Lab Exercises:

Exercise 1: **Basic Arithmetic Operations**

You are working as a cashier at a grocery store. Your task is to create a program that simulates the checkout process for a customer's shopping cart. The program should calculate the total cost of the items, including tax, and provide a detailed receipt.

1. DeCine a list of products, each represented as a dictionary with keys: "name", "price", and "quantity".
2. Allow the cashier to input the products in the customer's shopping cart, including the name, price, and quantity of each item.
3. Calculate the subtotal (price \* quantity) for each item and display a detailed receipt with product names, quantities, prices, and subtotals.
4. Calculate the total cost of the items in the cart before tax.
5. Apply a tax rate (e.g., 8%) to the total cost to calculate the tax amount.
6. Calculate the Cinal total cost by adding the tax amount to the total cost before tax.

Code in R: # List of products # List of products

products <- list(

list(name = "Apple", price = 0.5), list(name = "Banana", price = 0.3), list(name = "Milk", price = 2), list(name = "Bread", price = 1.5), list(name = "Eggs", price = 2.5)

)

# Initialize shopping cart as an empty list shopping\_cart <- list()

# DeCine items to be added to the cart cart\_items\_to\_add <- list(

list(name = "Apple", quantity = 3), list(name = "Milk", quantity = 2)

)

# Add items to the shopping cart for (item in cart\_items\_to\_add) { product\_name <- item$name quantity <- item$quantity

# Find the product in the list product <- NULL

for (p in products) {

if (p$name == product\_name) { product <- p

break

}

}

if (!is.null(product)) {

cart\_item <- list(name = product$name, price = product$price, quantity = quantity) shopping\_cart <- c(shopping\_cart, list(cart\_item))

cat("Item added to cart.\n")

} else {

cat("Product not found.\n")

}

}

# Calculate and display receipt subtotal <- 0 cat("\nReceipt:\n")

for (item in shopping\_cart) {

item\_subtotal <- item$price \* item$quantity

cat(sprintf("%s (%d units) - Price: $%.2f - Subtotal: $%.2f\n", item$name, item$quantity, item$price, item\_subtotal))

subtotal <- subtotal + item\_subtotal

}

tax\_rate <- 0.08

tax\_amount <- subtotal \* tax\_rate total\_cost\_before\_tax <- subtotal

total\_cost <- total\_cost\_before\_tax + tax\_amount

cat("\nSubtotal: $%.2f\n", subtotal)

cat("Tax Amount (8%): $%.2f\n", tax\_amount) cat("Total Cost: $%.2f\n", total\_cost)

Exercise 2: **Loops Operations**

You have been tasked with creating a program that calculates and assigns grades for students enrolled in multiple courses. The program will take input for the marks obtained by 10 students in 5 different courses, compute the total and average marks for each student, and assign corresponding grades based on their average performance.

Declare constants for the number of students (num\_students) and the number of courses (num\_courses).

Initialize an empty list to store student information. For each student:

* Input the student's name.
* Input marks for each of the 5 courses.
* Calculate the total marks and average marks.
* Determine the grade based on the average marks using a grading scale.
* Display the student information, including their name, individual course marks, total marks, average marks, and the assigned grade.

Program:

# Constants num\_students <- 5

num\_courses <- 5

# PredeCined student names

student\_names <- c("Arun Rahul", "Bheem Kumar","Raj jumar","Jahal A R","Suresh")

# PredeCined course marks for each student course\_marks <- matrix(c(

85, 92, 78, 88, 95,

75, 80, 85, 70, 60,

100,78,56,34,56,

78,45,67,89,90,

89,80,67,78,90

), nrow = num\_students, byrow = TRUE)

# Initialize a list to store student information student\_records <- list()

# Loop for each student

for (student\_index in 1:num\_students) { student\_name <- student\_names[student\_index]

# Initialize variables for calculations

total\_marks <- sum(course\_marks[student\_index, ]) average\_marks <- total\_marks / num\_courses

# Determine grade based on average marks grade <- ifelse(average\_marks >= 90, "A",

ifelse(average\_marks >= 80, "B", ifelse(average\_marks >= 70, "C",

ifelse(average\_marks >= 60, "D", "F"))))

# Store student information in a record

student\_record <- list(name = student\_name, marks = course\_marks[student\_index, ], total = total\_marks, average = average\_marks, grade = grade)

student\_records <- c(student\_records, list(student\_record))

}

# Display student information cat("\nStudent Grade Report:\n")

for (student\_record in student\_records) {

cat("\nName:", student\_record$name, "\n") cat("Marks:", student\_record$marks, "\n") cat("Total Marks:", student\_record$total, "\n") cat("Average Marks:", student\_record$average, "\n") cat("Grade:", student\_record$grade, "\n")

