





Indira Gandhi National Open University
School of Computer & Information Sciences

MCSL-070
DATA ANALYSIS
LAB

DATA ANALYSIS LAB

**SECTION 1 DATA WRANGLING AND
VISUALISATION**

SECTION 2 PREDICTIVE DATA ANALYSIS

PROGRAMME DESIGN COMMITTEE

Prof. D. K. Lobiyal, JNU, New Delhi
Prof. Dinesh Kumar Vishwakarma, DTU, New Delhi

Prof. Sandeep Singh Rawat, SOCIS, IGNOU
Prof. P.V. Suresh, SOCIS, IGNOU
Prof. V.V. Subrahmanyam, SOCIS, IGNOU
Prof. Divakar Yadav, SOCIS, IGNOU
Dr. Akshay Kumar, Associate Professor, SOCIS
IGNOU
Dr. M.P. Mishra, Associate Professor,
SOCIS IGNOU
Dr. Sudhansh Sharma, Assistant Professor, SOCIS
IGNOU

COURSE DESIGN COMMITTEE

Prof. D. K. Lobiyal, JNU, New Delhi
Prof. Dinesh Kumar Vishwakarma, DTU, New Delhi
Prof. T. V. Vijay Kumar, JNU, New Delhi
Prof. D. P. Vidyarthi, JNU, New Delhi
Prof. Vivek Kumar Singh, University of Delhi, Delhi
Prof. A. K. Mohapatra, IGDUTW, New Delhi
Mr. Kapil Pandey, HCL Technologies Limited, Delhi – NCR

Prof. Manish Trivedi, SOS (Statistics), IGNOU

Dr. Rajesh Kaliraman, Assistant Professor,
SOS (Statistics), IGNOU
Dr. Prabhat Kumar Sangal, Assistant Professor,
SOS (Statistics), IGNOU

Prof. Sandeep Singh Rawat, SOCIS, IGNOU
Prof. P.V. Suresh, SOCIS, IGNOU
Prof. V.V. Subrahmanyam, SOCIS, IGNOU
Prof. Divakar Yadav, SOCIS, IGNOU
Dr. Akshay Kumar, Associate Professor,
SOCIS IGNOU
Dr.M.P.Mishra, Associate Professor, SOCIS
IGNOU
Dr. Sudhansh Sharma, Assistant Professor, SOCIS
IGNOU

SOCIS FACULTY

Prof. Sandeep Singh Rawat, Director, SOCIS, IGNOU
Prof. V.V. Subrahmanyam, SOCIS, IGNOU
Prof. Akshay Kumar, SOCIS, IGNOU
Dr. Sudhansh Sharma, Associate Professor, SOCIS, IGNOU

Prof. P.V. Suresh, SOCIS, IGNOU
Prof. Divakar Yadav, SOCIS, IGNOU
Prof. M.P. Mishra, SOCIS, IGNOU

COURSE PREPARATION TEAM

Content Writer (Section-1)

(Partially adapted from MCSL205)
Prof. Akshay Kumar (Co-author Section-1)
SOCIS, IGNOU

Content Editor

Section:1: Dr. Sudhansh Sharma
SOCIS-IGNOU

Content Writer (Section-2)

(Partially adapted from MCSL229)
Dr. Sudhansh Sharma (Co-author Section-2),
Associate Professor, SOCIS, IGNOU

Content Editor

Section 2: Prof. Akshay Kumar
SOCIS-IGNOU

Course Coordinator

Prof. Akshay Kumar & Dr. Sudhansh Sharma
SOCIS, IGNOU SOCIS, IGNOU

PRINT PRODUCTION

Sh. Sanjay Aggarwal
Assistant Registrar, MPDD, IGNOU, New Delhi

May, 2026

©Indira Gandhi National Open University, 2026

ISBN – xx-xxxx-xxx-x

All rights reserved. No part of this work may be reproduced in any form, by mimeography or any other means, without permission in writing from the Indira Gandhi National Open University.

Further information on the Indira Gandhi National Open University courses may be obtained from the University's office at Maidan Garhi, New Delhi 110068.

CRC prepared in-house at SOCIS by Mr. Vishal Singh (Skilled Worker)

Printed and published on behalf of the Indira Gandhi National Open University, New Delhi by MPDD, IGNOU.

COURSE INTRODUCTION

The Lab course *MCSL-070: Data Analysis Lab* is designed as a comprehensive practical companion to the theoretical courses *MCS-067: Data Wrangling and Visualisation* and *MCS-068: Predictive Data Analysis*. It aims to bridge the gap between conceptual understanding and real-world implementation by providing hands-on experience in data handling, analysis, and interpretation using two of the most widely used programming languages in analytics—Python and R. The course is structured into two major sections, each focusing on a specific aspect of data analysis and comprising ten lab sessions that guide learners through practical problem-solving approaches.

Section 1: Data Wrangling and Visualisation (Use Python Programming Language) focuses on developing strong foundational skills in preparing and exploring data using Python. This section is aligned with the content of MCS-067 and emphasises the practical implementation of data preprocessing techniques such as data cleaning, handling missing values, transformation, and integration of datasets. Learners gain hands-on exposure to operations like filtering, grouping, aggregation, and reshaping of data, which are essential for making raw data analysis-ready. In addition, the section introduces powerful visualisation libraries such as matplotlib, pandas, and seaborn to create meaningful graphical representations, including line plots, bar charts, histograms, and scatter plots. The lab sessions progressively build the learner's ability to interpret patterns, trends, and relationships in data, thereby enhancing exploratory data analysis skills. It also incorporates advanced visualisation concepts such as multivariate analysis, graphical design principles, and visualisation of statistical models, ensuring a well-rounded practical understanding of data representation.

