (Top-Left): Write and save R scripts (.R files).
- Console (Bottom-Left): Directly execute commands.
- Environment/History (Top-Right): Shows created variables, datasets, and command history.
- Files/Plots/Packages/Help/Viewer (Bottom-Right): View plots, install packages, and access documentation.

Basic Components of R

- **Data Types:**
  - Numeric: Decimal numbers (e.g., 3.14, 100).
  - Integer: Whole numbers (e.g., 10L).
  - Character/String: Text data (e.g., "Hello").
  - Logical/Boolean: TRUE or FALSE.
  - Factor: Categorical values (e.g., "Male", "Female").
- **Operators in R:**

  - Arithmetic: $+, -, *, /, \hat{}$
  - Relational: $<, >, <=, >=, ==, !=$
  - Logical: $\&, |, !$

- **Variable Assignment:**
  - $x < -10$ (preferred in R)
  - $y = 20$

- **Functions:**
  - Predefined R functions like mean(), sum(), summary().

- **Inbuilt Datasets in R:**
  - iris – Flower measurements (classification).
  - mtcars – Automobile design and performance.
  - airquality – Daily air quality measurements in New York.
  - Titanic – Survival dataset.

Program

```
# Print a message
print("Hello, Welcome to R Programming")

# Assign variables
x <- 25
y <- 5

# Arithmetic operations
sum_val <- x + y
diff_val <- x - y
prod_val <- x * y
div_val <- x / y

sum_val
diff_val
prod_val
div_val

# Relational operations
greater_check <- x > y
equal_check <- x == y

# Load and explore iris dataset
data("iris")       # load dataset
head(iris)         # first 6 rows
str(iris)          # structure of dataset
summary(iris)      # descriptive statistics
```

Output

```
> # Print a message
> print("Hello, Welcome to R Programming")
[1] "Hello, Welcome to R Programming"
>
> # Assign variables
> x <- 25
> y <- 5
>
> # Arithmetic operations
> sum_val <- x + y
> diff_val <- x - y
> prod_val <- x * y
> div_val <- x / y
>
> sum_val
[1] 30
> diff_val
[1] 20
> prod_val
[1] 125
> div_val
[1] 5
>
> # Relational operations
> greater_check <- x > y
> equal_check <- x == y
>
> # Load and explore iris dataset
> data("iris")      # load dataset
> head(iris)        # first 6 rows
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> str(iris)         # structure of dataset
'data.frame':    150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> summary(iris)     # descriptive statistics
 Sepal.Length    Sepal.Width     Petal.Length
Min.   :4.300   Min.   :2.000   Min.   :1.000
1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600
Median :5.800   Median :3.000   Median :4.350
Mean   :5.843   Mean   :3.057   Mean   :3.758
3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100
Max.   :7.900   Max.   :4.400   Max.   :6.900
 Petal.Width          Species
Min.   :0.100   setosa    :50
1st Qu.:0.300   versicolor:50
Median :1.300   virginica :50
```

```
 Mean   :1.199
 3rd Qu.:1.800
 Max.   :2.500
>
```

Exercise Questions

- Display the first 15 rows of the `mtcars` dataset.
- Find the maximum and minimum value of `Sepal.Length` from `iris`.
- Calculate the mean of the variable `mpg` in `mtcars`.
- Display the structure of the `airquality` dataset.
- Check whether the number 100 is greater than 50 in R.

Viva Questions

- What is R? How is it different from Python?
- Explain the different panes in RStudio.
- What is the purpose of `<-` in R?
- Define factor data type with an example.
- What are some inbuilt datasets available in R?
- How do you check the structure of a dataset in R?
- Differentiate between numeric and integer data types in R.

Conclusion

This practical introduced students to the R programming environment and RStudio IDE. By performing basic operations and exploring an inbuilt dataset, students gained confidence in navigating the interface, using variables, and running simple commands — laying the foundation for future analytical tasks.

Practical 2: Importing and Exploring a Dataset in R

Aim

To import a dataset into R, inspect its structure, summarize variable types and distributions, and identify missing values.

Objective

- Learn how to import datasets into R.
- Explore dataset structure using R functions.
- Summarize variables and their distributions.
- Detect and handle missing values.
- Practice with inbuilt datasets (iris, Titanic, mtcars).

Software Required

- R (version 4.0 or above)
- RStudio IDE
- Packages: `datasets`, `dplyr`, `ggplot2`

Theory

Importing Data in R

Data can be imported from multiple sources:

- CSV files: `read.csv("file.csv")`
- Excel files: using `readxl` package
- Text files: `read.table("file.txt")`
- Databases: using `DBI`, `RMySQL`
- Inbuilt datasets: e.g., `iris`, `mtcars`, `Titanic`

Exploring Data in R

Once imported, data must be checked and summarized:

- `head(dataset)`, `tail(dataset)` – view first/last rows
- `str(dataset)` – check structure
- `summary(dataset)` – descriptive statistics
- `colnames(dataset)` – view column names

Identifying Missing Values

- `is.na(dataset)` – identifies missing values
- `sum(is.na(dataset))` – total count of missing values
- `colSums(is.na(dataset))` – missing values per column

Program

```r
# Load required package
library(dplyr)

# 1. Import inbuilt dataset: iris
data("iris")
head(iris)        # first 6 rows
str(iris)         # structure
summary(iris)     # summary statistics

# 2. Import another dataset: mtcars
data("mtcars")
head(mtcars)
str(mtcars)

# 3. Titanic dataset
data("Titanic")
Titanic           # view dataset

# 4. Check for missing values
sum(is.na(iris))          # total missing values
colSums(is.na(mtcars))    # missing values per column

# 5. Summarize distribution by group
iris %>%
  group_by(Species) %>%
  summarise(
    Avg_Sepal_Length = mean(Sepal.Length),
    Avg_Petal_Length = mean(Petal.Length)
  )
```

Output

```
> # Load required package
> 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
>
> # 1. Import inbuilt dataset: iris
> data("iris")
```

```
> head(iris)        # first 6 rows
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> str(iris)         # structure
'data.frame':         150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> summary(iris)     # summary statistics
 Sepal.Length    Sepal.Width     Petal.Length
Min.   :4.300   Min.   :2.000   Min.   :1.000
1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600
Median :5.800   Median :3.000   Median :4.350
Mean   :5.843   Mean   :3.057   Mean   :3.758
3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100
Max.   :7.900   Max.   :4.400   Max.   :6.900
 Petal.Width          Species
Min.   :0.100   setosa    :50
1st Qu.:0.300   versicolor:50
Median :1.300   virginica :50
Mean   :1.199
3rd Qu.:1.800
 Max.   :2.500
>
> # 2. Import another dataset: mtcars
> data("mtcars")
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0
                  gear carb
Mazda RX4            4    4
Mazda RX4 Wag        4    4
Datsun 710           4    1
Hornet 4 Drive       3    1
Hornet Sportabout    3    2
Valiant              3    1
> str(mtcars)
'data.frame':         32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
```

```
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
>
> # 3. Titanic dataset
> data("Titanic")
> Titanic           # view dataset
, , Age = Child, Survived = No

      Sex
Class  Male Female
  1st     0      0
  2nd     0      0
  3rd    35     17
  Crew    0      0

, , Age = Adult, Survived = No

      Sex
Class  Male Female
  1st   118      4
  2nd   154     13
  3rd   387     89
  Crew  670      3

, , Age = Child, Survived = Yes

      Sex
Class  Male Female
  1st     5      1
  2nd    11     13
  3rd    13     14
  Crew    0      0

, , Age = Adult, Survived = Yes

      Sex
Class  Male Female
  1st    57    140
  2nd    14     80
  3rd    75     76
  Crew  192     20

>
> # 4. Check for missing values
> sum(is.na(iris))          # total missing values
[1] 0
> colSums(is.na(mtcars))    # missing values per column
 mpg  cyl disp   hp drat   wt qsec   vs   am gear carb
   0    0    0    0    0    0    0    0    0    0    0
>
> # 5. Summarize distribution by group
> iris %>%
+   group_by(Species) %>%
+   summarise(
+     Avg_Sepal_Length = mean(Sepal.Length),
+     Avg_Petal_Length = mean(Petal.Length)
+   )
```

```
# A tibble: 3 × 3
  Species    Avg_Sepal_Length Avg_Petal_Length
  <fct>                 <dbl>             <dbl>
1 setosa                 5.01              1.46
2 versicolor             5.94              4.26
3 virginica              6.59              5.55
>
```

Exercise Questions

- Import the `airquality` dataset and check its structure using `str()`.
- Find the number of missing values in `airquality`.
- Use `summary()` on the `mtcars` dataset and interpret the output.
- Find the mean horsepower (`hp`) in the `mtcars` dataset.
- Group the `iris` dataset by species and calculate the average `Sepal.Width`.

Viva Questions

- What are the different ways to import data into R?
- What is the difference between `head()` and `tail()` functions?
- How do you identify missing values in a dataset?
- What is the purpose of the `str()` function?
- Why is it important to summarize data before analysis?
- Which inbuilt datasets in R can be used for classification and regression?
- Explain the difference between `is.na()` and `na.omit()`.

Conclusion

Through dataset import and exploration, students learned to inspect data structures, summarize variables, and detect missing values. This practical highlighted the importance of understanding the dataset before analysis, ensuring accurate insights and informed decision-making.

Practical 3: Data Cleaning and Preprocessing in R

Aim

To clean and preprocess a dataset by removing duplicates, handling missing values, and standardizing data formats using R functions.

Objective

- Understand the importance of data cleaning in the data science workflow.
- Detect and remove duplicate records.
- Handle missing values using different strategies.
- Standardize data formats for consistency.
- Practice cleaning techniques on inbuilt datasets.

Software Required

- R (version 4.0 or above)
- RStudio IDE
- Packages: `dplyr`, `tidyr`

Theory

Importance of Data Cleaning

Real-world datasets often contain errors, inconsistencies, or missing values. Data cleaning ensures the dataset is accurate, consistent, and ready for analysis.

Common Data Issues

- Duplicates: Same record appearing multiple times.
- Missing values: Absent or `NA` entries in datasets.
- Inconsistent formats: Different date, text, or categorical formats.
- Outliers: Extreme values deviating from the rest of the data.

Techniques in R

- Detecting Duplicates:
  - `duplicated(dataset)` – returns TRUE for duplicate rows.
  - `distinct(dataset)` – removes duplicates.
- Handling Missing Values:
  - `is.na(dataset)` – identify missing values.

- `sum(is.na(dataset))` – total number of missing values.
- `na.omit(dataset)` – remove missing values.
- Replace NA with mean/median/mode.

- Standardizing Formats:
  - `tolower()`, `toupper()` – convert text cases.
  - `as.factor()` – convert categorical values to factors.
  - `as.numeric()`, `as.character()` – convert data types.

Program

```
library(dplyr)

# Load dataset
data("airquality")
head(airquality)

# 1. Check for missing values
sum(is.na(airquality))              # total NA count
colSums(is.na(airquality))          # missing values per column

# 2. Handle missing values
# Replace missing Ozone values with mean
airquality$Ozone[is.na(airquality$Ozone)] <- mean(airquality$Ozone, na.rm = TRUE)

# Replace missing Solar.R values with median
airquality$Solar.R[is.na(airquality$Solar.R)] <- median(airquality$Solar.R, na.rm =
↪  TRUE)

# 3. Remove duplicates (example with iris dataset)
data("iris")
iris_with_duplicates <- rbind(iris, iris[1:5, ])   # add duplicates artificially
nrow(iris_with_duplicates)  # before removing
iris_clean <- distinct(iris_with_duplicates)       # remove duplicates
nrow(iris_clean)            # after removing

# 4. Standardize formats
iris_clean$Species <- tolower(as.character(iris_clean$Species))
iris_clean$Species <- as.factor(iris_clean$Species)

# View cleaned dataset
head(iris_clean)
```

Output

```
> library(dplyr)
>
> # Load dataset
> data("airquality")
> head(airquality)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
```

```
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
>
> # 1. Check for missing values
> sum(is.na(airquality))              # total NA count
[1] 44
> colSums(is.na(airquality))          # missing values per column
  Ozone Solar.R    Wind    Temp   Month     Day
     37       7       0       0       0       0
>
> # 2. Handle missing values
> # Replace missing Ozone values with mean
> airquality$Ozone[is.na(airquality$Ozone)] <- mean(airquality$Ozone, na.rm = TRUE)
>
> # Replace missing Solar.R values with median
> airquality$Solar.R[is.na(airquality$Solar.R)] <- median(airquality$Solar.R, na.rm =
↪  TRUE)
>
> # 3. Remove duplicates (example with iris dataset)
> data("iris")
> iris_with_duplicates <- rbind(iris, iris[1:5, ])   # add duplicates artificially
> nrow(iris_with_duplicates)  # before removing
[1] 155
> iris_clean <- distinct(iris_with_duplicates)       # remove duplicates
> nrow(iris_clean)            # after removing
[1] 149
>
> # 4. Standardize formats
> iris_clean$Species <- tolower(as.character(iris_clean$Species))
> iris_clean$Species <- as.factor(iris_clean$Species)
>
> # View cleaned dataset
> head(iris_clean)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
>
```

Exercise Questions

- Count how many missing values are present in the `airquality` dataset.
- Replace missing values in `Ozone` with the median instead of mean.
- Create a duplicate dataset from `mtcars` and remove the duplicates.
- Convert all car names in `mtcars` row names to uppercase.
- Convert the `Species` column in `iris` dataset into numeric codes (1, 2, 3).