}

Output of the program:

Student Grade Report:

Name: Arun Rahul Marks: 85 92 78 88 95

Total Marks: 438 Average Marks: 87.6 Grade: B

Name: Bheem Kumar Marks: 75 80 85 70 60

Total Marks: 370 Average Marks: 74 Grade: C

Name: Raj jumar Marks: 100 78 56 34 56

Total Marks: 324 Average Marks: 64.8 Grade: D

Name: Jahal A R Marks: 78 45 67 89 90

Total Marks: 369 Average Marks: 73.8 Grade: C

Name: Suresh

Marks: 89 80 67 78 90

Total Marks: 404 Average Marks: 80.8 Grade: B

# Exercise 3: Conditional statement, Loops and Functions:

You are developing a library management system that needs a **<ine calculation feature.** Write a program that takes the number of days a book is overdue and calculates the Cine amount based on the library's policy. The policy states that for the Cirst 7 days, there is no Cine. After 7 days, a Cixed Cine per day is charged. Additionally, there's a cap on the Cine amount after 30 days.

Input the number of days the book is overdue.

* Use conditional statements to calculate the Cine amount based on the library's policy.
* Display the Cine amount along with a message indicating whether the Cine is within the cap or exceeded it.

# Program:

# Function to calculate library Cine calculate\_Cine <- function(days\_overdue) { if (days\_overdue <= 7) {

Cine <- 0 # No Cine for the Cirst 7 days

} else if (days\_overdue <= 30) {

Cine\_per\_day <- 2 # Fine per day after 7 days Cine <- (days\_overdue - 7) \* Cine\_per\_day

} else {

Cine\_cap <- 50 # Maximum Cine after 30 days Cine <- Cine\_cap

}

return(Cine)

}

# Input number of days overdue

days\_overdue <- as.integer(readline("Enter the number of days the book is overdue: "))

# Calculate Cine

Cine\_amount <- calculate\_Cine(days\_overdue)

# Display Cine information

cat("Fine Amount:", Cine\_amount, "\n") if (Cine\_amount == 0) {

cat("No Cine. Thank you for returning the book on time!\n")

} else {

if (days\_overdue > 30) {

cat("Fine exceeds the maximum cap. Please contact the library.\n")

} else {

cat("Please pay the Cine within the speciCied period.\n")

}

}

### Output:

Enter the number of days the book is overdue: 20 Fine Amount: 26

Please pay the fine within the specified period.

Exercise 4: **arrays and Functions:**

You are developing an inventory management system for a small store. The system needs to handle inventory items and their quantities. Write a program that uses arrays to store inventory items and their quantities, and includes functions to add new items, update quantities, and display the inventory.

* DeCine an array to store inventory items.
* DeCine an array to store corresponding quantities.
* Implement functions to:
  + Add a new item along with its quantity.
  + Update the quantity of an existing item.
  + Display the inventory items and quantities.
  + Use the functions to manage the inventory and handle user interactions.

Program:

# Initialize arrays for inventory items and quantities inventory\_items <- character(0) inventory\_quantities <- numeric(0)

# Function to add a new item with quantity add\_item <- function(item, quantity) { inventory\_items <<- c(inventory\_items, item)

inventory\_quantities <<- c(inventory\_quantities, quantity) cat("Item added to inventory.\n")

}

# Function to update quantity of an existing item update\_quantity <- function(item, new\_quantity) { if (item %in% inventory\_items) {

item\_index <- which(inventory\_items == item) inventory\_quantities[item\_index] <<- new\_quantity cat("Quantity updated.\n")

} else {

cat("Item not found in inventory.\n")

}

}

# Function to display inventory display\_inventory <- function() { cat("Inventory Items and Quantities:\n") for (i in 1:length(inventory\_items)) {

cat(sprintf("%s: %d\n", inventory\_items[i], inventory\_quantities[i]))

}

}

# Main program

while (TRUE) {

cat("\n1. Add Item\n2. Update Quantity\n3. Display Inventory\n4. Exit\n") choice <- as.integer(readline("Enter your choice: "))

if (choice == 1) {

item <- readline("Enter item name: ")

quantity <- as.integer(readline("Enter quantity: ")) add\_item(item, quantity)

} else if (choice == 2) {

item <- readline("Enter item name: ")

new\_quantity <- as.integer(readline("Enter new quantity: ")) update\_quantity(item, new\_quantity)

} else if (choice == 3) { display\_inventory()

} else if (choice == 4) {

cat("Exiting the program. Goodbye!\n") break

} else {

cat("Invalid choice. Please try again.\n")

}

}

Output:

1. Add Item
2. Update Quantity
3. Display Inventory
4. Exit

Enter your choice: 1 Enter item name: rice Enter quantity: 100

Item added to inventory.