Section-2: Predictive Data Analysis (R Programming) is centered on building analytical and predictive modeling capabilities using R. Based on the syllabus of MCS-068, this section provides practical exposure to statistical techniques and machine learning models used for predictive analytics. Through ten structured lab sessions, learners work with similarity measures, outlier detection techniques, regression models, time series forecasting methods, and both supervised and unsupervised learning algorithms. The section also covers performance evaluation metrics such as accuracy, precision, recall, and ROC curves, enabling learners to assess model effectiveness. Additionally, students develop proficiency in R programming, including data handling, visualization, and implementation of statistical tests and advanced analytics. This section equips learners with the ability to derive insights, make predictions, and support decision-making processes based on data-driven approaches.

The significance of *MCSL-070* lies in its strong practical orientation, which is highly valuable in both academic and industry contexts. Academically, it enhances conceptual clarity by allowing learners to experiment with real datasets and apply theoretical knowledge in controlled environments. It fosters analytical thinking, problem-solving abilities, and research skills essential for higher studies and academic projects. From an industry perspective, the course aligns with the growing demand for data professionals skilled in Python and R. It prepares learners for roles such as data analyst, data scientist, and business analyst by equipping them with practical skills in data preprocessing, visualisation, and predictive modelling. The lab-based approach ensures that learners are job-ready, capable of handling real-world data challenges, and able to contribute effectively to data-driven organisations.

Overall, *MCSL-070: Data Analysis Lab* serves as a crucial link between theory and practice, empowering learners with the technical competence and analytical mindset required in today's data-centric world.

BLOCK INTRODUCTION

This lab manual has been designed to help learners understand data analysis through practical learning. Instead of only studying theory, this course allows you to actually work with data using programming tools. It is structured as a self-learning experience, where you gradually build your skills through hands-on lab exercises.

The manual follows a *session-wise structure* encompassing a broad range of practices in data science and analysis—from Data wrangling & visualisation to Predictive Data Analysis, as per industry-standard platforms. The lab sessions are organised in progressive phases, each dedicated to a specific toolset.

The course is divided into two sections, and each section contains 10 lab sessions that are directly based on the concepts learned in the courses *MCS-067: Data Wrangling and Visualisation* and *MCS-068: Predictive Data Analysis*, respectively. This makes the learning process more interactive, applied, and easy to understand.

This block consists of 2 Sections of 10 Sessions each, their description is as follows:

Section 1: *Data Wrangling and Visualisation*, comprises 10 sessions and is based on Python Programming. The sessions are focused on learning how to prepare and understand data using Python. In real life, data is often messy and incomplete, so the first step is to clean and organise it. Through the lab sessions, you will learn how to handle missing values, remove duplicates, transform data, and combine multiple datasets. You will also practice grouping and summarising data to extract useful information. Along with this, the section introduces data visualisation techniques using tools like matplotlib and seaborn. You will create graphs, such as bar charts, line plots, histograms, and scatter plots, to better understand patterns and trends in the data. These activities help you develop the ability to explore data and present it meaningfully.

Section 2: *Predictive Data Analysis*, comprises 10 sessions, and they are based on R programming. These sessions help you move one step ahead, from understanding data to making predictions. This section uses R programming and focuses on applying statistical and machine learning techniques. Through the 10 lab sessions, you will learn how to build models such as regression, classification, and clustering. You will also explore time series analysis and forecasting methods. The labs guide you in detecting outliers, measuring similarity, and evaluating model performance using metrics like accuracy and precision. By practising these techniques, you will learn how to make data-driven decisions and predictions based on real-world datasets.

This lab manual is more than a set of exercises; it is a step-by-step journey through the practical landscape of data science tools and practices. By following the prescribed structure and adhering to the general guidelines, students will not only enhance their technical proficiency but also develop a disciplined approach to analytical problem-solving skills highly valued in both academic and professional domains.

SECTION 1: DATA WRANGLING AND VISUALISATION LAB

Structure

Page No.

- 1.0 Introduction
- 1.1 Objectives
- 1.2 General Guidelines
- 1.3 Salient Features of Python
- 1.4 Working with Python
- 1.5 Running Python Programs
- 1.6 List of Session-wise Problems
- 1.7 Summary

1.0 INTRODUCTION

This section of the lab course provides a hands-on experience on the use of data wrangling and visualisation techniques on sets of data. The tool used for this course is the Python programming language with pandas and matplotlib libraries, which are part of the Anaconda distribution. If you are new to Python programming, then you may study the supporting course content of MCS-201 Programming in C & Python. You may also use the Google Colab environment. This section provides a brief input on these environments. A session-wise list of practical problems is also provided. You may please read the general guidelines and the program documentation also.

1.1 OBJECTIVES

After implementation of the problems given in this section of the lab course, you will be able to:

- use Python libraries for several data wrangling and visualisation tasks;
- create and manipulate dataframes using pandas from .csv files;
- perform some basic descriptive analysis on data using pandas;
- identify missing values and handle data having missing values;
- replace abbreviations and multiple representations in data with consistent data values;
- convert numeric data to categories;
- split and join the datasets, removing duplicates;
- be able to write some simple output on the screen as well as in the files;
- use different types of join operations on two data sets;
- create hierarchical indexing on the datasets;
- perform grouping of data based on some criteria;
- enhance different kinds of plots by adding visual features;
- draw different visualisations like bar chart, line chart, histograms etc.; and
- draw plots relating to decision trees, density estimation, regression, etc.