Viva Questions

- Why is data cleaning important in data analysis?
- What are different strategies for handling missing values?
- How do you detect duplicate records in R?
- Explain the difference between `na.omit()` and `is.na()`.
- Why should categorical data be converted into factors?
- What are outliers and why do we need to handle them?
- Give an example of data standardization in R.

Conclusion

Students practiced handling missing values, removing duplicates, and standardizing data formats. The exercise emphasized that data quality is crucial for reliable analysis and demonstrated key preprocessing techniques to prepare raw data for further exploration.

Practical 4: Descriptive Statistics and Basic Visualizations in R

Aim

To generate and interpret descriptive statistics (mean, median, mode, standard deviation) and create basic visualizations (histograms, scatterplots, boxplots) using R.

Objective

- Understand measures of central tendency and dispersion.
- Apply descriptive statistical functions in R.
- Visualize data distributions using histograms.
- Explore relationships between variables using scatterplots.
- Interpret statistical results with plots for decision-making.

Software Required

- R (version 4.0 or above)
- RStudio IDE
- Packages: `dplyr`, `ggplot2`, `modeest`

Theory

Descriptive Statistics

- **Mean:** Average of values.
- **Median:** Middle value when data is sorted.
- **Mode:** Most frequently occurring value.
- **Standard Deviation:** Measure of spread around the mean.
- **Range:** Difference between maximum and minimum.

Data Visualization

- **Histogram:** Shows frequency distribution of a numeric variable.
- **Scatterplot:** Displays relationship between two numeric variables.
- **Boxplot:** Summarizes data using quartiles and highlights outliers.

Program

```
library(dplyr)
library(ggplot2)
library(modeest)    # for mode

# Load dataset
data("iris")

# Descriptive statistics
mean(iris$Sepal.Length)           # mean
median(iris$Sepal.Length)         # median
mlv(iris$Sepal.Length, method="mfv")  # mode
sd(iris$Sepal.Length)             # standard deviation
range(iris$Sepal.Length)          # min and max
summary(iris$Sepal.Length)        # summary

# Histogram
hist(iris$Sepal.Length,
     main="Histogram of Sepal Length",
     xlab="Sepal Length", col="lightblue", border="black")

# Scatterplot
plot(iris$Sepal.Length, iris$Petal.Length,
     main="Scatterplot of Sepal vs Petal Length",
     xlab="Sepal Length", ylab="Petal Length",
     col="blue", pch=19)

# Boxplot
boxplot(Sepal.Length ~ Species, data=iris,
        main="Boxplot of Sepal Length by Species",
        xlab="Species", ylab="Sepal Length",
        col=c("lightgreen","lightblue","pink"))
```

Output

```
> library(dplyr)
> library(ggplot2)
> library(modeest)    # for mode
> # Load dataset
> data("iris")
> # Descriptive statistics
> mean(iris$Sepal.Length)           # mean
[1] 5.843333
> median(iris$Sepal.Length)         # median
[1] 5.8
> mlv(iris$Sepal.Length, method="mfv")  # mode
[1] 5
> sd(iris$Sepal.Length)             # standard deviation
[1] 0.8280661
> range(iris$Sepal.Length)          # min and max
[1] 4.3 7.9
> summary(iris$Sepal.Length)        # summary
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.300   5.100   5.800   5.843   6.400   7.900
> # Histogram
> hist(iris$Sepal.Length,
+      main="Histogram of Sepal Length",
```

```
+      xlab="Sepal Length", col="lightblue", border="black")
> # Scatterplot
> plot(iris$Sepal.Length, iris$Petal.Length,
+      main="Scatterplot of Sepal vs Petal Length",
+      xlab="Sepal Length", ylab="Petal Length",
+      col="blue", pch=19)
> # Boxplot
> boxplot(Sepal.Length ~ Species, data=iris,
+         main="Boxplot of Sepal Length by Species",
+         xlab="Species", ylab="Sepal Length",
+         col=c("lightgreen","lightblue","pink"))
>
```