1. Add Item
2. Update Quantity
3. Display Inventory
4. Exit

Enter your choice: 1 Enter item name: bicuits Enter quantity: 23

Item added to inventory.

Enter your choice:

**Lab5: Dataframe**

You are working as an educational analyst and need to analyze the performance of students in a school. You have data on student names, their scores in different subjects, and attendance. Write a program that uses data frames to manage and analyze student data, including calculating average scores, identifying students with low attendance, and generating a report.

Create a data frame to store student information with columns: "Name", "Math\_Score", "Science\_Score", "History\_Score", "Attendance".

Implement functions to:

* Calculate the average scores for each student.
* Identify students with attendance below a certain threshold.
* Generate a report with student names, average scores, and attendance status.
* Use the functions to analyse student performance and generate the report. Program:

# Load the 'dplyr' package for data manipulation library(dplyr)

# Create a data frame to store student information student\_data <- data.frame(

Name = character(0), Math\_Score = numeric(0), Science\_Score = numeric(0), History\_Score = numeric(0), Attendance = numeric(0)

)

# Function to add student information

add\_student <- function(name, math\_score, science\_score, history\_score, attendance) { new\_student <- data.frame(

Name = name, Math\_Score = math\_score,

Science\_Score = science\_score, History\_Score = history\_score, Attendance = attendance

)

student\_data <<- bind\_rows(student\_data, new\_student) cat("Student information added.\n")

}

# Function to calculate average scores calculate\_average\_scores <- function() { avg\_scores <- student\_data %>%

mutate(Average\_Score = (Math\_Score + Science\_Score + History\_Score) / 3) %>% select(Name, Average\_Score)

return(avg\_scores)

}

# Function to identify students with low attendance identify\_low\_attendance <- function(threshold) { low\_attendance <- student\_data %>%

Cilter(Attendance < threshold) %>% select(Name, Attendance) return(low\_attendance)

}

# Function to generate a performance report generate\_report <- function() {

avg\_scores <- calculate\_average\_scores() low\_attendance <- identify\_low\_attendance(70)

report <- merge(avg\_scores, low\_attendance, by = "Name", all = TRUE) report$Attendance[is.na(report$Attendance)] <- 100

cat("Performance Report:\n")

print(report)

}

# Main program while (TRUE) {

cat("\n1. Add Student\n2. Generate Report\n3. Exit\n") choice <- as.integer(readline("Enter your choice: "))

if (choice == 1) {

name <- readline("Enter student name: ")

math\_score <- as.numeric(readline("Enter math score: ")) science\_score <- as.numeric(readline("Enter science score: ")) history\_score <- as.numeric(readline("Enter history score: ")) attendance <- as.numeric(readline("Enter attendance percentage: "))

add\_student(name, math\_score, science\_score, history\_score, attendance)

} else if (choice == 2) { generate\_report()

} else if (choice == 3) {

cat("Exiting the program. Goodbye!\n") break

} else {

cat("Invalid choice. Please try again.\n")

}

}

### Explanation:

# Load the 'dplyr' package for data manipulation library(dplyr)

This line loads the 'dplyr' package, which provides a wide range of functions for data manipulation and transformation. It's used in this program for tasks like Ciltering, summarizing, and selecting data within data frames.

# Create a data frame to store student information student\_data <- data.frame(

Name = character(0), Math\_Score = numeric(0), Science\_Score = numeric(0), History\_Score = numeric(0), Attendance = numeric(0)

)

Here, a data frame named student\_data is created. It's initialized with empty columns for student names, math scores, science scores, history scores, and attendance. This data frame will hold all the student information.

# Function to add student information

add\_student <- function(name, math\_score, science\_score, history\_score, attendance) { new\_student <- data.frame(

Name = name, Math\_Score = math\_score,

Science\_Score = science\_score, History\_Score = history\_score, Attendance = attendance

)

student\_data <<- bind\_rows(student\_data, new\_student) cat("Student information added.\n")

}

This function add\_student adds a new student's information to the student\_data data frame. It creates a new data frame new\_student with the provided information and then uses the bind\_rows function from the 'dplyr' package to append this new student's data to the existing data frame.

# Function to calculate average scores calculate\_average\_scores <- function() { avg\_scores <- student\_data %>%

mutate(Average\_Score = (Math\_Score + Science\_Score + History\_Score) / 3) %>% select(Name, Average\_Score)

return(avg\_scores)

}

The calculate\_average\_scores function calculates the average scores for each student. It uses the

%>% (pipe) operator from the 'dplyr' package to perform operations sequentially. The mutate

function adds a new column Average\_Score calculated by averaging math, science, and history scores. The select function extracts only the "Name" and "Average\_Score" columns.