1.2 GENERAL GUIDELINES

- Python is a high-level, general-purpose, interpreted language. You should attempt all the problems given in the session-wise list. In addition, you need to complete all the questions asked in the assignment of this course.
- You may seek assistance in doing the lab exercises from the concerned lab instructor.
- Since the assignments have credits, the lab instructor is not expected to tell you how to solve assignment questions, but you may ask questions concerning the Python language or a technical problem.
- You should write comments in each program. Also, write the algorithm prior to writing the program. Also, write comments on the purpose and parameters of the function written by you. Also, give details of the value returned by a function.
- You should implement a function with proper documentation and data Input and Output.
- It is your responsibility to create a separate directory to store all the programs so that no one else can read or copy them.
- You should create a proper practical book with duly listed programs that you have completed in a lab session.
- The session-wise list of problems is available in this lab manual. For each session, you must come prepared with the algorithms and the programs written in the practical book. You should utilise the lab hours for executing the programs, testing for various desired outputs and enhancements of the programs.
- As soon as you have finished a lab exercise, contact one of the lab instructor / in-charge in order to get the exercise evaluated and also get the signature from him/her on the practical book.
- The total number of 10 lab sessions. Each lab session is of 3 hours duration. Make the best use of these lab sessions by following the guidelines as suggested in this section.

1.3 SALIENT FEATURE OF PYTHON

Python is a dynamically typed language that supports garbage collection. It supports multiple programming paradigms, including structured, object-oriented, and functional programming. It has a very comprehensive standard library. Some of the features of Python are listed in the following table:

Metrics	Python Characteristics
<i>Introduction</i>	It is a high-level, general-purpose and interpreted programming language.

<i>Speed</i>	Being Interpreted programming language its execution speed is slower than that of the compiled programming languages.
<i>Usage</i>	Number of lines of code written in Python is quite less in comparison to other typed programming languages.
<i>Declaration of variables</i>	Declaration of Variable type is not required, they are untyped. A variable can contain different types of values at different instances during the execution of a Python program.
<i>Error Debugging</i>	Error debugging is simple, as it takes only one instruction at a time. Errors are instantly shown, and execution stops at that instruction only.
<i>Function renaming mechanism</i>	Same function can be used by two different names i.e. it supports the mechanism of function renaming.
<i>Complexity</i>	Syntax of Python programs is Quite simple, easy to read, write and learn.
<i>Memory-management</i>	It supports automatic garbage collection for memory management.
<i>Applications</i>	Python is a General-Purpose programming language can be used for web design, AI, ML etc.
<i>Built-in functions</i>	Library of built-in functions is quite large in Python.
<i>Implementing Data Structures</i>	You can implement data structures easily, as Python has built-in insert, append functions.
<i>Pointers</i>	Pointers functionality is not available in Python.

1.4 WORKING WITH PYTHON

Now, we have a bit of clarity about Python as a programming language. Python has a rich set of reusable libraries of modules. Each of these modules has been designed for implementing a specific task and can be integrated into a program. It also supports a set of frameworks for application development. Some of the important libraries and frameworks are listed below:

Name of Library	Purpose
NumPy	Numerical Computing Using Array Objects
Pandas	Data Frame or tabular data manipulation and analysis
Matplotlib	Powerful Visualisations
Seaborn	High level interface for statistical graphics
Tkinter	Simple Desktop application development
OpenCV	Computer Vision and Image Processing

Name of Framework	Purpose
Django	Secure and scalable web applications
Flask	For the development of Small web applications
Scikit-learn	Classical Machine Learning
TensorFlow	Deep Learning models
PyTorch	Graph based research models
Keras	Deep Learning API

In addition, we need to work on the Integrated Development Environments (IDEs) for Python. There are many IDEs like Jupyter Notebook, Spyder, VS Code, R Programming, etc., all of which are collectively available in Anaconda. Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing. It includes application development tools for data science, machine learning, large-scale data processing and predictive analytics. You may also go for the cloud versions like Google Colab Notebook, where you need not have high configuration hardware; only internet is required and through your Gmail account you can work on Python using Jupyter notebook.

We will discuss in brief some of the ways to work with Python, you may choose any or try all and other options too.

- 1) Just browse for <https://www.python.org> and perform the following steps:
 - a) Download the latest version of Python for the operating system installed on your computer. For example, you may download the latest Python Install Manager 26.1 or Windows Installer (64-bit) for Python version 3.13.12.
 - b) Now Run the downloaded file and install the Python, till the setup installation is finished
 - c) Finally, you are having an interface for Python programming
- 2) To install Anaconda, you may visit <https://www.anaconda.com/> and perform the following steps:
 - a) Click the Download button on the webpage of <https://www.anaconda.com/>
 - b) The distribution section <https://www.anaconda.com/distribution/> will open,
 - c) Click the download option given on this page <https://www.anaconda.com/distribution/>
 - d) <https://www.anaconda.com/distribution/#download-section> will open, offering 64-bit and 32-bit installers.
 - e) The Anaconda setup file will be downloaded.
 - f) Now, just run the interactive setup file until the installation is complete.