Graphical Output

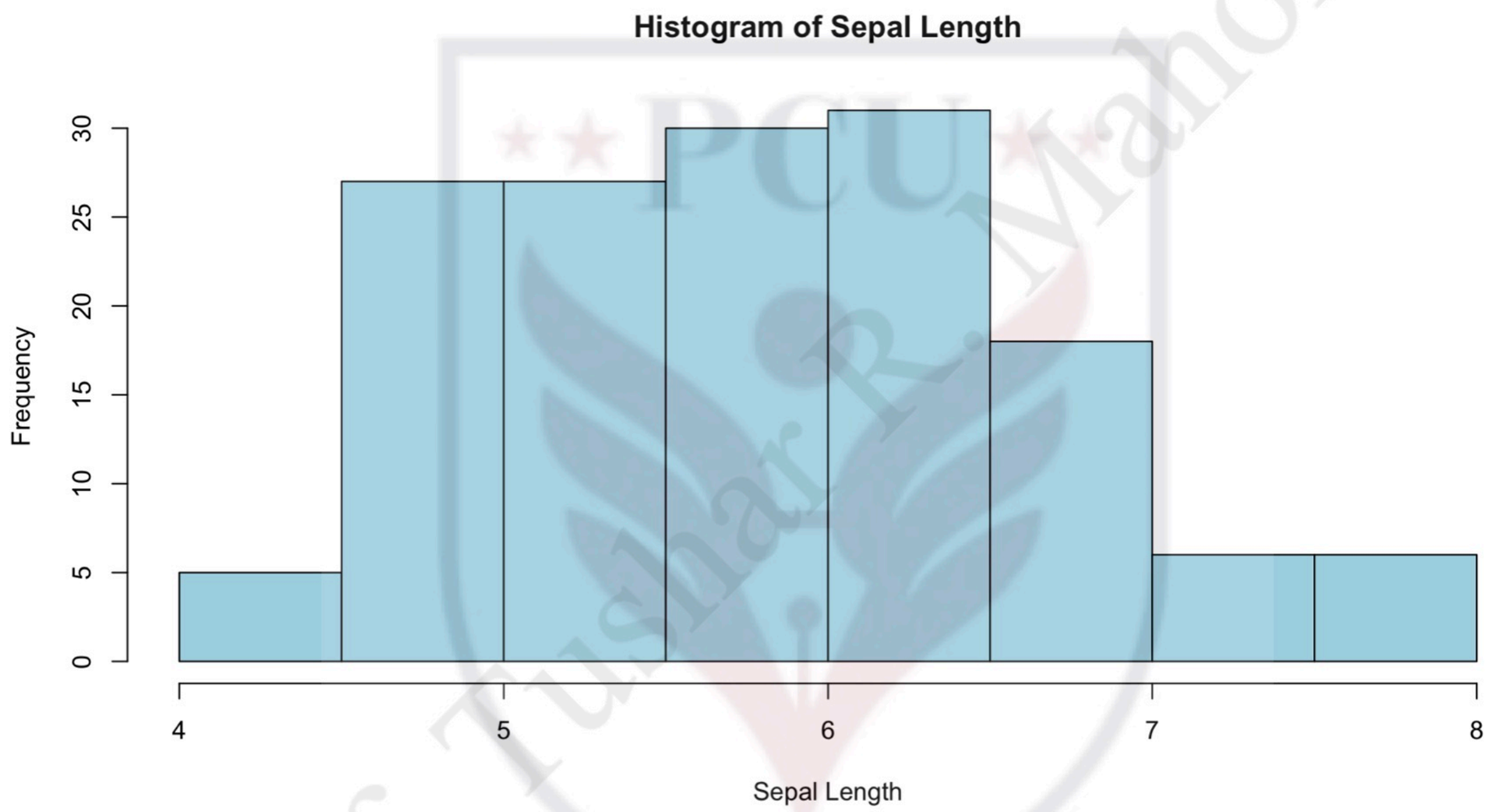

Figure 18: Histogram of Sepal Length from the Iris dataset

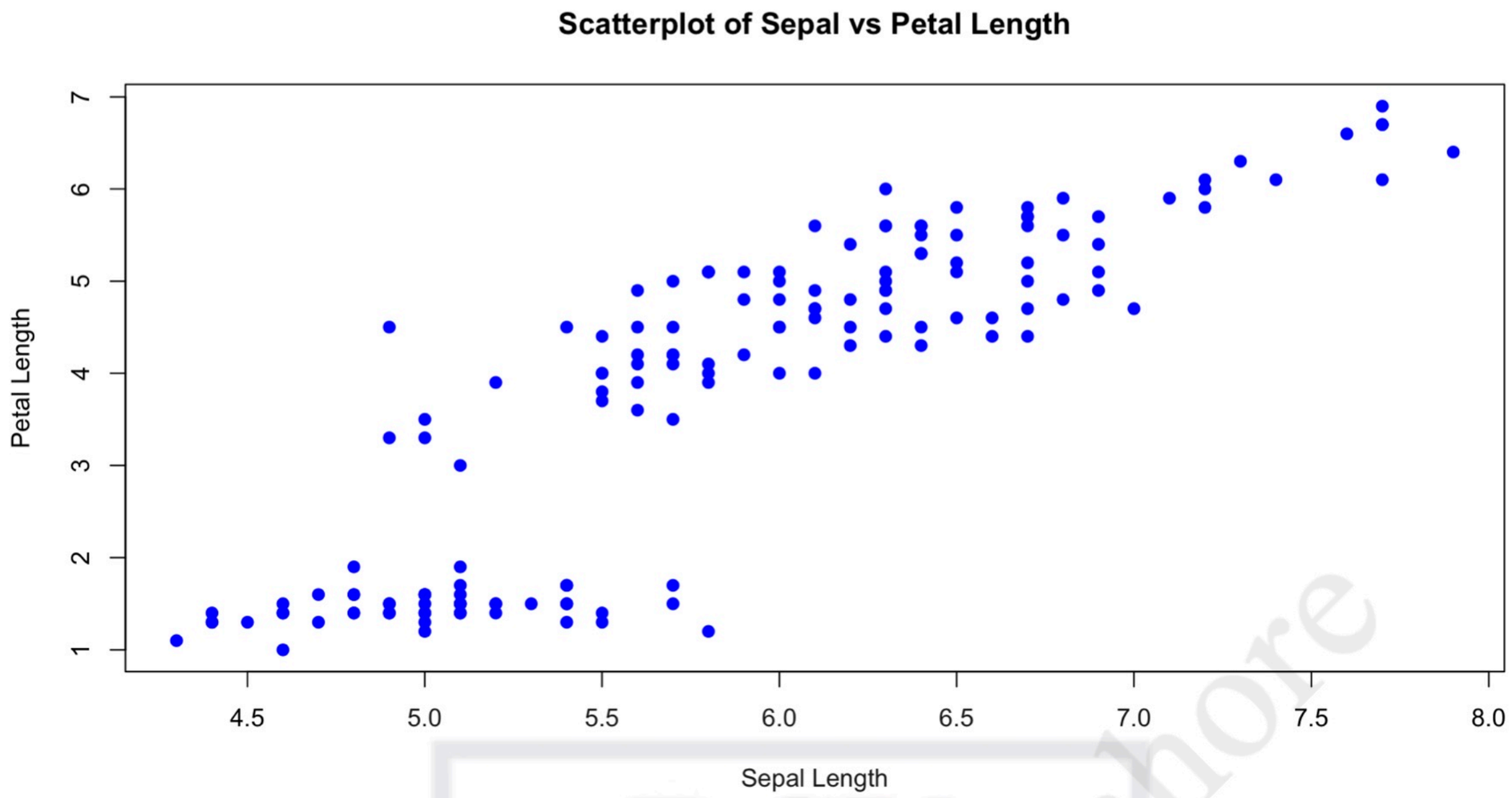

Figure 19: Scatterplot of Sepal Length vs Petal Length

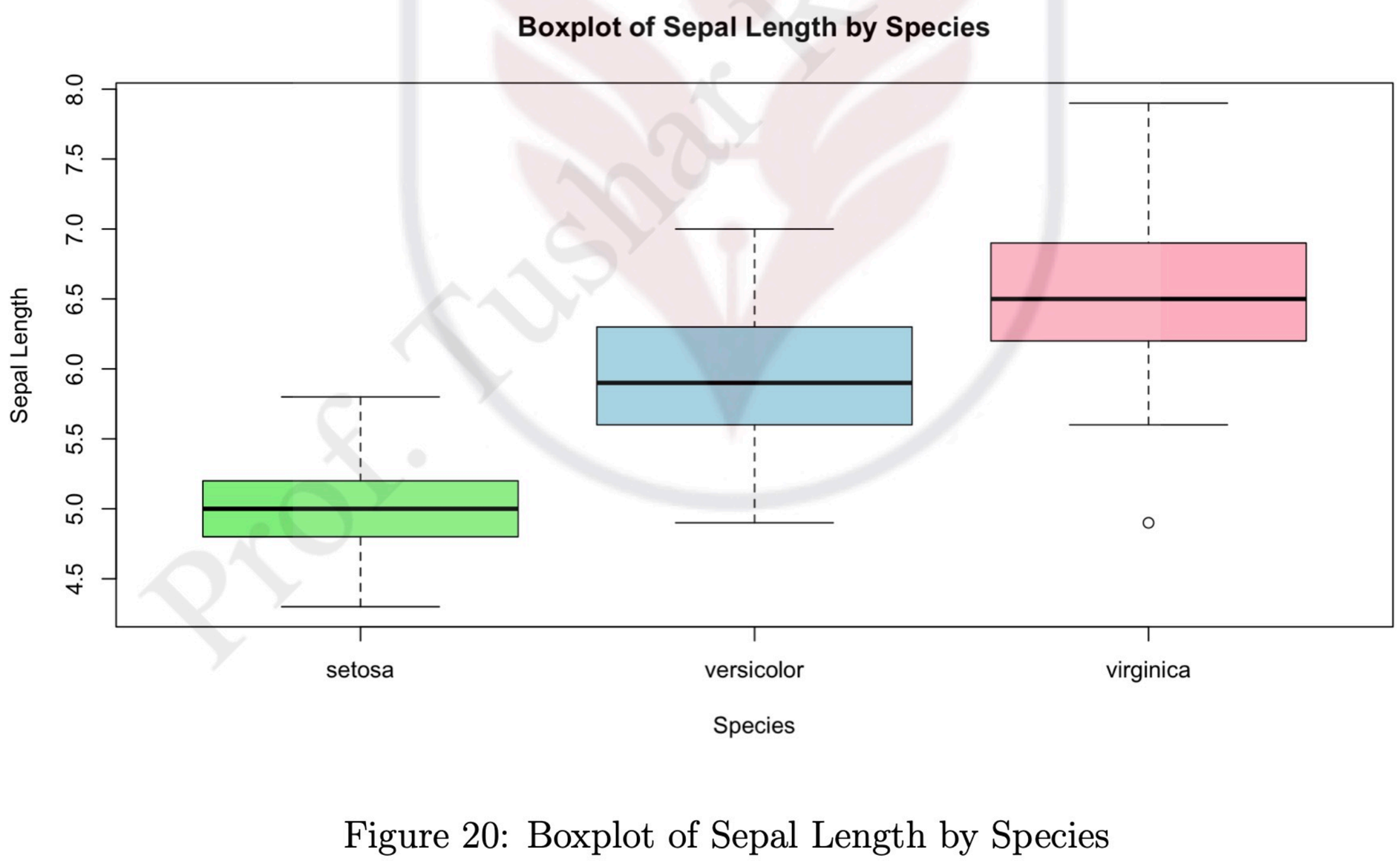

Figure 20: Boxplot of Sepal Length by Species

Exercise Questions

- Calculate the mean, median, mode, and standard deviation of `Petal.Width` from `iris`.
- Draw a histogram of `mpg` from the `mtcars` dataset.
- Create a scatterplot of `hp` vs `mpg` from `mtcars`.

- Generate a boxplot for `Ozone` in the `airquality` dataset.
- Find the variance of `Sepal.Width` in the `iris` dataset.

Viva Questions

- Define mean, median, and mode with examples.
- What is the significance of standard deviation?
- Differentiate between histogram and bar chart.
- What insights can you derive from a scatterplot?
- What does a boxplot show about data distribution?
- Why do we need both central tendency and dispersion measures?

Conclusion

This session reinforced the role of descriptive statistics in summarizing data and introduced graphical methods like histograms, scatterplots, and boxplots. Students learned how statistical measures and visualizations together provide a clear picture of dataset characteristics.

Practical 5: Variable Transformation and Feature Engineering in R

Aim

To perform variable transformation (binning, encoding, scaling) and create new features or derived columns for deeper analysis using R.

Objective

- Understand the role of feature engineering in data analysis and machine learning.
- Apply binning/discretization to continuous variables.
- Convert categorical variables into numeric using encoding techniques.
- Apply scaling/normalization to bring variables to the same scale.
- Create derived columns for improved analysis and model performance.

Software Required

- R (version 4.0 or above)
- RStudio IDE
- Packages: `dplyr`, `caret`, `ggplot2`

Theory

Feature Engineering

Feature engineering is the process of transforming raw data into meaningful features that improve the performance of models.

Variable Transformation

- **Binning:** Converts continuous data into categories (e.g., Age → Child, Adult, Senior).
- **Encoding:**
  - Label Encoding – assigns numeric codes to categories.
  - One-Hot Encoding – creates binary columns for each category.
- **Scaling:**
  - Normalization (Min-Max Scaling): $X' = \frac{X - X_{min}}{X_{max} - X_{min}}$
  - Standardization (Z-score): $Z = \frac{X - \mu}{\sigma}$

Feature Creation

New features can be created by combining or transforming existing ones. Examples: BMI = Weight / Height^2, Sales per Customer = Total Sales / Number of Customers.

Program

```
library(dplyr)
library(caret)

# Load dataset
data("mtcars")
head(mtcars)

# 1. Binning: Categorize mpg into Low, Medium, High
mtcars$mpg_category <- cut(mtcars$mpg,
                           breaks = c(-Inf, 15, 25, Inf),
                           labels = c("Low", "Medium", "High"))
table(mtcars$mpg_category)

# 2. Encoding: Convert Species to numeric codes (iris dataset)
data("iris")
iris$Species_code <- as.numeric(as.factor(iris$Species))
head(iris[, c("Species", "Species_code")])

# 3. Normalization: Scale wt (weight) column
mtcars$wt_normalized <- (mtcars$wt - min(mtcars$wt)) /
                        (max(mtcars$wt) - min(mtcars$wt))
head(mtcars$wt_normalized)

# 4. Standardization: Z-score for hp (horsepower)
mtcars$hp_zscore <- scale(mtcars$hp)
head(mtcars$hp_zscore)

# 5. Feature Creation: Power-to-Weight Ratio
mtcars$power_to_weight <- mtcars$hp / mtcars$wt
head(mtcars$power_to_weight)
```

Output

```
> library(dplyr)
> library(caret)
Loading required package: lattice
> # Load dataset
> data("mtcars")
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0
                  gear carb
Mazda RX4            4    4
Mazda RX4 Wag        4    4
Datsun 710           4    1
```

```
Hornet 4 Drive       3    1
Hornet Sportabout    3    2
Valiant              3    1
> # 1. Binning: Categorize mpg into Low, Medium, High
> mtcars$mpg_category <- cut(mtcars$mpg,
+                            breaks = c(-Inf, 15, 25, Inf),
+                            labels = c("Low", "Medium", "High"))
> table(mtcars$mpg_category)

   Low Medium   High
     6     20      6
> # 2. Encoding: Convert Species to numeric codes (iris dataset)
> data("iris")
> iris$Species_code <- as.numeric(as.factor(iris$Species))
> head(iris[, c("Species", "Species_code")])
  Species Species_code
1  setosa            1
2  setosa            1
3  setosa            1
4  setosa            1
5  setosa            1
6  setosa            1
> # 3. Normalization: Scale wt (weight) column
> mtcars$wt_normalized <- (mtcars$wt - min(mtcars$wt)) /
+   (max(mtcars$wt) - min(mtcars$wt))
> head(mtcars$wt_normalized)
[1] 0.2830478 0.3482485 0.2063411 0.4351828 0.4927129
[6] 0.4978266
> # 4. Standardization: Z-score for hp (horsepower)
> mtcars$hp_zscore <- scale(mtcars$hp)
> head(mtcars$hp_zscore)
            [,1]
[1,] -0.5350928
[2,] -0.5350928
[3,] -0.7830405
[4,] -0.5350928
[5,]  0.4129422
[6,] -0.6080186
> # 5. Feature Creation: Power-to-Weight Ratio
> mtcars$power_to_weight <- mtcars$hp / mtcars$wt
> head(mtcars$power_to_weight)
[1] 41.98473 38.26087 40.08621 34.21462 50.87209 30.34682
```

Exercise Questions

- Perform binning on `Sepal.Length` from `iris` into categories: Short, Medium, Long.
- Convert `Species` column in `iris` into dummy variables (One-Hot Encoding).
- Normalize the `mpg` column in `mtcars` dataset.
- Standardize the `Sepal.Width` column in `iris`.
- Create a new feature in `mtcars`: efficiency = mpg / hp.

Viva Questions

- What is feature engineering? Why is it important?
- Differentiate between normalization and standardization.
- What is binning, and when is it useful?
- What is the difference between label encoding and one-hot encoding?
- Why is scaling required in machine learning?
- Give an example of a derived feature in a dataset.
- How can poor feature engineering affect a machine learning model?

Conclusion

Students applied transformations such as binning, encoding, and scaling, along with creating derived features. This practical underlined how thoughtful feature engineering improves model performance and analytical depth, making data more meaningful for interpretation.

Practical 6: Exploratory Data Analysis (EDA) in R

Aim

To conduct Exploratory Data Analysis (EDA) to uncover patterns, correlations, and trends in the data, and visualize the findings using R plotting libraries.

Objective

- Understand the concept and importance of EDA in data science.
- Use summary statistics to understand data distribution.
- Identify relationships between variables using correlation and plots.
- Visualize patterns and trends using histograms, scatterplots, and pair plots.
- Interpret findings for better decision-making.

Software Required

- R (version 4.0 or above)
- RStudio IDE
- Packages: `dplyr`, `ggplot2`, `GGally`

Theory

Exploratory Data Analysis

EDA is the initial step in data analysis where we summarize the dataset, identify patterns, detect anomalies, and check assumptions before modeling.

Key Steps in EDA

- Data Inspection – check structure, missing values, and types.
- Univariate Analysis – analyze a single variable (e.g., histogram).
- Bivariate Analysis – study relationships between two variables (e.g., scatterplot, correlation).
- Multivariate Analysis – interactions among multiple variables (e.g., pair plots).

Correlation Coefficient

$$r = \frac{\text{Cov}(X, Y)}{\sigma_X \cdot \sigma_Y}$$

- $r = +1$: strong positive relationship
- $r = -1$: strong negative relationship
- $r = 0$: no linear relationship

Program

```r
library(dplyr)
library(ggplot2)
library(GGally)
library(ggcorrplot)

# Load dataset
data("iris")
head(iris)

# 1. Summary statistics
summary(iris)

# 2. Histogram of Sepal.Length
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram(bins = 20, fill = "lightblue", color = "black") +
  labs(title = "Distribution of Sepal Length", x = "Sepal Length", y = "Frequency")

# 3. Scatterplot Sepal.Length vs Petal.Length
ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, color = Species)) +
  geom_point(size = 3) +
  labs(title = "Sepal Length vs Petal Length")

# 4. Boxplot of Sepal.Width by Species
ggplot(iris, aes(x = Species, y = Sepal.Width, fill = Species)) +
  geom_boxplot() +
  labs(title = "Boxplot of Sepal Width by Species")

# 5. Correlation matrix
corr_matrix = cor(iris[, 1:4])
corr_matrix
ggcorrplot(corr_matrix, lab=TRUE, title="Correlation Matrix Heatmap")

# 6. Pair plot
ggpairs(iris[, 1:4])
```

