

**DATA ANALYTICS USING R LAB-R22**

Laboratory Manual

## II YEAR B.TECH-II SEM (CSD)





DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (DATA SCIENCE )



1



# LAB MANUAL

# DATA ANALYTICS USING R LAB-A6DS03



# Department of CSE(Data Science)

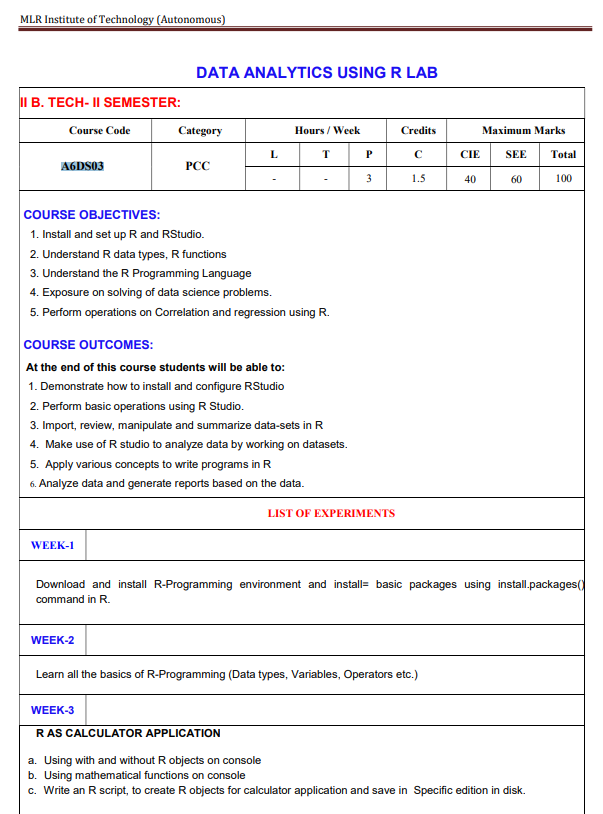


|  |  |  |  |
| --- | --- | --- | --- |
| Document No: MLRIT/LAB MANUAL/DS | Date of Issue  20 /2/2024 | Compiled by  D.Jyothi | Authorized by Dr.M.Chiranjeevi  HOD,CSD |