- g) Finally, you will find the Anaconda Navigator shortcut on your desktop. Click it, and you can start working with any IDE, be it Jupyter Notebook, Spyder, VS Code, etc., or even with R Programming.

Important: Before working with IDEs, you need to understand how to work with them and which one is suitable. The following are some of the observations:

- a) When you start Jupyter Notebook on windows (by clicking on the jupyter section, given in the Anaconda Navigator), a browser will open in internet explorer, many a times Jupyter Notebook won't work here, then just copy the URL from the Internet Explorer and paste it in the Google Chrome, you will find that Jupyter Notebook starts working, other details regarding the writing, saving, execution of program, will be discussed later.
- b) Program execution in Jupyter is line by line, and in Spyder, it goes sideways; even errors can also be seen sideways. Anyway, all these software are good to work with Python.

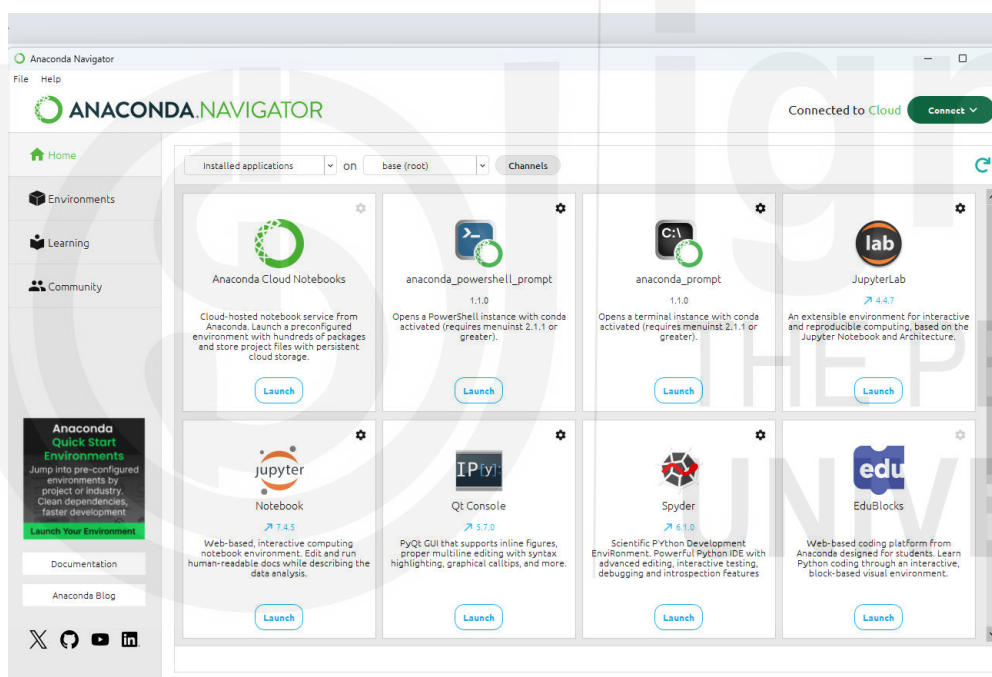


Figure 1: Anaconda Navigator

- 3) Often, learners may not have a computer system with the latest hardware configuration required for installing Python, or there may be compatibility issues with the operating system, or, for any reason, you may not be able to install and start working with Python. Under such circumstances, the solution is Google Colab Notebook (<https://colab.research.google.com/notebooks/welcome.ipynb>), use this and just log in with your Gmail account and start your work on Jupyter Notebook, as simple as that.

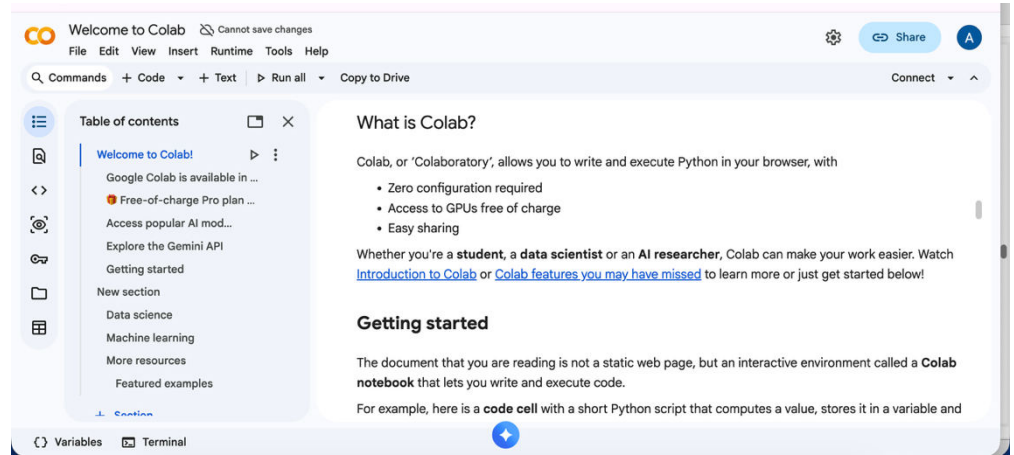


Figure 2: Colab Notebook using Browser

1.5 RUNNING PYTHON PROGRAMS

Just click the file option and select “New Notebook in Drive”, and a new Jupyter notebook will open in Google Colab. Now you may start your work.