Output

```
> library(dplyr)
> library(ggplot2)
> library(GGally)
> library(ggcorrplot)
> # Load dataset
> data("iris")
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
> # 1. Summary statistics
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length
 Min.   :4.300   Min.   :2.000   Min.   :1.000
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600
```

```
Median :5.800   Median :3.000   Median :4.350
Mean   :5.843   Mean   :3.057   Mean   :3.758
3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100
Max.   :7.900   Max.   :4.400   Max.   :6.900
 Petal.Width          Species
Min.   :0.100   setosa    :50
1st Qu.:0.300   versicolor:50
Median :1.300   virginica :50
Mean   :1.199
3rd Qu.:1.800
Max.   :2.500
> # 2. Histogram of Sepal.Length
> ggplot(iris, aes(x = Sepal.Length)) +
+   geom_histogram(bins = 20, fill = "lightblue", color = "black") +
+   labs(title = "Distribution of Sepal Length", x = "Sepal Length", y = "Frequency")
> # 3. Scatterplot Sepal.Length vs Petal.Length
> ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, color = Species)) +
+   geom_point(size = 3) +
+   labs(title = "Sepal Length vs Petal Length")
> # 4. Boxplot of Sepal.Width by Species
> ggplot(iris, aes(x = Species, y = Sepal.Width, fill = Species)) +
+   geom_boxplot() +
+   labs(title = "Boxplot of Sepal Width by Species")
> # 5. Correlation matrix
> corr_matrix = cor(iris[, 1:4])
> corr_matrix
             Sepal.Length Sepal.Width Petal.Length
Sepal.Length    1.0000000  -0.1175698    0.8717538
Sepal.Width    -0.1175698   1.0000000   -0.4284401
Petal.Length    0.8717538  -0.4284401    1.0000000
Petal.Width     0.8179411  -0.3661259    0.9628654
             Petal.Width
Sepal.Length   0.8179411
Sepal.Width   -0.3661259
Petal.Length   0.9628654
Petal.Width    1.0000000
> ggcorrplot(corr_matrix, lab=TRUE, title="Correlation Matrix Heatmap")
> # 6. Pair plot
> ggpairs(iris[, 1:4])

>
```

Graphical Output

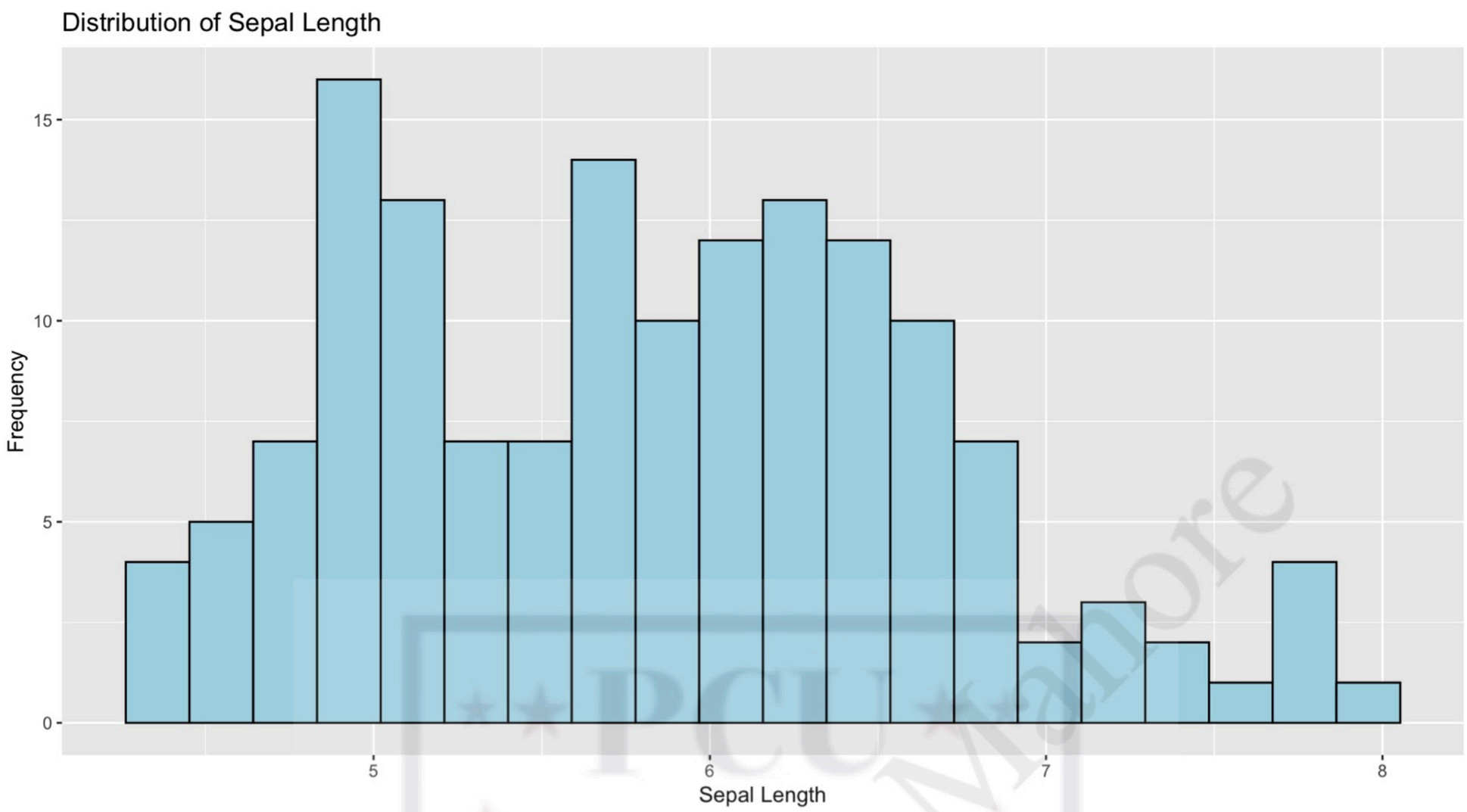

Figure 21: Histogram of Sepal Length from the Iris dataset

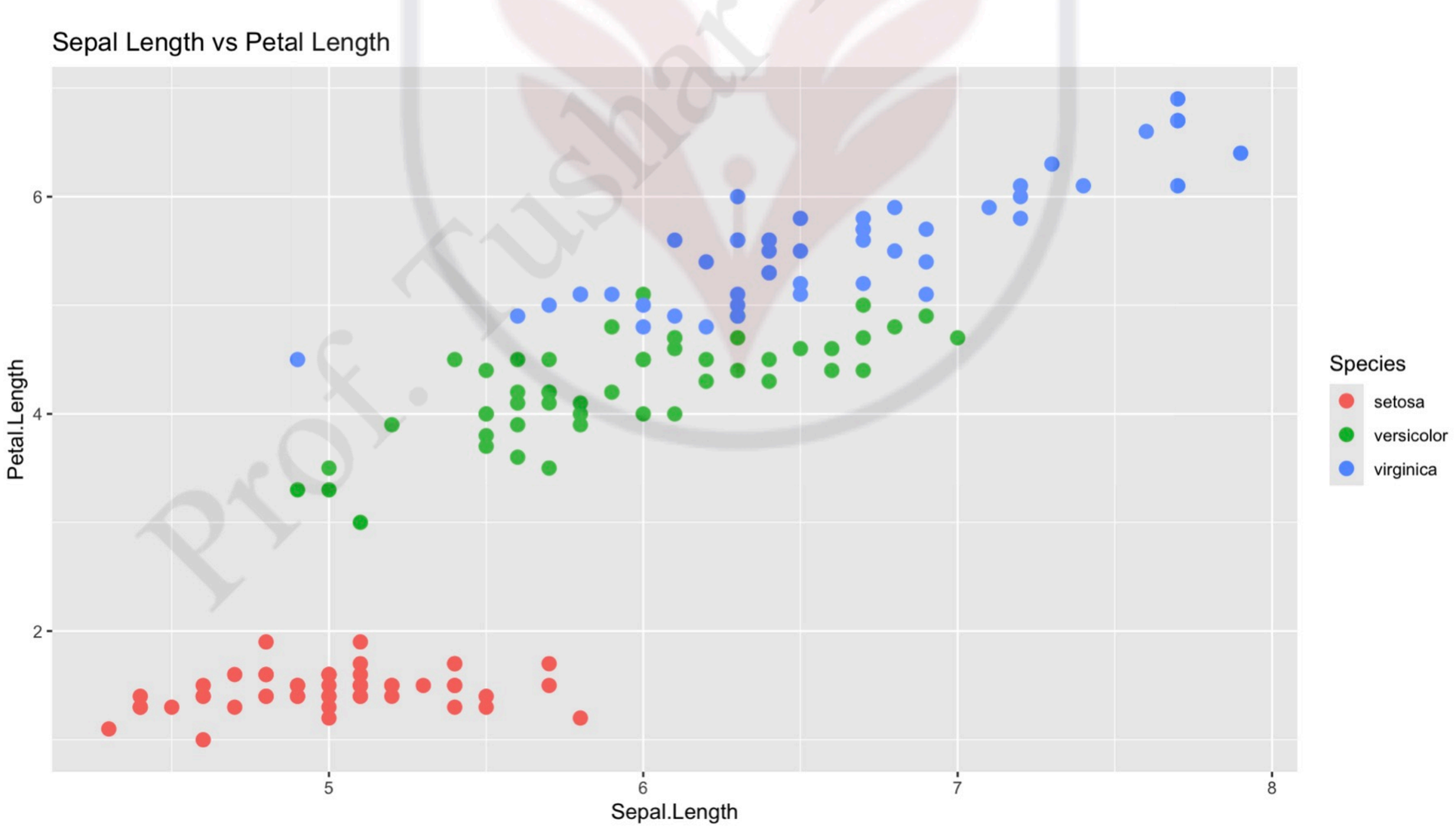

Figure 22: Scatterplot of Sepal Length vs Petal Length (colored by Species)