# Function to identify students with low attendance identify\_low\_attendance <- function(threshold) { low\_attendance <- student\_data %>%

Cilter(Attendance < threshold) %>% select(Name, Attendance) return(low\_attendance)

}

The identify\_low\_attendance function identiCies students with attendance below a speciCied threshold. It uses the Cilter function to extract rows where attendance is below the threshold, and the select function to extract the "Name" and "Attendance" columns.

# Function to generate a performance report generate\_report <- function() {

avg\_scores <- calculate\_average\_scores() low\_attendance <- identify\_low\_attendance(70)

report <- merge(avg\_scores, low\_attendance, by = "Name", all = TRUE) report$Attendance[is.na(report$Attendance)] <- 100

cat("Performance Report:\n") print(report)

}

The generate\_report function generates a performance report. It calls the calculate\_average\_scores and identify\_low\_attendance functions to get average scores and low attendance data. The merge function combines the data based on the "Name" column. It also handles missing attendance values by replacing them with 100 using is.na and assignment.

### Output:

1. Add Student
2. Generate Report
3. Exit

Enter your choice: 1

Enter student name: ramesh Enter math score: 67

Enter science score: 89 Enter history score: 89

Enter your choice:

**Lab 6 and 7: R functions for statistical analysis**

R provides a wide range of functions for statistical analysis. Here's an overview of some commonly used R packages and their functions for statistical analysis:

Base R:

mean(), median(), sd(): Calculate mean, median, and standard deviation. cor(), cov(): Compute correlation and covariance matrices.

t.test(): Perform t-tests for comparing means. wilcox.test(), kruskal.test(): Perform non-parametric tests. lm(): Fit linear regression models.

anova(), summary(): Perform analysis of variance and obtain model summaries. chisq.test(), Cisher.test(): Conduct tests for contingency tables.

dplyr Package:

Cilter(), select(), mutate(): Manipulate and transform data.

group\_by(), summarize(): Group data and calculate summary statistics. arrange(): Sort data based on variables.

join(): Merge data frames. ggplot2 Package:

ggplot(), geom\_point(), geom\_line(), etc.: Create sophisticated visualizations. facet\_wrap(), facet\_grid(): Create small multiples plots.

theme(), labs(): Customize plot appearance and labels. tidyr Package:

gather(), spread(): Convert data between wide and long formats. separate(), unite(): Split and combine variables.

stats Package:

pnorm(), qnorm(): Compute cumulative and quantile normal distribution probabilities. pf(), qf(): Compute cumulative and quantile F-distribution probabilities.

pchisq(), qchisq(): Compute cumulative and quantile chi-squared distribution probabilities. MASS Package:

Citdistr(): Fit distributions to data.

stepAIC(): Perform stepwise model selection. survival Package:

survCit(): Fit Kaplan-Meier survival curves. coxph(): Fit Cox proportional hazards models. caret Package:

train(): Train predictive models.

confusionMatrix(): Compute confusion matrices for classiCication models. forecast Package:

auto.arima(): Fit automatic ARIMA time series models. ets(), tbats(): Fit exponential smoothing models.

These are just a few examples of functions available for statistical analysis in R. Depending on your speciCic analysis needs, you might need to explore and use additional R packages and functions that cater to your domain and research interests.

### Lab 6 :

You are a data analyst at a retail company that sells products online. The company is interested in predicting sales for the upcoming months to better manage inventory and plan marketing strategies. As part of your role, you need to develop a program that utilizes time series analysis to forecast sales based on a historical sales dataset.

Write an R program to forecast sales for the next three months using time series analysis techniques. The program should perform the following steps:

* Load the required libraries, including the forecast package.
* Create a data frame with two columns: Month and Sales. The Month column should contain a sequence of dates from January 2023 to June 2023 (inclusive), and the Sales column should contain the corresponding sales amounts (12000, 15000, 18000, 16000, 20000, 22000).
* Convert the sales data into a time series object with a monthly frequency.
* Fit an ARIMA (AutoRegressive Integrated Moving Average) model to the sales time series using the auto.arima() function.
* Forecast sales for the next three months using the Citted ARIMA model and the forecast() function.
* Display the forecasted sales results, including point forecasts and prediction intervals. Code:

# Load required libraries library(forecast)

# Create a data frame with Month and Sales columns sales\_data <- data.frame(

Month = seq(as.Date("2023-01-01"), as.Date("2023-06-01"), by = "months"), Sales = c(12000, 15000, 18000, 16000, 20000, 22000)

)

# Convert to time series

sales\_ts <- ts(sales\_data$Sales, frequency = 12)

# Fit ARIMA model

arima\_model <- auto.arima(sales\_ts)

# Forecast sales for next 3 months forecast\_result <- forecast(arima\_model, h = 3)

# Display forecast results print(forecast\_result) Output:

print(forecast\_result)

Point Forecast Lo 80 Hi 80

Jul 1 22000 18285.70 25714.30

Aug 1 22000 16747.19 27252.81

Sep 1 22000 15566.65 28433.35

Lo 95 Hi 95

Jul 1 16319.47 27680.53

Aug 1 13966.52 30033.48

Sep 1 12161.04 31838.96

### Lab 8 : Customer Purchase Analysis for E-commerce Company (Enhanced)

You are a data analyst working for an e-commerce company that specializes in selling a variety of products online. The company aims to analyze customer purchase data comprehensively to gain insights into customer behavior and spending patterns.