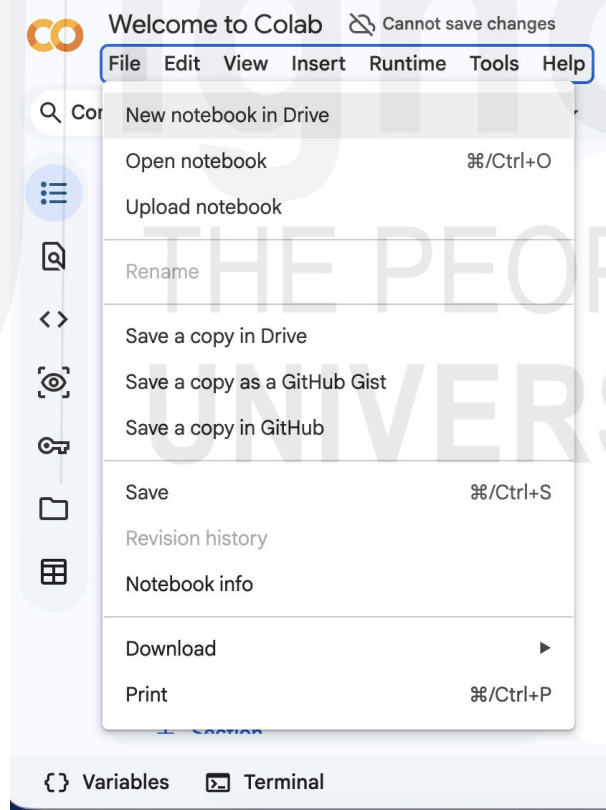
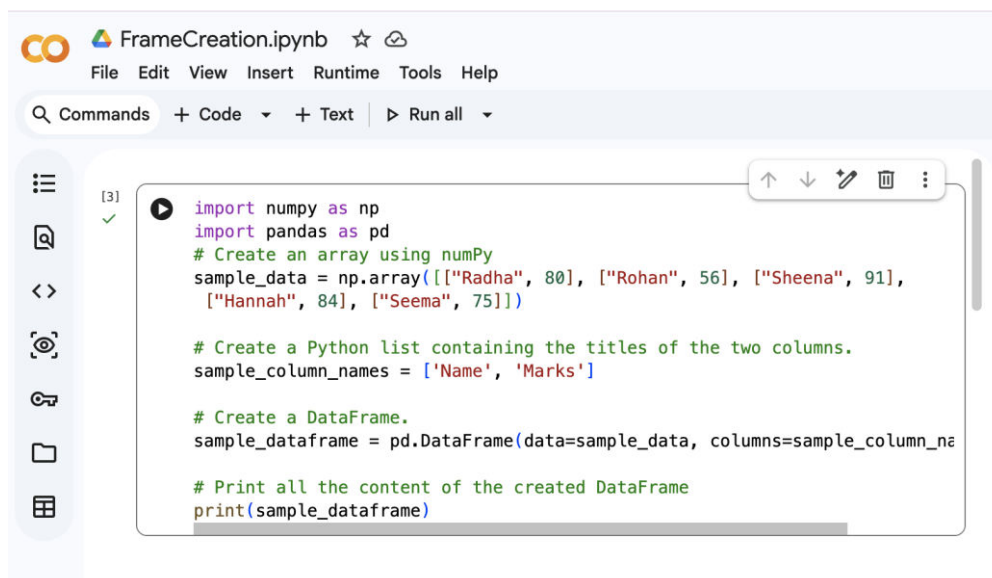


Figure 3: The File Menu

A new file will open, and you can write your Python Program in this file. As shown in Figure 4, we have written a simple program to create a data frame using NumPy and the pandas library in Python. Also, note that the file has been renamed to *FrameCreation.ipynb*.



```

import numpy as np
import pandas as pd
# Create an array using numPy
sample_data = np.array([["Radha", 80], ["Rohan", 56], ["Sheena", 91],
                        ["Hannah", 84], ["Seema", 75]])

# Create a Python list containing the titles of the two columns.
sample_column_names = ['Name', 'Marks']

# Create a DataFrame.
sample_dataframe = pd.DataFrame(data=sample_data, columns=sample_column_names)

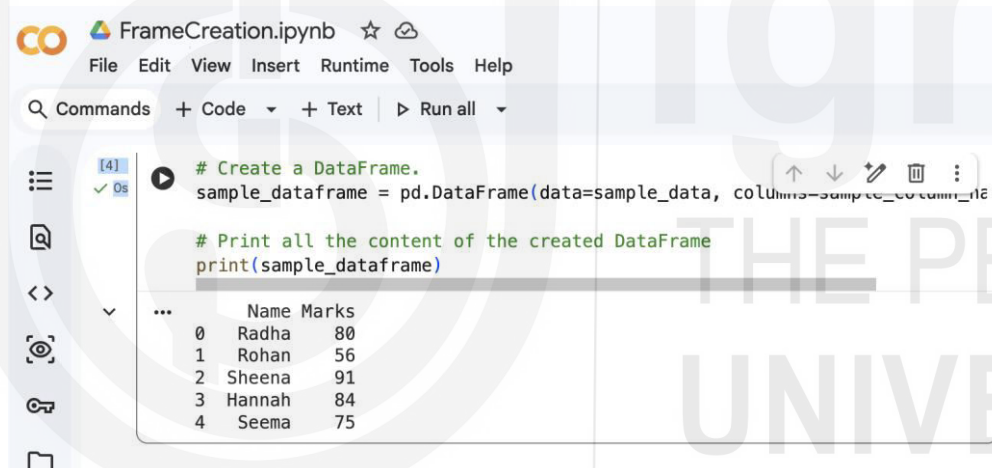
# Print all the content of the created DataFrame
print(sample_dataframe)

```

Figure 4: A Sample program to create a data frame

You can run this program using either the Run all Options or the  button.