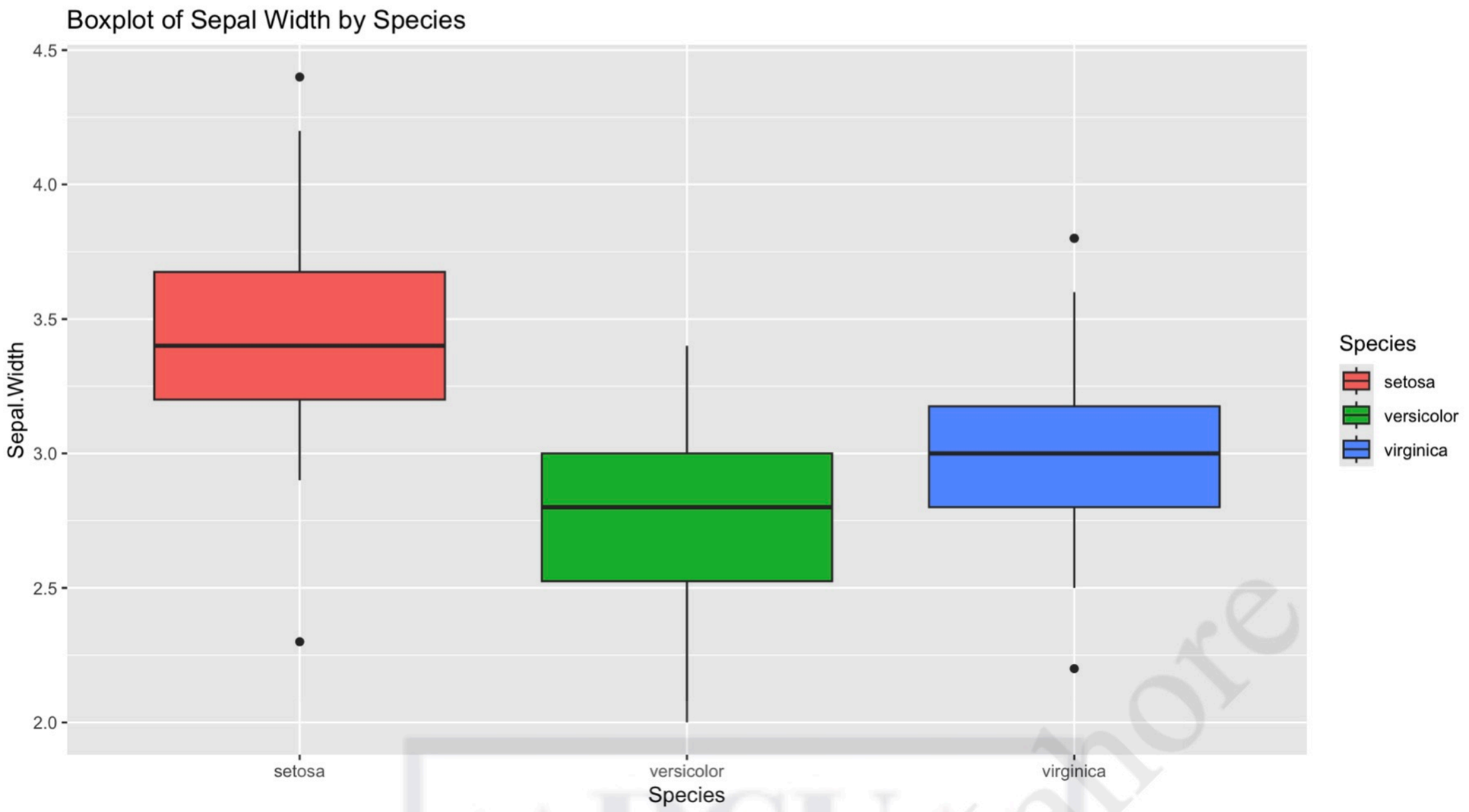

Figure 23: Boxplot of Sepal Width by Species

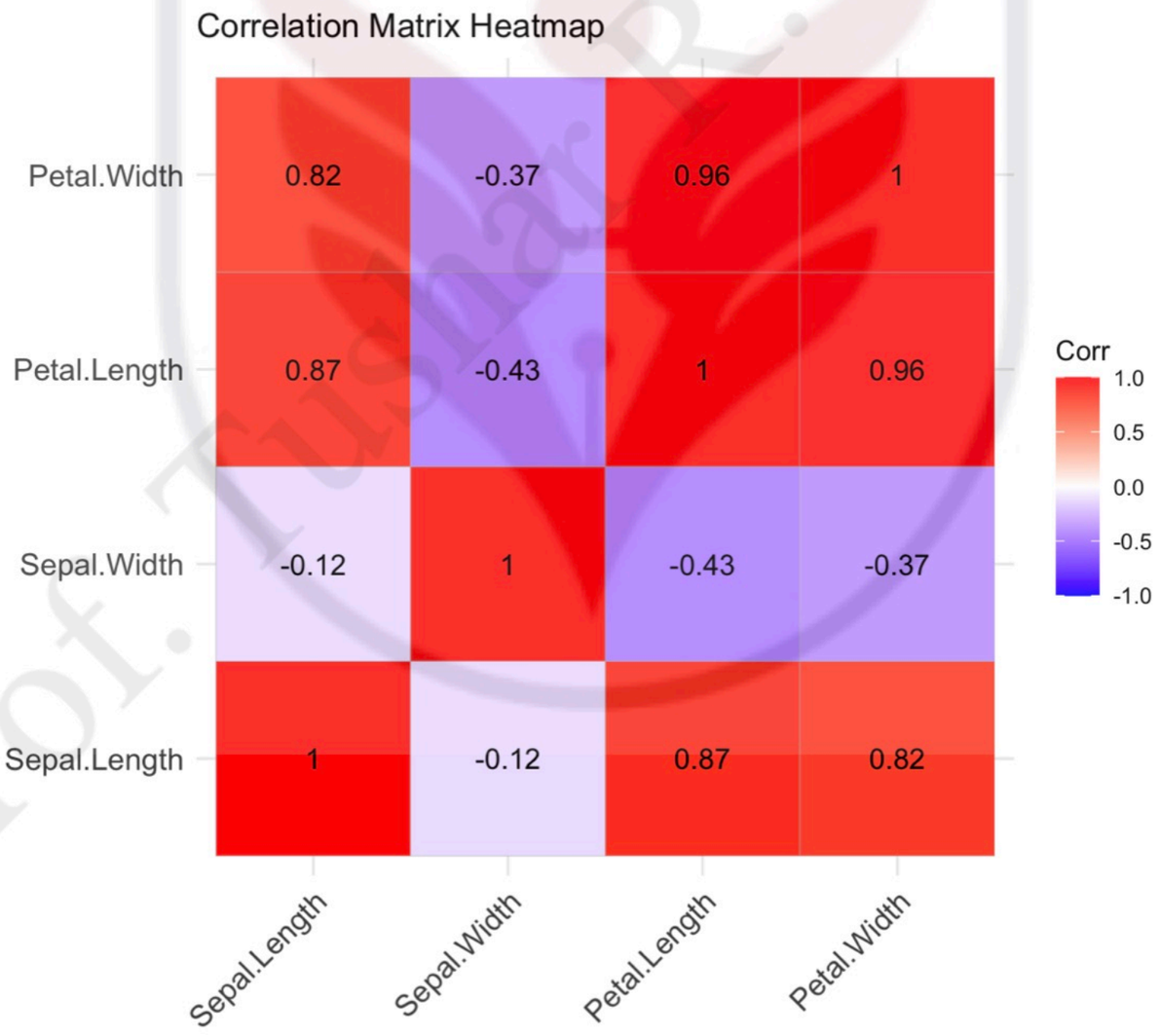

Figure 24: Correlation Matrix Heatmap for Iris numeric variables

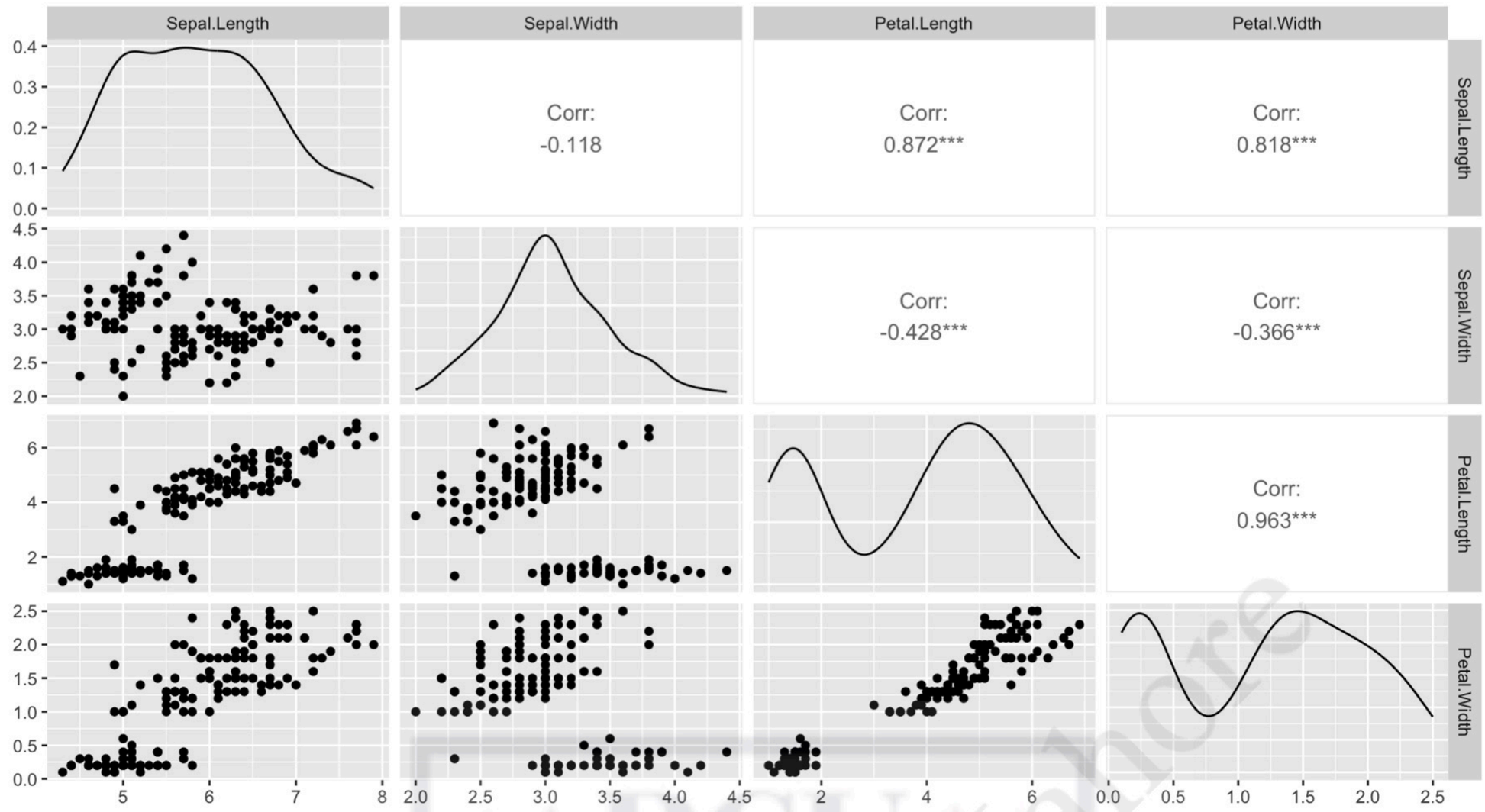

Figure 25: Pair Plot (Scatterplot Matrix) of Iris Dataset

Exercise Questions

- Create a histogram of `mpg` from the `mtcars` dataset.
- Generate a scatterplot of `hp` vs `wt` from `mtcars`.
- Draw a boxplot of `Ozone` values from the `airquality` dataset.
- Find the correlation between `Sepal.Length` and `Sepal.Width` in `iris`.
- Create a pair plot of the first four columns of `mtcars`.

Viva Questions

- What is the purpose of EDA in data science?
- Differentiate between univariate, bivariate, and multivariate analysis.
- What does a correlation coefficient of +0.85 indicate?
- How do histograms and boxplots differ?
- Why is it important to visualize data before applying models?
- Explain why pair plots are useful.

Conclusion

By performing EDA, students uncovered hidden patterns, trends, and correlations in the dataset. Visualization and summary techniques enabled them to derive initial insights, reinforcing the critical role of EDA as the bridge between raw data and formal modeling.

Practical 7: Statistical Tests in R

Aim

To apply statistical tests (t-test, ANOVA, correlation) in R to answer data-driven questions.

Objective

- Understand the need for statistical hypothesis testing.
- Apply t-test to compare means of groups.
- Apply ANOVA to test differences among multiple groups.
- Compute correlation coefficients to assess relationships between variables.
- Interpret statistical test results in the context of datasets.

Software Required

- R (version 4.0 or above)
- RStudio IDE
- Packages: `dplyr`, `stats`

Theory

Hypothesis Testing

- **Null Hypothesis (H_0):** No effect or no difference.
- **Alternative Hypothesis (H_1):** There is an effect or a difference.
- Decision Rule: If $p < 0.05$, reject H_0; otherwise, fail to reject H_0.

t-test

Used to compare means.

- One-sample t-test: compares mean of sample with a given value.
- Two-sample t-test: compares means of two independent groups.
- Paired t-test: compares means of paired/related groups.

ANOVA (Analysis of Variance)

- Tests whether there are statistically significant differences between means of three or more groups.
- Based on the F-statistic.

Correlation

- Measures strength and direction of relationship between two variables.
- Pearson's correlation coefficient r ranges from -1 to +1.

Program

```
library(dplyr)

# Load iris dataset
data("iris")

# 1. t-test: Compare Sepal.Length of setosa and versicolor
t_test_result <- t.test(Sepal.Length ~ Species,
                        data = iris %>% filter(Species %in% c("setosa",
                        ↪  "versicolor")))
t_test_result

# 2. ANOVA: Compare Sepal.Length across all species
anova_model <- aov(Sepal.Length ~ Species, data = iris)
summary(anova_model)

# 3. Correlation: Sepal.Length and Petal.Length
correlation <- cor(iris$Sepal.Length, iris$Petal.Length)
correlation

# 4. Correlation test with significance
cor_test <- cor.test(iris$Sepal.Length, iris$Petal.Length)
cor_test
```

Output

```
> library(dplyr)
> # Load iris dataset
> data("iris")
> # 1. t-test: Compare Sepal.Length of setosa and versicolor
> t_test_result <- t.test(Sepal.Length ~ Species,
+                         data = iris %>% filter(Species %in% c("setosa",
↪  "versicolor")))
> t_test_result

        Welch Two Sample t-test

data:  Sepal.Length by Species
t = -10.521, df = 86.538, p-value < 2.2e-16
alternative hypothesis: true difference in means between group setosa and group
↪  versicolor is not equal to 0
95 percent confidence interval:
 -1.1057074 -0.7542926
sample estimates:
    mean in group setosa mean in group versicolor
                   5.006                    5.936

> # 2. ANOVA: Compare Sepal.Length across all species
> anova_model <- aov(Sepal.Length ~ Species, data = iris)
> summary(anova_model)
```

```
             Df Sum Sq Mean Sq F value Pr(>F)
Species       2  63.21  31.606   119.3 <2e-16 ***
Residuals   147  38.96   0.265
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> # 3. Correlation: Sepal.Length and Petal.Length
> correlation <- cor(iris$Sepal.Length, iris$Petal.Length)
> correlation
[1] 0.8717538
> # 4. Correlation test with significance
> cor_test <- cor.test(iris$Sepal.Length, iris$Petal.Length)
> cor_test

        Pearson's product-moment correlation

data:  iris$Sepal.Length and iris$Petal.Length
t = 21.646, df = 148, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.8270363 0.9055080
sample estimates:
      cor
0.8717538

>
```

Exercise Questions

- Perform a one-sample t-test to check if the mean `mpg` in `mtcars` differs from 20.
- Compare `Petal.Width` across species in `iris` using ANOVA.
- Find the correlation between `mpg` and `hp` in `mtcars`.
- Perform a paired t-test on `Sepal.Length` and `Sepal.Width` in `iris`.
- Check if `Ozone` and `Temp` in `airquality` are significantly correlated.

Viva Questions

- What is hypothesis testing and why is it important?
- Differentiate between one-sample, two-sample, and paired t-tests.
- What does a p-value represent?
- What is the purpose of ANOVA?
- How do you interpret a correlation coefficient of -0.85?
- When would you use a t-test instead of ANOVA?
- Can correlation imply causation? Why or why not?