Your goal is to develop a R program that performs an in-depth analysis of customer purchase data. You will calculate various statistical measures and generate visualizations to understand the distribution of purchase amounts among customers.

Note: Load the necessary libraries, including the dplyr and ggplot2 packages.

Given the example customer purchase data provided below, create a data frame named purchase\_data with two columns: CustomerID and PurchaseAmount.

Calculate and display the following statistical measures:

* Mean (average) purchase amount
* Median purchase amount
* Standard deviation of purchase amounts
* 1st quartile (25th percentile) of purchase amounts
* 3rd quartile (75th percentile) of purchase amounts

Create a histogram to visualize the distribution of purchase amounts using the ggplot2 package. Display the histogram with appropriate labels and titles.

Example Customer Purchase Data:

|  |  |
| --- | --- |
| CustomerID | PurchaseAmount |
| 101 | 150 |
| 102 | 200 |
| 103 | 120 |
| 104 | 300 |
| 105 | 80 |

### Solution:

# Load required libraries library(dplyr) library(ggplot2)

# Example customer purchase data purchase\_data <- data.frame(

CustomerID = c(101, 102, 103, 104, 105),

PurchaseAmount = c(150, 200, 120, 300, 80)

)

# Calculate statistical measures

mean\_purchase <- mean(purchase\_data$PurchaseAmount) median\_purchase <- median(purchase\_data$PurchaseAmount) sd\_purchase <- sd(purchase\_data$PurchaseAmount)

q1\_purchase <- quantile(purchase\_data$PurchaseAmount, probs = 0.25) q3\_purchase <- quantile(purchase\_data$PurchaseAmount, probs = 0.75)

# Display results

cat("Mean Purchase Amount:", mean\_purchase, "\n") cat("Median Purchase Amount:", median\_purchase, "\n")

cat("Standard Deviation of Purchase Amounts:", sd\_purchase, "\n") cat("1st Quartile of Purchase Amounts:", q1\_purchase, "\n") cat("3rd Quartile of Purchase Amounts:", q3\_purchase, "\n")

# Create a histogram

ggplot(purchase\_data, aes(x = PurchaseAmount)) + geom\_histogram(binwidth = 50, Cill = "blue", color = "black") +

labs(title = "Distribution of Purchase Amounts", x = "Purchase Amount", y = "Frequency")

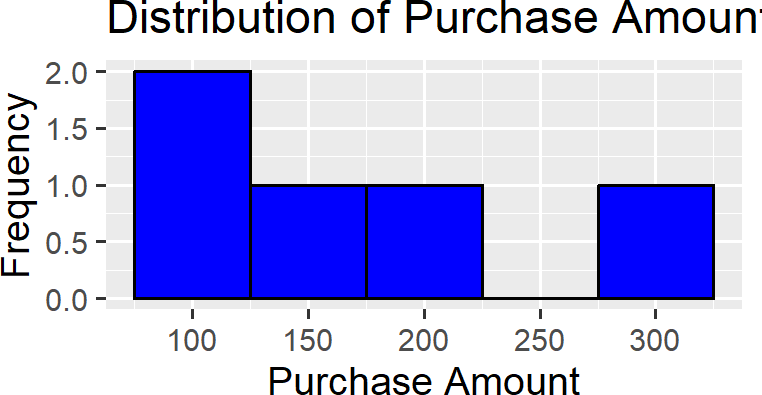
* # Calculate statistical measures
* mean\_purchase <- mean(purchase\_data$PurchaseAmount)
* median\_purchase <- median(purchase\_data$PurchaseAmount)
* sd\_purchase <- sd(purchase\_data$PurchaseAmount)
* q1\_purchase <- quantile(purchase\_data$PurchaseAmount, probs = 0.25)
* q3\_purchase <- quantile(purchase\_data$PurchaseAmount, probs = 0.75)

>

* # Display results
* cat("Mean Purchase Amount:", mean\_purchase, "\n") Mean Purchase Amount: 170
* cat("Median Purchase Amount:", median\_purchase, "\n") Median Purchase Amount: 150
* cat("Standard Deviation of Purchase Amounts:", sd\_purchase, "\n") Standard Deviation of Purchase Amounts: 84.85281
* cat("1st Quartile of Purchase Amounts:", q1\_purchase, "\n") 1st Quartile of Purchase Amounts: 120
* cat("3rd Quartile of Purchase Amounts:", q3\_purchase, "\n") 3rd Quartile of Purchase Amounts: 200

>

* # Create a histogram



### Lab 8: Matrix Manipulation in R

Write an R program that generates two matrices, matrix\_A and matrix\_B, and conducts operations including element-wise addition, scalar multiplication, matrix transpose, and multiplication.