The dataframe will be printed, as shown in Figure 5.



```

# Create a DataFrame.
sample_dataframe = pd.DataFrame(data=sample_data, columns=sample_column_names)

# Print all the content of the created DataFrame
print(sample_dataframe)

```

	Name	Marks
0	Radha	80
1	Rohan	56
2	Sheena	91
3	Hannah	84
4	Seema	75

Figure 5: The output of the sample program

1.6 SESSION-WISE LIST OF LAB PROBLEMS

You should solve the following problems using Python for data wrangling and visualisation.

Session 1 (Covers Unit 1 of MCS-67):

- Q1: Create a CSV file from the following dataset. Perform the following tasks with this dataset using Python code:

ID	Name	Age	Gender	Marks
1001	Alisha	20	F	78
1002	Ravi	21	M	67
1003	Neha	19	F	90
1004	Salim	22	M	45
1005	Priya	20	F	88

- Load the dataset as a DataFrame using pandas
- Display the first 2 rows of the DataFrame
- Display the last 3 rows of the DataFrame
- Display the dimensions of the data sets (Hint: Use the shape function)
- Display the names of different columns

Q2: Create the following dataset as a CSV file. Load the dataset into a pandas DataFrame and perform the following tasks with this dataset using Python code:

Dataset: employees.csv

Emp_ID	Name	Salary	Joining_Date	Dept
101	John	40000	2023-01-15	IT
102	Sarita	60000	2025-06-10	IT
103	Amit	55000	2024-03-20	Finance
104	Salim	45000	2026-04-10	Finance

- Find the datatype of each column.
- Convert the data type of the “Joining_Date” column to datetime and verify this data type
- Find the mean, median, variance and standard deviation of the Salary data.
- Assuming the salary column shows annual salaries, find the total salary paid to all employees.

Q3: Create the following dataset as a CSV file. Load the dataset into a pandas DataFrame and perform the following set-based profiling with this dataset using Python code:

Dataset: orders.csv

Order_ID	Customer	Region	Product
O001	ABC Pvt Ltd	North	Laptop
O002	XYZ Company	South	Phone
O003	XYZ Company	North	Laptop
O004	ABC Pvt Ltd	East	Tablet

- Which of the columns contains the categorical data?

- b) Find unique values of each categorical variable and count the frequency of each category.
- c) List the most frequent regions using Python.

Q4: Create the following dataset as a CSV file. Load the dataset into a pandas DataFrame and perform the following missing value analysis with this dataset using Python code:

Dataset: marks.csv

Student	Math	Science	English
Alisha	78	85	82
Ravi	NaN	72	70
Neha	90	NaN	89
Salim	45	55	NaN

- a) Identify missing values in each column.
- b) Compute the percentage of missing values per column.
- c) Identify the row with the maximum missing values.

Q5: Create the following dataset as a CSV file. Load the dataset into a pandas DataFrame and perform the following distribution analysis with this dataset using Python code:

Dataset: salary.csv

Emp_Name	Salary
Alisha	35000
Ravi	32000
Neha	42000
Salim	43000
Priya	100000

- a) Find the mean, median, and mode of employees' salaries.
- b) Is there any skewness in the data? Give reasons in support of your answer.
- c) Use descriptive statistics to find the outliers in the data.

Q6: Create the following dataset as a CSV file. Load the dataset into a pandas DataFrame and perform the following analysis to find inconsistent data on this dataset using Python code:

Dataset: city_data.csv

Person_Name	City
Alisha	Delhi
Ravi	delhi
Neha	DELHI
Salim	Mumbai
Priya	mumbai

- Find all the inconsistent entries.
- Standardise and convert all city names to uppercase.
- Now, count the names of unique cities.

Q7: Create the following two datasets as CSV files. Load the datasets into two different pandas DataFrames and perform the following tasks to compare the two datasets using Python code:

Dataset 1: employeeSet1.csv

Emp_Name	Salary
Alisha	35000
Ravi	32000
Neha	42000

Dataset 2: employeeSet2.csv

Emp_Name	Salary
Neha	42000
Salim	43000
Priya	100000

- Find the rows that are common between Dataset 1 and Dataset 2.
- Find the rows that are present in Dataset 1 and not in Dataset 2; also, find the rows that are present in Dataset 2 but not in Dataset 1.

Q8: Create the following dataset as CSV files. Load the datasets into a pandas DataFrame and perform the following tasks dealing with correlation and redundancy using Python code:

Dataset: answers.csv

Answers by Student A	Answers by Student B	Answers by Student C
1	2	2
2	4	4
3	3	3
4	1	1

- Compute the correlation matrix for the data given above.
- List the highly correlated columns having a correlation of close to 1. Find a group of students with almost similar answers. Give reasons in support of your answer.

Session 2 (Covers Unit 2 of MCS-67):

Q9: Create a dataset named `employee_attendance.csv` that contains the following records.