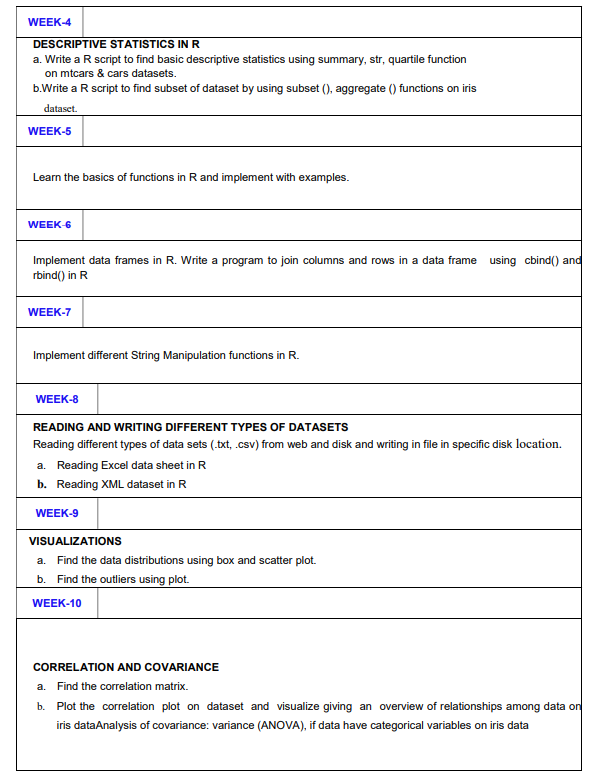
2

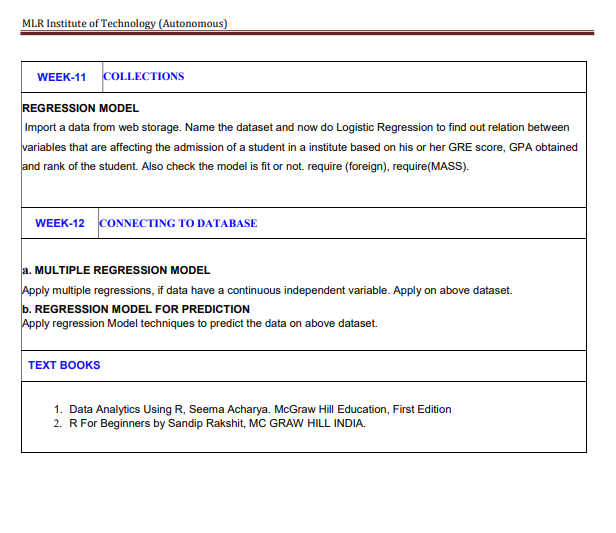


3



MLR-22





PROGRAM OUTCOMES

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| COURSE ARTICULATION MATRIX | | | | | | | | | | | | | | | |
| COURSE CODE | COURSE OUTCOME | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 | PSO1 | PSO2 |
|  | CO1 | 3 | 3 |  |  |  |  |  |  |  |  |  |  |  |  |
| CO2 |  | 3 | 3 |  |  |  |  |  |  |  |  |  |  |  |
| CO3 |  |  |  |  |  | 3 |  |  | 3 |  |  |  |  |  |
| CO4 |  |  |  |  |  |  |  |  | 3 | 3 |  |  | 3 |  |
| CO5 |  |  |  |  |  |  |  |  |  | 3 |  | 3 |  | 3 |



#### INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| SNO | EXPERIMEN T NO | NAME OF THE EXPERIMENT | PAGE NO |
| 1 | 0 | Introduction | 8 |
| 2 | 1 | Install R-Programming Environment | 10 |
| 3 | 2 | Install R-Programming Environment | 11 |
| 4 | 3 | R As Calculator Application | 13 |
| 5 | 4 | Descriptive Statistics In R | 16 |
| 6 | 5 | Basics Of Functions In R | 18 |
| 7 | 6 | Data Frames In R | 20 |
| 8 | 7 | String Manipulation Functions In R | 21 |
| 9 | 8 | Reading Different Types Of Data Sets (.Txt, .Csv) From Web | 22 |
| 10 | 9 | Visualizations | 23 |
| 11 | 10 | Correlation And Covariance | 24 |
| 12 | 11 | Regression Model | 26 |
| 13 | 12 | Multiple Regression Model | 27 |



DATA ANALYTICS USING R LAB MANUAL

Week-0-introduction COURSE OBJECTIVES:

1. Install and set up R and RStudio.

2. Understand R data types, R functions

3. Understand the R Programming Language

4. Exposure on solving of data science problems.

5. Perform operations on Correlation and regression using R.

COURSE OUTCOMES: At the end of this course students will be able to:

1. Demonstrate how to install and configure RStudio

2. Perform basic operations using R Studio.

3. Import, review, manipulate and summarize data-sets in R

4. Make use of R studio to analyze data by working on datasets

5. Apply various concepts to write programs in R

6. Analyze data and generate reports based on the data.

## Tools and Language

R studio on the windows operating system

#### Lab Related Instructions:

1. Submission : All submissions must be made using *LMS* after completion of lab week wise
2. Programming Language : All programs must be written in the C programming language.
3. Plagiarism : We have a zero tolerance policy towards plagiarism. Any case of cheating or stealing codes would result in imposition of “ Unfair Means “ charge on you and you will have to face the disciplinary committee of the Department leading to probable de-registration from the course.
4. Comments and Indentation : We want your programs to follow a proper indentation style as instructed in the



|  |  |
| --- | --- |
| WEEK - 1 |  |
| Download and install R-Programming environment and install= basic packages using install.packages() command in R. | |

Step 1: Download and Install R

For Windows:

1.Go to the CRAN (Comprehensive R Archive Network) website: CRAN.

2.Click on "Download R for Windows" and choose a nearby CRAN mirror.

3.Download the base distribution (e.g., R-4.x.x-win.exe).

4.Run the downloaded executable file and follow the installation instructions.

Step 2: Open R Console

Open R by launching the R console. On Windows, you can find it in the Start menu or desktop. On macOS or Linux, you can open a terminal and type R.