### Code:

**# Task 1: Matrix Creation**

### matrix\_A <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3, ncol = 3, byrow = TRUE)

**matrix\_B <- matrix(c(9, 8, 7, 6, 5, 4, 3, 2, 1), nrow = 3, ncol = 3, byrow = TRUE)**

### # Task 2: Matrix Manipulation sum\_matrix <- matrix\_A + matrix\_B scaled\_matrix <- matrix\_A \* 2

**# Task 3: Matrix Operations transposed\_A <- t(matrix\_A)**

### product\_matrix <- matrix\_A %\*% matrix\_B

**# Task 4: Matrix Statistics sum\_matrix\_A <- sum(matrix\_A) mean\_matrix\_B <- mean(matrix\_B) sd\_matrix\_B <- sd(matrix\_B)**

### # Task 5: Visualization library(ggplot2) library(reshape2)

**# Create a heatmap of matrix\_A heatmap\_data <- melt(matrix\_A)**

### heatmap\_plot <- ggplot(heatmap\_data, aes(x = Var1, y = Var2, <ill = value)) + geom\_tile() +

**scale\_<ill\_gradient(low = "white", high = "blue") +**

### labs(title = "Heatmap of Matrix A", x = "Column", y = "Row")

**# Create a bar plot comparing sums of rows in matrix\_B row\_sums <- rowSums(matrix\_B)**

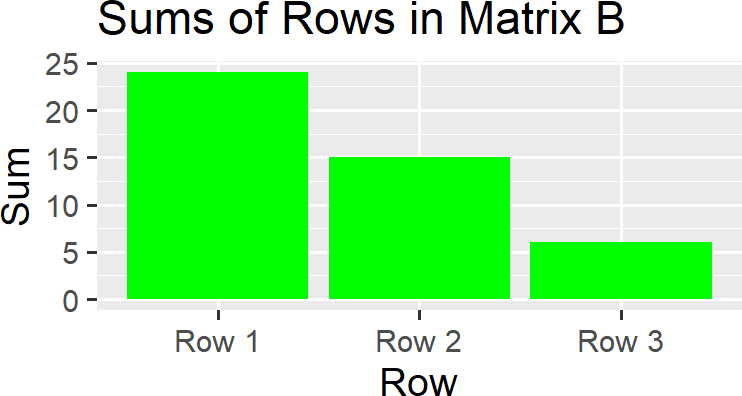
### row\_names <- paste("Row", 1:3)

**barplot\_data <- data.frame(Row = row\_names, Sum = row\_sums) barplot\_plot <- ggplot(barplot\_data, aes(x = Row, y = Sum)) + geom\_bar(stat = "identity", <ill = "green") +**

### labs(title = "Sums of Rows in Matrix B", x = "Row", y = "Sum")

**# Display the visualizations print(heatmap\_plot) print(barplot\_plot)**

### output:



**Lab 9 and 10: Visualization**

You are a data analyst tasked with analyzing and visualizing a dataset. The dataset contains information about student performance in a course. You need to create a program that generates various types of plots to help understand and present the data effectively.

Write a program that performs data analysis and generates visualizations for a given dataset. The program should:

1. Load the necessary libraries (ggplot2).
2. Prepare example data with columns for student names, scores, and attendance percentages.
3. Perform the following tasks:
   1. Create a scatter plot to visualize the relationship between scores and attendance percentages.
   2. Generate a bar plot to show the distribution of scores among different students.
   3. Create a line plot to display the trend of scores over time (assuming different students' scores were collected at different time intervals).
   4. Generate a histogram to visualize the distribution of scores.
4. Customize the appearance of each plot, such as color, labels, and titles.
5. Arrange the plots in a way that they are easy to compare and understand.
6. Provide appropriate titles for each plot and the axes.