Emp_ID	Name	Department	Attendance (%)
1001	Alisha	Finance	92
1002	Ravi	IT	NaN
1003	Neha	Finance	85
1004	Salim	IT	NaN
1005	Priya	IT	88

Use this data set to perform the following tasks relating to missing values from the dataset as per the following tasks using Python code:

- List those rows that contain missing value in any column.
- Clean the dataset so that there is no row with any missing value.
- Find number of rows which were removed during cleaning.
- Display the cleaned dataset.

Q10: A dataset `Student_Marks.csv` is given below. Use this data set to perform tasks relating to replacement of Missing Values using Mean/Median/Mode as asked..

Roll No	Marks in Maths	Marks in Science	Marks in English
1001	78	85	82
1002	NaN	72	70
1003	90	NaN	89
1004	45	55	NaN
1005	88	79	84

- Replace missing marks in Maths with the mean score in Maths examination.
- Replace missing marks in Science with the median score in Science
- Replace missing marks in English with the mode score in English examination.
- Display the modified dataset highlighting the values that are used for replacing the missing values.

Q11: A dataset `Customer_Info.csv` contains missing categorical data. Fill this Missing Categorical data as per the following tasks:

Cust_ID	Name	City	Membership
1001	Alisha	Delhi	Gold
1002	Ravi	Mumbai	NaN
1003	Neha	Delhi	Silver
1004	Salim	Jaipur	NaN
1005	Priya	Mumbai	Gold

- Find the missing values in the column named Membership.
- Replace the missing values with the placeholder "Not Shared" and display the dataset
- Count how many missing values were replaced in the dataset.

Q12: A dataset Sales_Records.csv is given below. Filter the data with non-null values in specific columns, as asked in the following tasks:

Order_ID	Product	Total Sales	Profit/loss
1001	Laptop	50000	8000
1002	Mobile	NaN	3000
1003	Mobile	7000	NaN
1004	Mobile	12000	2000
1005	Laptop	52000	9000

- Filter only those rows where both Total Sales and Profit/loss columns have non-null values.
- Display the filtered dataset.
- Find out how many rows satisfy this condition.

Q13: A dataset Orders.csv contains duplicate rows. Perform the following tasks for this data set.

Order_ID	Customer	Amount
1001	A	2000
1002	B	3000
1001	A	2000
1003	A	2500
1002	B	3000

- Find the duplicate rows in the data set.
- Remove the duplicate rows. Just keep the first occurrence of a duplicate row in the data.
- Display the cleaned dataset and identify the number of duplicate rows removed.

Q14: A dataset Sales_Records.csv is given below. Rename the columns of this data set as the following task:

Order_ID	Product	Total Sales	Profit/loss
1001	Laptop	50000	8000
1002	Mobile	NaN	3000
1003	Mobile	7000	NaN
1004	Mobile	12000	2000
1005	Laptop	52000	9000

- Rename the column **Order_ID** to **Order**; **Total Sales** to **Sales**; and **Profit/loss** to **Profit**.
- Display the updated dataset with modified column names.

Q15: Create a dataset named employee_attendance.csv that contains the following records. Replace the abbreviations in the Department using the following and display the modified data set.

HR to Human Resources
IT to Information Technology

Emp_ID	Name	Department	Attendance (%)
1001	Alisha	HR	92
1002	Ravi	IT	NaN
1003	Neha	HR	85
1004	Salim	IT	NaN
1005	Priya	IT	88

Q16: A dataset **Transactions.csv** is given in the following table. Convert the data of Date column into a proper datetime format. Also create two more columns – first for storing the month and second for the year. Both these column values should be extracted from Date column.

TransactionNo	Date	Amount
1	2026-01-10	2000
2	2026-02-15	3500
3	2026-03-20	1500

Q17: A dataset **User_Input.csv** contains the following comments data. Normalise the Comment column by converting all text to lowercase and by removing leading and trailing spaces. Display the cleaned data set.

User_ID	Comment
1001	Excellent Service
1002	good product
1003	VERY BAD EXPERIENCE
1004	Average

Session 3 (Covers Unit 3 of MCS-67):

For the following questions, create the file **employee_Data.csv** using the data given in the following table:

Emp_ID	Name	Age	Salary	Depart	Email	Join_Date	Feedback
1001	Alisha	25	30000	Finance	alisha@xyz.in	2024-01-10	Good work!!!
1002	Ravi	32	35000	IT	ravi@xyz.in	2024-05-15	Excellent service!!!
1003	Neha	45	120000	Finance	neha@xyz.in	2025-03-20	Great...Nice work
1004	Salim	28	32000	IT	salim@xyz.in	2026-07-12	Very Good...
1005	Priya	38	37000	IT	priya@xyz.in	2024-11-01	Very good...

Q18: Categorise the age data of employees based on the following age categories:

20–35 as Youth

36–45 as Middle Age

46–60 → Seniors

Create a new column, **GroupOfAge**, and display the updated data.

Q19: Find the outliers for the data in the column salary by finding Quartile 1 and Quartile 3 and Inter Quartile Range. What is the name of the employee whose salary is an outlier?

Q20: It was found that the salary of employee 1003's salary is only 33000. Now check for an outlier in salary again.

Q21: Create a random sample containing 20% of the dataset records. Set a starting seed value. List the sampled data.

Q22: Create two new columns, namely **Username** and **DomainName**, from the Email column. Assume @ as a separator. Display the transformed data.