Step 3: Install Packages

In the R console, you can install packages using the install.packages() command. For example, let's install the "tidyverse" package, which includes several essential packages for data manipulation:

Step 3: Install Packages

In the R console, you can install packages using the install.packages() command. For example, let's install the "tidyverse" package, which includes several essential packages for data manipulation:

install.packages("tidyverse")

install.packages("ggplot2")

library(tidyverse)

|  |  |
| --- | --- |
| WEEK - 2 |  |
| Learn all the basics of R-Programming (Data types, Variables, Operators etc | |

a. Numeric:

num\_var <- 42

b. Character:

char\_var <- "Hello, R!"

c. Logical:

logical\_var <- TRUE

d. Factor

factor\_var <- factor(c("apple", "orange", "banana"))

e. Vector:

numeric\_vector <- c(1, 2, 3, 4, 5)

char\_vector <- c("a", "b", "c")

f. List:

list\_var <- list(numeric\_vector, char\_vector)

g. Matrix:

matrix\_var <- matrix(c(1, 2, 3, 4), nrow = 2, ncol = 2)

h. Data Frame:

data\_frame\_var <- data.frame(Name = c("John", "Alice"), Age = c(25, 30))

2. Variables:

x <- 10 # Assigns the value 10 to the variable x

y <- x + 5 # Assigns the value of x + 5 to the variable y

3. Operators:

a. Arithmetic Operators:

a <- 10

b <- 5

addition <- a + b

subtraction <- a - b

multiplication <- a \* b

division <- a / b

exponentiation <- a ^ b

modulo <- a %% b

b. Relational Operators:

a <- 10

b <- 5

greater\_than <- a > b

less\_than <- a < b

equal\_to <- a == b

not\_equal\_to <- a != b

greater\_than\_equal\_to <- a >= b

less\_than\_equal\_to <- a <= b

c. Logical Operators

logical\_and <- TRUE & FALSE

logical\_or <- TRUE | FALSE

logical\_not <- !TRUE

4. Functions:

# Define a function

multiply <- function(x, y) {

result <- x \* y

return(result)

}

# Call the function

product <- multiply(3, 4)

print(product)

|  |  |
| --- | --- |
| WEEK - 3 |  |
| R AS CALCULATOR APPLICATION  a. Using with and without R objects on console  b. Using mathematical functions on console  c. Write an R script, to create R objects for calculator application and save in Specific edition in disk | |

1. Without R Objects:

# Addition

2 + 3

# Subtraction

7 - 4

# Multiplication

5 \* 6

# Division

10 / 2

# Exponentiation

3^2

# Modulo

15 %% 4

2. With R Objects:

# Create variables

a <- 2

b <- 3

# Addition

result\_addition <- a + b

print(result\_addition)

# Subtraction

result\_subtraction <- a - b

print(result\_subtraction)

# Multiplication

result\_multiplication <- a \* b

print(result\_multiplication)

# Division

result\_division <- a / b

print(result\_division)

# Exponentiation

result\_exponentiation <- a^b

print(result\_exponentiation)

# Modulo

result\_modulo <- 15 %% 4

print(result\_modulo)

b. Using mathematical functions on console

Trigonometric Functions:

# Sine

sin(pi/2)

# Cosine

cos(pi)

# Tangent

tan(pi/4)

Exponential and Logarithmic Functions:

# Exponential

exp(2)

# Natural Logarithm

log(10)

# Logarithm with base 10

log10(100)

Square Root:

sqrt(25)

Absolute Value:

abs(-7)

Round and Floor/Ceiling

# Round to the nearest integer

round(3.14159)

# Floor (round down)

floor(3.999)

# Ceiling (round up)

ceiling(3.001)

Random Number Generation:

# Generate a random number between 0 and 1

runif(1)

# Generate a random integer between 1 and 10

sample(1:10, 1)

c. Write an R script, to create R objects for calculator application and save in Specific edition in disk.