### Scatter Plot:

library(ggplot2)

# Example data

data <- data.frame(x = rnorm(50), y = rnorm(50))

# Scatter plot ggplot(data, aes(x, y)) + geom\_point() +

labs(title = "Scatter Plot", x = "X-axis", y = "Y-axis")

Bar Plot:

library(ggplot2) # Example data

data <- data.frame(category = c("A", "B", "C"), value = c(10, 20, 15)) # Bar plot

ggplot(data, aes(x = category, y = value)) + geom\_bar(stat = "identity") +

labs(title = "Bar Plot", x = "Category", y = "Value") Line Plot:

library(ggplot2) # Example data

data <- data.frame(x = 1:10, y = 2 \* (1:10)) # Line plot

ggplot(data, aes(x, y)) +

geom\_line() +

labs(title = "Line Plot", x = "X-axis", y = "Y-axis") Histogram:

library(ggplot2) # Example data

data <- data.frame(values = rnorm(100)) # Histogram

ggplot(data, aes(values)) +

geom\_histogram(binwidth = 0.5, Cill = "blue", color = "black") + labs(title = "Histogram", x = "Values", y = "Frequency")

## scatter, bar, line, histogram, and box plots

# Load required libraries library(ggplot2) library(patchwork) library(plotly)

# Example data set.seed(123)

data <- data.frame(

Date = seq(as.Date("2023-01-01"), as.Date("2023-12-31"), by = "months"), Sales = cumsum(sample(5000:10000, 12, replace = TRUE)),

Engagement = sample(1:10, 12, replace = TRUE),

Category = rep(c("Electronics", "Clothing", "Books"), each = 4)

)

# Scatter plot

scatter\_plot <- ggplot(data, aes(x = Sales, y = Engagement, color = Category)) + geom\_point() +

labs(title = "Sales vs. Customer Engagement", x = "Sales", y = "Engagement")

# Bar plot for sales by category

bar\_plot <- ggplot(data, aes(x = Category, y = Sales, Cill = Category)) +

geom\_bar(stat = "identity") +

labs(title = "Sales by Category", x = "Category", y = "Sales") + theme(axis.text.x = element\_text(angle = 45, hjust = 1))

# Line plot for sales over time

line\_plot <- ggplot(data, aes(x = Date, y = Sales, color = Category)) + geom\_line() +

labs(title = "Sales Trend Over Time", x = "Date", y = "Sales")

# Histogram for customer engagement histogram\_plot <- ggplot(data, aes(Engagement)) +

geom\_histogram(binwidth = 1, Cill = "green", color = "black") +

labs(title = "Customer Engagement Distribution", x = "Engagement", y = "Frequency")

# Box plot for sales by category

box\_plot <- ggplot(data, aes(Category, Sales, Cill = Category)) + geom\_boxplot() +

labs(title = "Sales Distribution by Category", x = "Category", y = "Sales")

# Combine plots using patchwork combined\_plots <- scatter\_plot + bar\_plot +

line\_plot + histogram\_plot + box\_plot

# Display combined plots print(combined\_plots)

PIE chart in R:

# Load required libraries library(ggplot2)

# Example data data <- data.frame(

Category = c("Electronics", "Clothing", "Books"), Sales = c(25000, 18000, 12000)

)

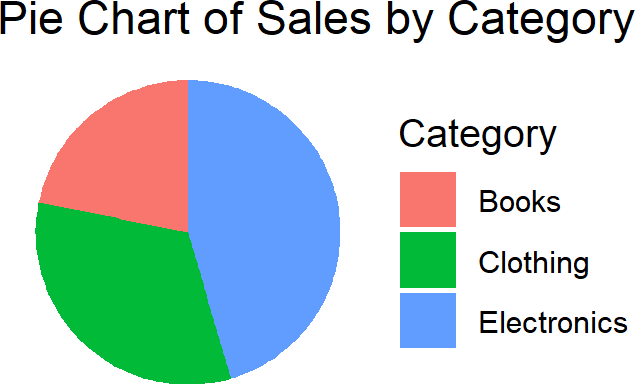
# Create a pie chart

pie\_chart <- ggplot(data, aes(x = "", y = Sales, Cill = Category)) + geom\_bar(stat = "identity", width = 1) +

coord\_polar("y", start = 0) + # Convert to polar coordinates labs(title = "Pie Chart of Sales by Category") +

theme\_void() # Remove axes and labels

# Display the pie chart print(pie\_chart)



LAB11: Data Analysis: Statement: Customer Purchase Analysis

You are a data analyst at an e-commerce company that sells a variety of products online. The company has provided you with a dataset containing information about customer purchases. Your task is to perform a comprehensive data analysis to gain insights into customer behavior and spending patterns.

Dataset Description:

The dataset customer\_purchases.csv contains the following columns: CustomerID: Unique identiCier for each customer.

PurchaseAmount: The amount spent by the customer on a purchase. Problem Tasks:

You are required to perform the following tasks using R:

Task 1: Load the Dataset

Load the necessary libraries, including readr and dplyr.

Read the dataset customer\_purchases.csv into a data frame named purchase\_data. Task 2: Data Summary

Calculate and display the total number of records in the dataset.