Conclusion

This practical demonstrated hypothesis testing to make data-driven decisions. By applying t-tests, ANOVA, and correlation, students learned how to statistically validate relationships and differences, enhancing their ability to draw reliable inferences from data.

Practical 8: Regression Analysis in R

Aim

To build and evaluate a regression model (linear regression) to predict a numeric variable and interpret the results using R.

Objective

- Understand the concept of regression analysis.
- Apply simple linear regression for predicting a variable using one predictor.
- Apply multiple linear regression for prediction using multiple predictors.
- Evaluate model performance using R^2, residuals, and plots.
- Interpret regression coefficients for decision-making.

Software Required

- R (version 4.0 or above)
- RStudio IDE
- Packages: `dplyr`, `ggplot2`, `stats`

Theory

Regression Analysis

Regression is a supervised learning technique used to model the relationship between a dependent variable (Y) and one or more independent variables (X).

Simple Linear Regression

$$Y = \beta_0 + \beta_1 X + \epsilon$$

- β_0: Intercept
- β_1: Slope (effect of X on Y)
- ϵ: Error term

Multiple Linear Regression

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \epsilon$$

Model Evaluation

- R^2: Proportion of variance explained by the model.
- Residuals: Difference between observed and predicted values.
- p-values of coefficients: Significance of predictors.

Program

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

# Load dataset
data("mtcars")
head(mtcars)

# 1. Simple Linear Regression: mpg predicted by wt
model_simple <- lm(mpg ~ wt, data = mtcars)
summary(model_simple)

# 2. Plot regression line
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point(color = "blue") +
  geom_smooth(method = "lm", se = TRUE, color = "red") +
  labs(title = "Simple Linear Regression: MPG vs Weight",
       x = "Weight (1000 lbs)", y = "Miles per Gallon")

# 3. Multiple Linear Regression: mpg predicted by wt and hp
model_multiple <- lm(mpg ~ wt + hp, data = mtcars)
summary(model_multiple)

# 4. Residual diagnostics
plot(model_multiple, which = 1)    # Residuals vs Fitted
plot(model_multiple, which = 2)    # Q-Q Plot
```

Output

```
> library(dplyr)
> library(ggplot2)
> # Load dataset
> data("mtcars")
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0
                  gear carb
Mazda RX4            4    4
Mazda RX4 Wag        4    4
Datsun 710           4    1
Hornet 4 Drive       3    1
Hornet Sportabout    3    2
Valiant              3    1
> # 1. Simple Linear Regression: mpg predicted by wt
> model_simple <- lm(mpg ~ wt, data = mtcars)
> summary(model_simple)

Call:
lm(formula = mpg ~ wt, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
```

```
-4.5432 -2.3647 -0.1252  1.4096  6.8727

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
wt           -5.3445     0.5591  -9.559 1.29e-10 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.046 on 30 degrees of freedom
Multiple R-squared:  0.7528,        Adjusted R-squared:  0.7446
F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10

> # 2. Plot regression line
> ggplot(mtcars, aes(x = wt, y = mpg)) +
+   geom_point(color = "blue") +
+   geom_smooth(method = "lm", se = TRUE, color = "red") +
+   labs(title = "Simple Linear Regression: MPG vs Weight",
+        x = "Weight (1000 lbs)", y = "Miles per Gallon")
`geom_smooth()` using formula = 'y ~ x'
> # 3. Multiple Linear Regression: mpg predicted by wt and hp
> model_multiple <- lm(mpg ~ wt + hp, data = mtcars)
> summary(model_multiple)

Call:
lm(formula = mpg ~ wt + hp, data = mtcars)

Residuals:
   Min     1Q Median     3Q    Max
-3.941 -1.600 -0.182  1.050  5.854

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 37.22727    1.59879  23.285  < 2e-16 ***
wt          -3.87783    0.63273  -6.129 1.12e-06 ***
hp          -0.03177    0.00903  -3.519  0.00145 **
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.593 on 29 degrees of freedom
Multiple R-squared:  0.8268,        Adjusted R-squared:  0.8148
F-statistic: 69.21 on 2 and 29 DF,  p-value: 9.109e-12

> # 4. Residual diagnostics
> plot(model_multiple, which = 1)   # Residuals vs Fitted
> plot(model_multiple, which = 2)   # Q-Q Plot
>
```

Graphical Output

Figure 26: Regression Line Plot: MPG vs Weight with fitted regression line

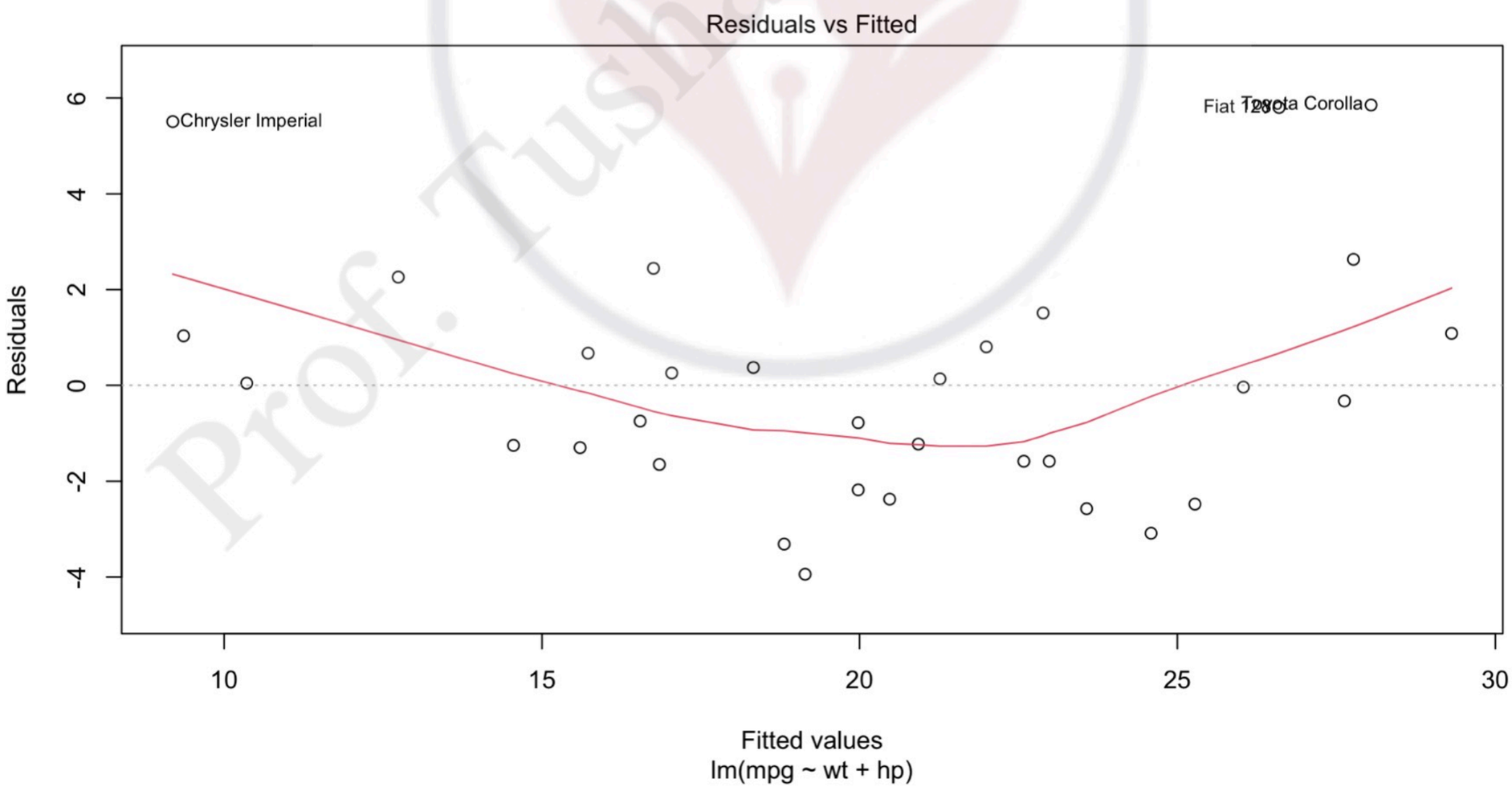

Figure 27: Residuals vs Fitted Plot for Multiple Regression Model

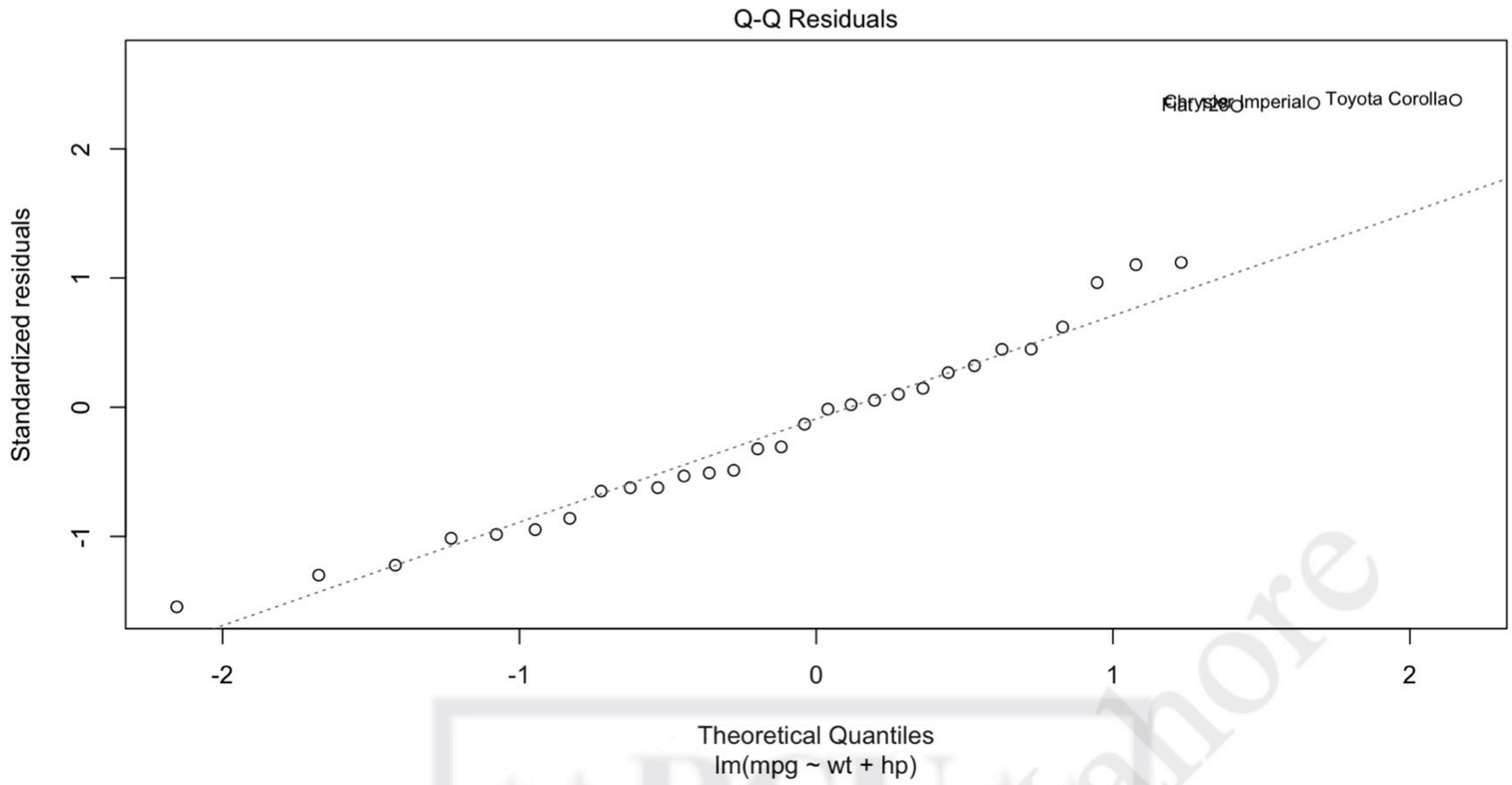

Figure 28: Q-Q Plot of Residuals for Multiple Regression Model

Exercise Questions

- Build a regression model predicting `Petal.Length` from `Sepal.Length` in `iris`.
- Create a multiple regression model predicting `mpg` using `wt`, `hp`, and `drat` in `mtcars`.
- Check the R^2 and Adjusted R^2 of the model in Q2.
- Plot residuals of your regression model in Q1 and interpret the result.
- Build a regression model predicting `Ozone` using `Temp` from the `airquality` dataset.

Viva Questions

- What is regression analysis and why is it important?
- Differentiate between simple and multiple regression.
- What does the slope coefficient represent in linear regression?
- Explain the meaning of R^2 and Adjusted R^2.
- Why do we check residual plots in regression?
- What are the assumptions of linear regression?
- Can regression be used for prediction as well as inference? Give examples.

Conclusion

Students built and evaluated linear regression models, interpreting coefficients, R^2 values, and residuals. This practical showcased regression as a powerful predictive tool while stressing the importance of assumptions and diagnostics for accurate modeling.

Practical 9: Classification Analysis in R

Aim

To perform classification analysis (using Logistic Regression and Decision Trees) and assess model accuracy using performance metrics such as Confusion Matrix and ROC Curve in R.

Objective

- Understand the concept of classification models.
- Apply logistic regression to predict categorical outcomes.
- Build a decision tree classifier for classification problems.
- Evaluate classification performance using confusion matrix, accuracy, precision, recall, and ROC curve.
- Interpret model results for practical decision-making.

Software Required

- R (version 4.0 or above)
- RStudio IDE
- Packages: `dplyr`, `caret`, `rpart`, `rpart.plot`, `pROC`

Theory

Classification

Classification is a supervised learning method where the target variable is categorical (e.g., Yes/No, Spam/Not Spam).

Logistic Regression

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

- Used when dependent variable is binary.
- Decision rule: If probability ≥ 0.5 then Class = 1, else Class = 0.

Decision Tree

- Splits data into branches based on conditions.
- Handles both categorical and numeric data.
- Provides easy interpretability.

Model Evaluation Metrics

- Confusion Matrix: Compares actual vs predicted classes.
- Accuracy, Precision, Recall.
- ROC Curve: Plots True Positive Rate vs False Positive Rate.
- AUC: Higher values indicate better performance.

Program