1. Open a text editor (e.g., Notepad, RStudio, VSCode) and create a new script. Let's name it calculator\_script.R.

2. Add the following content to the script:

# Calculator Script

# Define variables

a <- 5

b <- 3

# Perform calculations

addition\_result <- a + b

subtraction\_result <- a - b

multiplication\_result <- a \* b

division\_result <- a / b

# Display results

cat("Addition Result:", addition\_result, "\n")

cat("Subtraction Result:", subtraction\_result, "\n")

cat("Multiplication Result:", multiplication\_result, "\n")

cat("Division Result:", division\_result, "\n")

# Save R objects to a file

save(a, b, addition\_result, subtraction\_result, multiplication\_result, division\_result, file = "calculator\_objects.RData")

3. Save the script.

4. Run the script in your R environment. You can do this by opening R or RStudio, navigating to the directory where the script is saved, and running:

source("calculator\_script.R")

This script defines variables a and b, performs basic calculations, displays the results, and saves the R objects into a file named calculator\_objects.RData. You can customize the script as needed for your specific calculations.

After running the script, you'll find the calculator\_objects.RData file in the same directory as your script. You can load these objects back into R later using the load() function:

load("calculator\_objects.RData")

|  |  |
| --- | --- |
| WEEK - 4 |  |
| DESCRIPTIVE STATISTICS IN R  a. Write a R script to find basic descriptive statistics using summary, str, quartile function on mtcars & cars datasets. b.Write a R script to find subset of dataset by using subset (), aggregate () functions on iris dataset. | |

a. Write a R script to find basic descriptive statistics using

• Loads the mtcars and cars datasets.

• Displays the structure of each dataset using str.

• Provides summary statistics using summary.

• Calculates and displays quartiles using the quantile function.

# Descriptive Statistics Script

# Load datasets

data(mtcars)

data(cars)

# Display structure of mtcars dataset

cat("Structure of mtcars dataset:\n")

str(mtcars)

# Display summary statistics for mtcars dataset

cat("\nSummary Statistics for mtcars dataset:\n")

summary(mtcars)

# Display quartiles for mtcars dataset

cat("\nQuartiles for mtcars dataset:\n")

quantiles\_mtcars <- quantile(mtcars$mpg)

print(quantiles\_mtcars)

# Display structure of cars dataset

cat("\nStructure of cars dataset:\n")

str(cars)

# Display summary statistics for cars dataset

cat("\nSummary Statistics for cars dataset:\n")

summary(cars)

# Display quartiles for cars dataset

cat("\nQuartiles for cars dataset:\n")

quantiles\_cars <- quantile(cars$speed)

print(quantiles\_cars)

b.Write a R script to find subset of dataset by using subset (), aggregate () functions on iris dataset.

1. Loads the iris dataset.

2. Displays the structure of the iris dataset using str.

3. Creates a subset of the dataset where Sepal.Length is greater than 5 using subset().

4. Displays the subset of the dataset.

5. Uses aggregate() to calculate mean values of Sepal.Length and Sepal.Width grouped by Species.

6. Displays the aggregated results.

# Subset and Aggregate Script

# Load iris dataset

data(iris)

# Display structure of iris dataset

cat("Structure of iris dataset:\n")

str(iris)

# Subset the dataset to include only rows where Sepal.Length is greater than 5

subset\_data <- subset(iris, Sepal.Length > 5)

# Display the subset of the dataset

cat("\nSubset of iris dataset where Sepal.Length > 5:\n")

print(subset\_data)

# Aggregate mean values of Sepal.Length and Sepal.Width by Species

agg\_result <- aggregate(cbind(Sepal.Length, Sepal.Width) ~ Species, data = iris, FUN = mean)

# Display aggregated results

cat("\nAggregated mean values by Species:\n")

print(agg\_result)

|  |  |
| --- | --- |
| WEEK - 5 |  |
| Learn the basics of functions in R and implement with examples. | |

Defining a Function:

You can define a function in R using the function() keyword. Here's the basic syntax:

• function\_name: The name you give to your function.

• arg1, arg2, ...: Arguments that the function can take.

function\_name <- function(arg1, arg2, ...) {

# Code to be executed

# Return statement (if needed)

}

Example 1: Simple Addition Function

# Define a function to add two numbers

add\_numbers <- function(x, y) {

result <- x + y

return(result)

}

# Call the function

sum\_result <- add\_numbers(5, 3)

print(sum\_result) # Output: 8

Example 2: Function with Default Values

# Define a function with default values

greet\_person <- function(name = "Guest") {

message <- paste("Hello,", name, "!")

return(message)

}

# Call the function

greeting <- greet\_person("John")

print(greeting) # Output: Hello, John!