Calculate and display the total number of unique customers in the dataset. Task 3: Calculate Statistical Measures

Calculate and display the mean (average) purchase amount. Calculate and display the median purchase amount.

Calculate and display the standard deviation of purchase amounts. Task 4: Customer Segmentation

Create a new column named Segment in the purchase\_data data frame based on the following criteria:

"Low Spender" if the purchase amount is less than the median.

"High Spender" if the purchase amount is greater than or equal to the median. Task 5: Visualize Data

Create a histogram to visualize the distribution of purchase amounts using the ggplot2 package. Customize the plot with appropriate labels and titles.

Code:

# Load required libraries library(readr) library(dplyr) library(ggplot2)

# Task 1: Load the Dataset

purchase\_data <- read\_csv("D:/RPrograms2023/LAb/customer\_purchases.csv") # Load required libraries

library(readr) library(dplyr) library(ggplot2)

# Task 1: Load the Dataset # Task 1: Load the Dataset

purchase\_data <- read\_csv("D:/RPrograms2023/LAb/customer\_purchases.csv")

# Task 2: Data Summary total\_records <- nrow(purchase\_data)

total\_customers <- n\_distinct(purchase\_data$CustomerID)

cat("Total Number of Records:", total\_records, "\n")

cat("Total Number of Unique Customers:", total\_customers, "\n")

# Task 3: Calculate Statistical Measures

mean\_purchase <- mean(purchase\_data$PurchaseAmount) median\_purchase <- median(purchase\_data$PurchaseAmount) sd\_purchase <- sd(purchase\_data$PurchaseAmount)

cat("Mean Purchase Amount:", mean\_purchase, "\n") cat("Median Purchase Amount:", median\_purchase, "\n")

cat("Standard Deviation of Purchase Amounts:", sd\_purchase, "\n")

# Task 4: Customer Segmentation (Based on Quartiles)

q1\_purchase <- quantile(purchase\_data$PurchaseAmount, probs = 0.25) q2\_purchase <- quantile(purchase\_data$PurchaseAmount, probs = 0.5) q3\_purchase <- quantile(purchase\_data$PurchaseAmount, probs = 0.75)

purchase\_data <- purchase\_data %>% mutate(Segment = case\_when(

PurchaseAmount < q1\_purchase ~ "Low Spender",

PurchaseAmount >= q1\_purchase & PurchaseAmount < q3\_purchase ~ "Medium Spender", PurchaseAmount >= q3\_purchase ~ "High Spender"

))

# Task 5: Visualize Data (Histogram) ggplot(purchase\_data, aes(x = PurchaseAmount)) +

geom\_histogram(binwidth = 50, Cill = "blue", color = "black") +

labs(title = "Distribution of Purchase Amounts", x = "Purchase Amount", y = "Frequency")

# Task 6: Visualize Relationship (Scatter Plot) ggplot(purchase\_data, aes(x = CustomerID, y = PurchaseAmount)) + geom\_point(color = "green") +

labs(title = "Customer Purchase Amounts", x = "Customer ID", y = "Purchase Amount")

# Save the visualizations as image Ciles ggsave("purchase\_histogram.png", width = 8, height = 6) ggsave("customer\_purchase\_scatter.png", width = 8, height = 6)

LAB12: Data Analysis:

You are a data analyst tasked with performing an Exploratory Data Analysis (EDA) on the Indian Premier League (IPL) dataset. The IPL dataset contains information about various IPL matches, including match dates, teams, venues, outcomes, and performance metrics. Your objective is to gain insights into the dataset by conducting an in-depth analysis using R programming.

Dataset Description:

The dataset named "ipl\_data.csv" includes the following columns:

Match\_ID: Unique identiCier for each match. Date: Date of the match.

Team1: Name of the Cirst team participating in the match. Team2: Name of the second team participating in the match. Venue: Stadium where the match was played.

Winner: Name of the winning team. Total.Runs: Total runs scored in the match.

Total.Wickets: Total wickets taken in the match. Other relevant columns (if any).

Problem Tasks:

Your task is to perform the following Exploratory Data Analysis (EDA) tasks using R:

Task 1: Data Overview and Structure

Display the structure of the dataset using str() function.

Output summary statistics of numerical columns using summary(). Task 2: Basic Data Insights

Calculate and display the total number of matches in the dataset.

Determine the number of unique teams that have participated in IPL matches. List the unique teams from both Team1 and Team2.

Task 3: Team Performance Analysis

Calculate the number of matches won by each team and display the results. Compute the average total runs scored in the matches.

Calculate the average total wickets taken in the matches. Task 4: Venue Insights

Identify and display the most frequently used venue for matches. Task 5: Visualization

Create a bar plot to visualize the number of matches won by each team.