```r
library(dplyr)
library(caret)
library(rpart)
library(rpart.plot)
library(pROC)
library(ggplot2)

# Load dataset
data("iris")

# Convert to binary classification: setosa vs non_setosa
iris_bin <- iris %>%
  mutate(Species = factor(ifelse(Species == "setosa", "setosa", "non_setosa"),
                          levels = c("non_setosa","setosa")))

# Train-test split
set.seed(123)
idx <- createDataPartition(iris_bin$Species, p = 0.7, list = FALSE)
trainData <- iris_bin[idx, ]
testData  <- iris_bin[-idx, ]

# 1. Logistic Regression
log_model <- glm(Species ~ Sepal.Length + Petal.Length,
                 data = trainData, family = binomial())
log_prob <- predict(log_model, testData, type = "response")
log_class <- ifelse(log_prob > 0.5, "setosa", "non_setosa")
confusionMatrix(factor(log_class, levels=levels(testData$Species)),
                testData$Species)

# ROC Curve
roc_obj <- roc(testData$Species, log_prob,
               levels=c("non_setosa","setosa"))
plot(roc_obj, col="blue", main="ROC Curve - Logistic Regression")

# 2. Decision Tree
tree_model <- rpart(Species ~ Sepal.Length + Petal.Length,
                    data = trainData, method="class")
rpart.plot(tree_model)

# Predictions & Confusion Matrix
tree_pred <- predict(tree_model, testData, type="class")
confusionMatrix(tree_pred, testData$Species)
```

Output

```
> library(dplyr)
> library(caret)
> library(rpart)
> library(rpart.plot)
> library(pROC)
> library(ggplot2)
> # Load dataset
> data("iris")
> # Convert to binary classification: setosa vs non_setosa
> iris_bin <- iris %>%
+   mutate(Species = factor(ifelse(Species == "setosa", "setosa", "non_setosa"),
+                           levels = c("non_setosa","setosa")))
> # Train-test split
> set.seed(123)
> idx <- createDataPartition(iris_bin$Species, p = 0.7, list = FALSE)
> trainData <- iris_bin[idx, ]
> testData  <- iris_bin[-idx, ]
> # 1. Logistic Regression
> log_model <- glm(Species ~ Sepal.Length + Petal.Length,
+                  data = trainData, family = binomial())
Warning messages:
1: glm.fit: algorithm did not converge
2: glm.fit: fitted probabilities numerically 0 or 1 occurred

> log_prob <- predict(log_model, testData, type = "response")
> log_class <- ifelse(log_prob > 0.5, "setosa", "non_setosa")
> confusionMatrix(factor(log_class, levels=levels(testData$Species)),
+                 testData$Species)
Confusion Matrix and Statistics

            Reference
Prediction   non_setosa setosa
  non_setosa         30      0
  setosa              0     15

               Accuracy : 1
                 95% CI : (0.9213, 1)
    No Information Rate : 0.6667
    P-Value [Acc > NIR] : 1.191e-08

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.6667
         Detection Rate : 0.6667
   Detection Prevalence : 0.6667
      Balanced Accuracy : 1.0000

       'Positive' Class : non_setosa

> # ROC Curve
```

```
> roc_obj <- roc(testData$Species, log_prob,
+                levels=c("non_setosa","setosa"))
Setting direction: controls < cases
> plot(roc_obj, col="blue", main="ROC Curve - Logistic Regression")
> # 2. Decision Tree
> tree_model <- rpart(Species ~ Sepal.Length + Petal.Length,
+                     data = trainData, method="class")
> rpart.plot(tree_model)
> # Predictions & Confusion Matrix
> tree_pred <- predict(tree_model, testData, type="class")
> confusionMatrix(tree_pred, testData$Species)
Confusion Matrix and Statistics

            Reference
Prediction   non_setosa setosa
  non_setosa         30      0
  setosa              0     15

               Accuracy : 1
                 95% CI : (0.9213, 1)
    No Information Rate : 0.6667
    P-Value [Acc > NIR] : 1.191e-08

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.6667
         Detection Rate : 0.6667
   Detection Prevalence : 0.6667
      Balanced Accuracy : 1.0000

       'Positive' Class : non_setosa

>
```

Graphical Output

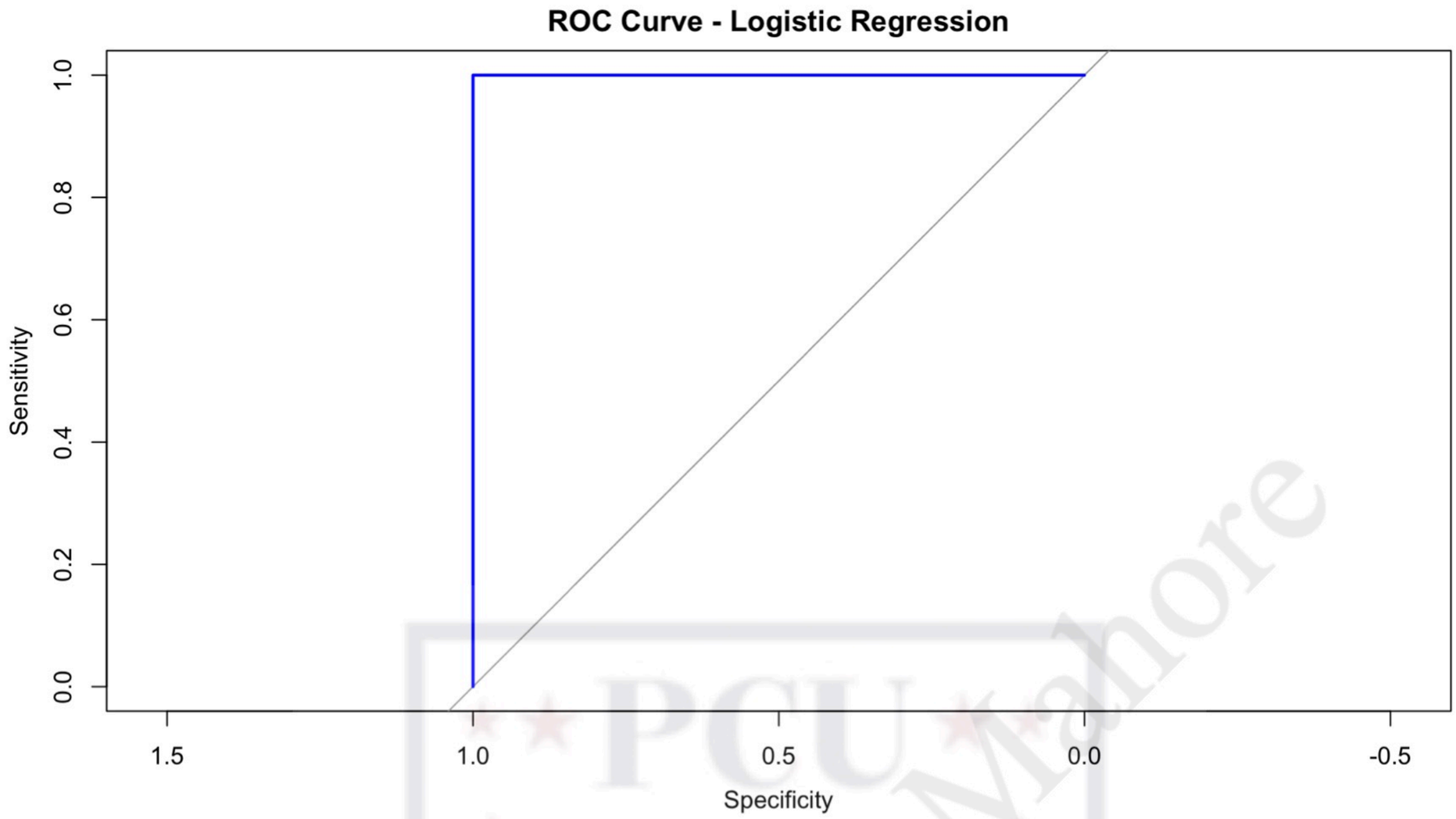

Figure 29: ROC Curve for Logistic Regression Classifier

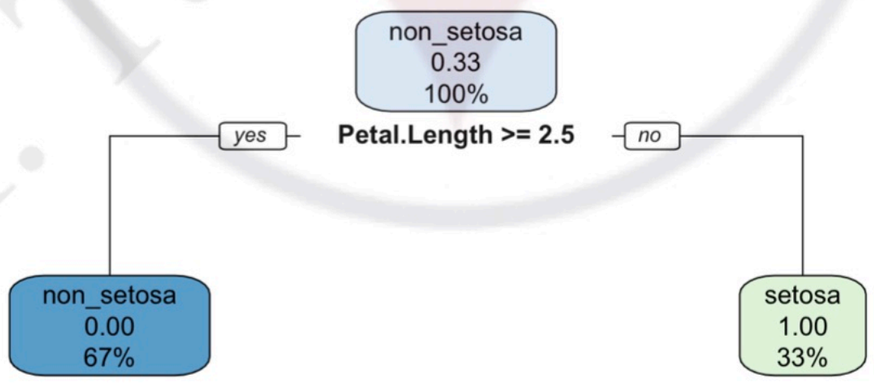

Figure 30: Decision Tree Model for Iris Binary Classification

Exercise Questions

- Build a logistic regression model predicting `am` (automatic/manual) in `mtcars` using `hp` and `wt`.

- Construct a decision tree to classify `Species` in the full `iris` dataset.
- Calculate confusion matrix and accuracy for Q1.
- Plot the ROC curve for the logistic regression model in Q1.
- Compare logistic regression and decision tree performance on the `iris` dataset.

Viva Questions

- Differentiate between regression and classification.
- What is the mathematical form of logistic regression?
- Why do we use a sigmoid function in logistic regression?
- What is a decision tree and what are its advantages?
- Define confusion matrix and explain TP, TN, FP, FN.
- What does the ROC curve represent?
- Why is AUC an important evaluation metric?
- Which is better: logistic regression or decision tree? Why?

Conclusion

Through logistic regression and decision trees, students gained hands-on experience in categorical prediction. Evaluation using confusion matrix and ROC curve emphasized the importance of accuracy and sensitivity measures, making students aware of strengths and limitations of classifiers.

Practical 10: Clustering Analysis in R (K-Means)

Aim

To apply a clustering technique (K-means) in R, visualize cluster assignments, and interpret group characteristics.

Objective

- Understand the concept of unsupervised learning.
- Apply K-means clustering to partition data into groups.
- Determine the optimal number of clusters using the elbow method.
- Visualize cluster assignments using plots.
- Interpret the results and compare cluster characteristics.

Software Required

- R (version 4.0 or above)
- RStudio IDE
- Packages: `dplyr`, `ggplot2`, `factoextra`

Theory

Clustering

Clustering is an unsupervised machine learning technique that groups similar data points together without predefined labels.

K-Means Algorithm

1. Choose K (number of clusters).
2. Randomly assign cluster centers (centroids).
3. Assign each point to the nearest centroid.
4. Update centroids as the mean of assigned points.
5. Repeat until centroids no longer change significantly.

Objective Function

$$J = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2$$

where C_i is cluster i and μ_i is its centroid.

Choosing Number of Clusters

- **Elbow Method:** Plot within-cluster sum of squares (WSS) vs K.
- The point where decrease slows ("elbow") indicates optimal K.

Program