# Call the function without specifying a name

default\_greeting <- greet\_person()

print(default\_greeting) # Output: Hello, Guest!

Example 3: Function with Multiple Return Values

# Define a function with multiple return values

calculate\_stats <- function(numbers) {

mean\_val <- mean(numbers)

median\_val <- median(numbers)

return(c(mean\_val, median\_val))

}

# Call the function

data <- c(10, 20, 30, 40, 50)

stats <- calculate\_stats(data)

print(stats[1]) # Output: Mean value

print(stats[2]) # Output: Median value

Example 4: Recursive Function

# Define a recursive function to calculate factorial

factorial <- function(n) {

if (n <= 1) {

return(1)

} else {

return(n \* factorial(n - 1))

}

}

# Call the function

result <- factorial(5)

print(result) # Output: 120

Example 5: Anonymous Function (Lambda)

# Define an anonymous function (lambda)

multiply <- function(x, y) {

return(x \* y)

}

# Using the anonymous function (lambda)

result\_lambda <- (function(x, y) x \* y)(5, 3)

print(result\_lambda) # Output: 15

|  |  |
| --- | --- |
| WEEK - 6 |  |
| Implement data frames in R. Write a program to join columns and rows in a data frame using cbind() and rbind() in R | |

# Creating a data frame

df1 <- data.frame(

Name = c("John", "Alice"),

Age = c(25, 30),

Score = c(85, 92)

)

df2 <- data.frame(

Name = c("Bob", "Eva"),

Age = c(22, 28),

Score = c(78, 89)

)

# Displaying original data frames

cat("Original Data Frames:\n")

print(df1)

print(df2)

# Using cbind() to join columns

combined\_df\_columns <- cbind(df1, df2$Score)

cat("\nData Frame After Joining Columns using cbind():\n")

print(combined\_df\_columns)

# Using rbind() to join rows

combined\_df\_rows <- rbind(df1, df2)

cat("\nData Frame After Joining Rows using rbind():\n")

print(combined\_df\_rows)

|  |  |
| --- | --- |
| WEEK - 7 |  |
| Implement different String Manipulation functions in R | |

1. Concatenation:

# Concatenate strings

string1 <- "Hello"

string2 <- "World"

concatenated\_string <- paste(string1, string2)

print(concatenated\_string)

2. Substring:

# Extract a substring

original\_string <- "Hello, World!"

substring <- substr(original\_string, start = 1, stop = 5)

print(substring)

3. String Length:

# Get the length of a string

string <- "Data Science"

length\_of\_string <- nchar(string)

print(length\_of\_string)

4. Uppercase and Lowercase:

# Convert to uppercase

uppercase\_string <- toupper("hello")

print(uppercase\_string)

# Convert to lowercase

lowercase\_string <- tolower("WORLD")

print(lowercase\_string)

5. String Replacement:

# Replace a substring

original\_string <- "I like apples."

replaced\_string <- gsub("apples", "bananas", original\_string)

print(replaced\_string)

6. String Splitting:

# Split a string into a vector

sentence <- "This is a sample sentence."

word\_vector <- strsplit(sentence, " ")[[1]]

print(word\_vector)

7. String Joining:

# Join elements of a vector into a string

word\_vector <- c("This", "is", "a", "sample", "sentence.")

joined\_string <- paste(word\_vector, collapse = " ")

print(joined\_string)

|  |  |
| --- | --- |
| WEEK - 8 |  |
| Reading different types of data sets (.txt, .csv) from web and disk and writing in file in specific disk location.  a. Reading Excel data sheet in R  b. Reading XML dataset in R | |

a. Reading Excel data sheet in R

Install and load the readxl package:

install.packages("readxl")

library(readxl)

Read Excel Data:

# Read Excel data

excel\_data <- read\_excel("example.xlsx")

# Display the data

print(excel\_data)

b. Reading XML dataset in R

Install and load the xml2 package:

install.packages("xml2")

library(xml2)

Read XML Data:

# Read XML data

xml\_data <- read\_xml("example.xml")

# Display the XML content

print(xml\_data)

Extract Information from XML:

# Find all elements with a specific tag

nodes <- xml\_find\_all(xml\_data, "//element\_tag")

# Extract content from nodes

content <- xml\_text(nodes)

print(content)

|  |  |
| --- | --- |
| WEEK - 9 |  |
| VISUALIZATIONS  a. Find the data distributions using box and scatter plot.  b. Find the outliers using plot. | |

Boxplot:

# Load iris dataset

data(iris)

# Create boxplot for Sepal.Length by Species

boxplot(Sepal.Length ~ Species, data = iris, main = "Boxplot of Sepal Length by Species", xlab = "Species", ylab = "Sepal Length")

Scatter Plot:

# Create scatter plot for Sepal.Length and Sepal.Width

plot(iris$Sepal.Length, iris$Sepal.Width, main = "Scatter Plot of Sepal Length vs Sepal Width", xlab = "Sepal Length", ylab = "Sepal Width", col = iris$Species)

legend("topright", legend = levels(iris$Species), col = 1:3, pch = 1:3, title = "Species")

|  |  |
| --- | --- |
| WEEK - 10 |  |
| CORRELATION AND COVARIANCE  a. Find the correlation matrix.  b. Plot the correlation plot on dataset and visualize giving an overview of relationships among data on iris dataAnalysis of covariance: variance (ANOVA), if data have categorical variables on iris data | |

a. Find the correlation matrix.

# Load iris dataset

data(iris)

# Calculate the correlation matrix

correlation\_matrix <- cor(iris[, c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")])

# Display the correlation matrix

print(correlation\_matrix)

correlation\_matrix <- cor(your\_dataset)

b. . Plot the correlation plot on dataset and visualize giving an overview of relationships among data on iris dataAnalysis of covariance: variance (ANOVA), if data have categorical variables on iris data

Visualizing Correlation Matrix:

# Install and load necessary libraries

install.packages("corrplot")

library(corrplot)

# Load iris dataset

data(iris)

# Calculate the correlation matrix

correlation\_matrix <- cor(iris[, c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")])

# Display the correlation matrix

print(correlation\_matrix)

# Visualize the correlation matrix using corrplot

corrplot(correlation\_matrix, method = "color", type = "upper", order = "hclust", tl.cex = 0.7)

Analysis of Covariance (ANCOVA):

# Perform ANCOVA

ancova\_result <- aov(Sepal.Width ~ Sepal.Length \* Species, data = iris)

# Display ANCOVA summary

summary(ancova\_result)

|  |  |
| --- | --- |
| WEEK - 11 |  |
| REGRESSION MODEL Import a data from web storage. Name the dataset and now do Logistic Regression to find out relation between variables that are affecting the admission of a student in a institute based on his or her GRE score, GPA obtained and rank of the student. Also check the model is fit or not. require (foreign), require(MASS). | |

# Install and load necessary libraries

install.packages(c("foreign", "MASS"))

library(foreign)

library(MASS)

# Replace the URL with the actual URL of your dataset

dataset\_url <- "https://example.com/your\_dataset.csv"

# Import the dataset

admission\_data <- read.csv(dataset\_url)

# Display the structure of the dataset

str(admission\_data)

# Perform logistic regression

logistic\_model <- glm(admitted ~ gre + gpa + rank, data = admission\_data, family = "binomial")

# Display summary of the logistic regression model

summary(logistic\_model)

|  |  |
| --- | --- |
| WEEK - 12 |  |
| a. MULTIPLE REGRESSION MODEL Apply multiple regressions, if data have a continuous independent variable. Apply on above dataset.  b. REGRESSION MODEL FOR PREDICTION Apply regression Model techniques to predict the data on above dataset. | |

a. MULTIPLE REGRESSION MODEL Apply multiple regressions, if data have a continuous independent variable. Apply on above dataset.

# Install and load necessary libraries

install.packages(c("foreign", "MASS"))

library(foreign)

library(MASS)

# Replace the URL with the actual URL of your dataset

dataset\_url <- "https://example.com/your\_dataset.csv"

# Import the dataset

admission\_data <- read.csv(dataset\_url)

# Display the structure of the dataset

str(admission\_data)

# Perform multiple regression

multiple\_regression\_model <- lm(admitted ~ gre + gpa, data = admission\_data)

# Display summary of the multiple regression model

summary(multiple\_regression\_model)

1. Apply regression Model techniques to predict the data on above dataset.