```
library(dplyr)
library(ggplot2)
library(factoextra)

# Load dataset (only numeric features)
data("iris")
iris_data <- iris[, 1:4]

# 1. Elbow Method to find optimal K
fviz_nbclust(iris_data, kmeans, method = "wss") +
  labs(title = "Elbow Method for Optimal K")

# 2. Apply K-means clustering with K=3
set.seed(123)
kmeans_model <- kmeans(iris_data, centers = 3, nstart = 20)

# 3. Cluster assignments
kmeans_model$cluster[1:10]
table(kmeans_model$cluster, iris$Species)

# 4. Visualize clusters
fviz_cluster(kmeans_model, data = iris_data,
             ellipse.type = "norm",
             palette = "jco",
             ggtheme = theme_minimal())
```

Output

```
> library(dplyr)
> library(ggplot2)
> library(factoextra)
Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
> # Load dataset (only numeric features)
> data("iris")
> iris_data <- iris[, 1:4]
> # 1. Elbow Method to find optimal K
> fviz_nbclust(iris_data, kmeans, method = "wss") +
+   labs(title = "Elbow Method for Optimal K")
> # 2. Apply K-means clustering with K=3
> set.seed(123)
> kmeans_model <- kmeans(iris_data, centers = 3, nstart = 20)
> # 3. Cluster assignments
> kmeans_model$cluster[1:10]
 [1] 3 3 3 3 3 3 3 3 3 3
> table(kmeans_model$cluster, iris$Species)

    setosa versicolor virginica
  1      0         48        14
  2      0          2        36
```

```
  3     50           0          0
> # 4. Visualize clusters
> fviz_cluster(kmeans_model, data = iris_data,
+              ellipse.type = "norm",
+              palette = "jco",
+              ggtheme = theme_minimal())
>
```

Graphical Output

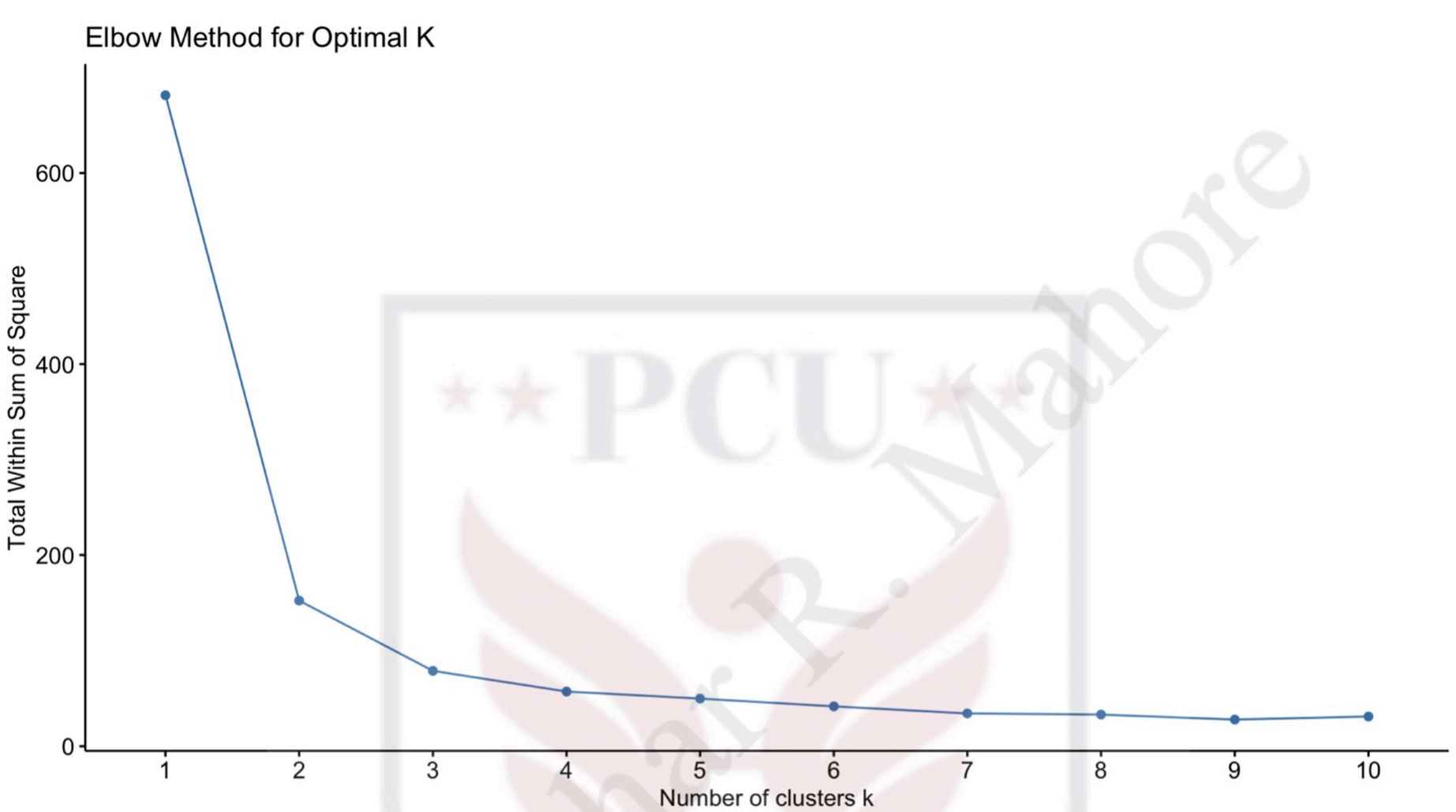

Figure 31: Elbow Method Plot to Determine Optimal Number of Clusters (K)

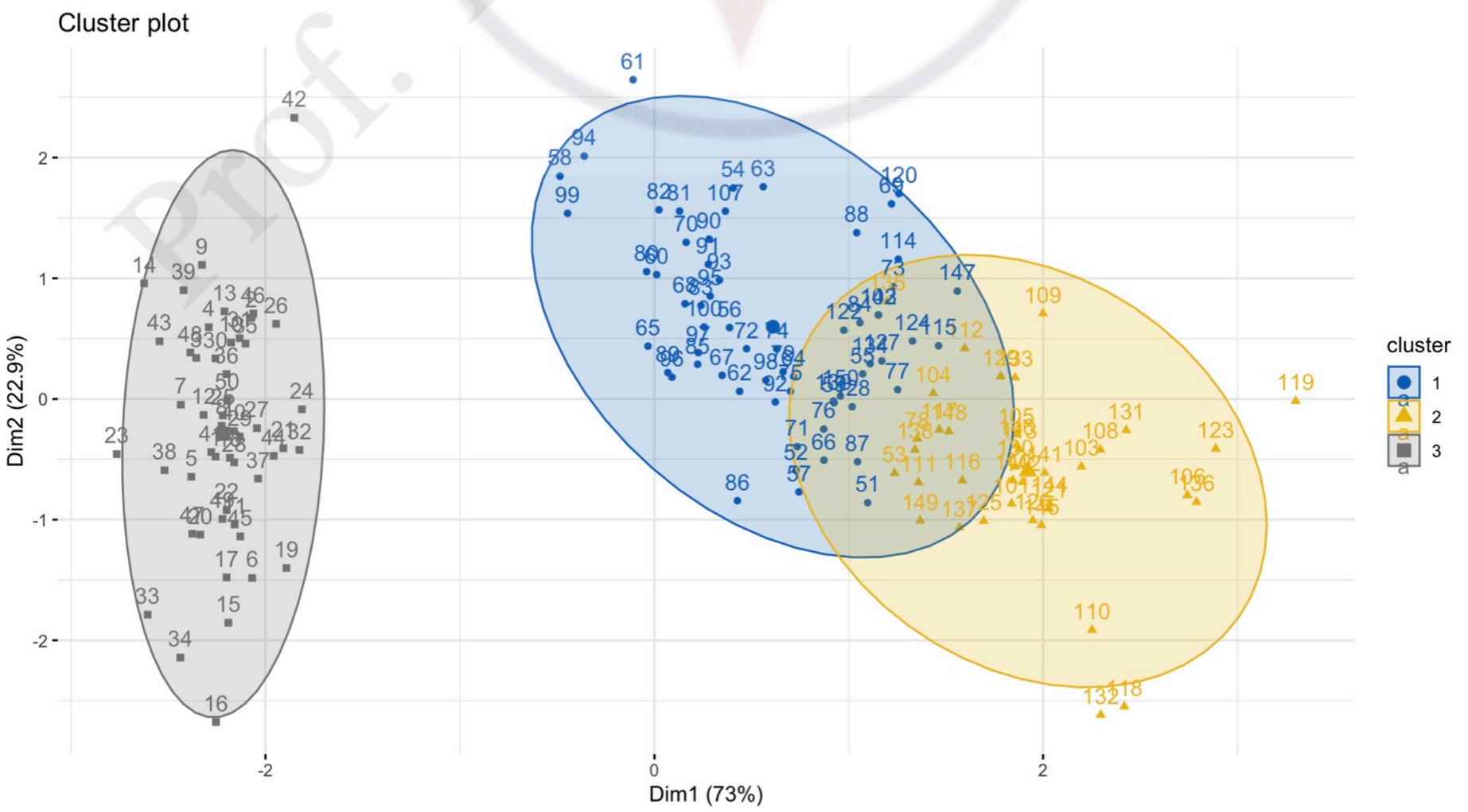

Figure 32: K-means Clustering Visualization of the Iris Dataset (K=3)

Exercise Questions

- Perform K-means clustering with $K = 4$ on `iris` dataset and compare with species labels.
- Apply clustering on `mtcars` using `mpg`, `hp`, and `wt`. Interpret clusters.
- Use the elbow method to find optimal number of clusters for `iris`.
- Visualize clusters using PCA-based scatterplot in `factoextra`.
- Create a subset of `iris` with only `Sepal.Length` and `Sepal.Width` and cluster it.

Viva Questions

- What is the difference between supervised and unsupervised learning?
- Explain the working of K-means algorithm.
- What is the objective function of K-means?
- How do you choose the number of clusters in K-means?
- Why do different runs of K-means sometimes give different results?
- What are real-world applications of clustering?
- What are the limitations of K-means clustering?

Conclusion

Students applied K-means clustering to group similar observations without prior labels. By using the elbow method and visualizing clusters, they learned how unsupervised learning can reveal natural groupings in data, useful for segmentation and pattern discovery.

Practical 11: Mini Project Guidelines

Objective

The mini project is intended to provide students with hands-on experience in applying concepts of Data Science and Analytics using R. Students are expected to demonstrate their ability to collect, clean, analyze, and visualize data, and present meaningful insights.

Scope of the Project

- The project should focus on solving a real-world problem using data analysis techniques.
- Students must apply the concepts learned in practical sessions: data import, preprocessing, EDA, statistical analysis, regression, classification, clustering, and visualization.
- Use of inbuilt datasets is permitted initially, but projects with external datasets (CSV/Excel/API) are encouraged.
- The project can be individual or group-based (maximum 3 students per group).

Suggested Project Areas

- Exploratory analysis of public datasets (e.g., Iris, Titanic, COVID-19, Stock Market, Weather).
- Predictive modeling (e.g., house prices, student performance, sales forecasting).
- Classification problems (e.g., disease prediction, spam detection, sentiment analysis).
- Clustering applications (e.g., customer segmentation, image grouping).
- Any innovative idea that demonstrates data-driven decision-making.

Project Deliverables

Each project submission must include:

1. **Project Report (in PDF/Word format)** covering:
    - Title of the project
    - Objective and problem statement
    - Dataset description (source, size, features)
    - Methodology (steps of analysis, tools used)
    - Results (statistical findings, visualizations, models)
    - Conclusion and insights
    - Future scope
2. **R Script/Notebook** with well-commented code.
3. **Presentation (PPT)** summarizing project highlights (8–10 slides).

Evaluation Criteria

Projects will be evaluated based on the following parameters:

- Problem formulation and clarity of objectives – 10 marks
- Data collection and preprocessing – 10 marks
- Application of analysis techniques (EDA, stats, models) – 30 marks
- Interpretation of results and insights – 20 marks
- Quality of report, code, and presentation – 30 marks

Total: 100 Marks (Converted to and merged with 50 marks as per practical assessment scheme).

General Instructions

- Students must use R/RStudio for all coding and analysis.
- Plagiarism is strictly prohibited. Each project must be original.
- Proper citations must be provided if external datasets or references are used.
- Submission deadlines must be strictly followed.
- Late submissions will not be entertained without valid justification.

Conclusion

The mini project provides students with the opportunity to integrate and apply knowledge gained throughout the laboratory course. It encourages independent thinking, teamwork, and problem-solving through data-driven approaches.