Q23: Remove all the special characters from the feedback column. You may use a regular expression to perform this task.

Q24: Add another column in the employee dataframe containing Surname (You may assume any surnames). Concatenate the Name column

with the Surname column to create a column named FullName. Name and Surname should be separated by a Blank/space.

Q25: Split the FullName column of the dataset into two newly created columns, FirstName and LastName.

Session 4 (Covers Unit 4 of MCS-67):

For the following questions, create the file **employee_SalesData.csv** using the data given in the following table:

Emp_ID	Name	Department	Region	Product	Sales	Year
1001	Alisha	Finance	North	Laptop	50000	2025
1002	Ravi	IT	South	Phone	20000	2026
1003	Neha	Finance	East	Tablet	15000	2024
1004	Salim	IT	West	Laptop	52000	2025
1005	Priya	IT	North	Phone	21000	2025
1006	Alvin	Finance	South	Tablet	18000	2024

Q26: First, split the datasets into two datasets named data1set, consisting of Emp_ID, Name, Department, year; and data2set, consisting of Emp_ID, Region, Product, Sales. Now, remove a record of Emp_ID 1004 from data1set and remove a record of Emp_ID 1002 from data2set. Perform an inner join using Python code. Verify the joined data and the missing records.

Q27: Perform the left outer join, right outer join, and outer join on the data1set and data2set. Compare the results of these join operations with the result obtained in Q26.

Q28: Create a dataframe using the file employee_SalesData.csv. Use this dataset to create overlapping row-wise fragments. The first consists of the first four rows, and the second consists of the last four rows. Concatenate the two dataframes so that no duplicate rows appear in the final output.

Q29: Create a dataframe using the file employee_SalesData.csv. Split this dataset to create two Datasets, the first having Emp_ID, Name, Department and the second having Emp_ID, Department, Region, and Sales. Concatenate the two datasets column wise on Emp_ID. How have you handled the overlapping Department.

Q30: Create a dataframe using the file employee_SalesData.csv. Now, create a two level hierarchical index using Department at level 1 and Product at level 2. Display the data set after indexing.

- Q31: Manipulate the index created in Q30 by making Product at level 1 and Department at level 2. Display the data set after the manipulation.
- Q32: Create a dataframe using the file `employee_SalesData.csv`. Index the dataset on `Emp_ID` and create a pivot table with Product as the Column and Sales as the values. Now, convert the data back to the original format.
- Q33: Create a dataframe using the file `employee_SalesData.csv`. Now, create a multilevel index as created in Q30. Now, convert the columns into rows using the `Stack()` function, display the resulting data, and revert it to the earlier format.

Session 5 (Covers Unit 5 of MCS-67):

For the following questions, create the file `employee_DataPerformance.csv` using the data given in the following table:

Emp_ID	Name	Department	Region	Salary	Sales	Rating
1001	Alisha	Finance	North	40000	5000	3.5
1002	Ravi	IT	South	60000	7000	4.2
1003	Neha	Finance	East	55000	6500	4.0
1004	Salim	IT	West	65000	8000	4.5
1005	Priya	IT	North	42000	5200	3.8
1006	Alvin	Finance	South	58000	7200	4.1

- Q34: Create a dataframe using the file `employee_SalesPerformance.csv`. Now, group the data by Department and find the mean Salary and Rating for each Department.
- Q35: Create a dataframe using the file `employee_SalesPerformance.csv`. Group the data by Region and find the mean Rating, total sales, and number of employees for each Region.
- Q36: Create a dataframe using the file `employee_SalesPerformance.csv`. Group the data by Department and iterate over each group to display the name and data.
- Q37: Create a dataframe using the file `employee_SalesPerformance.csv`. Group the data by Department, and select the Salary and Sales columns to compute the total for each Department.
- Q38: Create a dataframe using the file `employee_SalesPerformance.csv`. Group data by custom mapping on Region, as North and East are mapped to NORTHEAST, and South and West are mapped to SOUTHWEST. Now find the average Sales of each of these two groups. Also find the number of employees in each group.

- Q39: Create a dataframe using the file `employee_SalesPerformance.csv`. Group the data by Department, and find the mean, total, and maximum of Salary and Sales within each group.
- Q40: Create a dataframe using the file `employee_SalesPerformance.csv`. Group the data by Department, and Z-statistics for ratings using the standard formula. Create a new column named ZRating.
- Q41: Create a dataframe using the file `employee_SalesPerformance.csv`. Group the data by Region, and display only those groups that have more than one employee.

Session 6 (Covers Unit 6 of MCS-67)

For the following questions, create the file `employee_DataPerformance.csv` using the data given in the following table, and use matplotlib to perform the following questions:

Order_ID	Product	Category	Price	Quantity	Discount (%)	Sales	Rating
O01	Laptop	Electronics	55000	1	10	49500	4.5
O02	Headphones	Electronics	2000	2	5	3800	4.2
O03	T-shirt	Clothing	800	3	15	2040	4.0
O04	Shoes	Footwear	3000	1	20	2400	4.3
O05	Watch	Accessories	2500	2	10	4500	4.6
O06	Mobile	Electronics	20000	1	8	18400	4.4
O07	Jeans	Clothing	1500	2	12	2640	4.1
O08	Sandals	Footwear	1200	2	10	2160	3.9
O09	Backpack	Accessories	1800	1	5	1710	4.2
O10	Tablet	Electronics	15000	1	7	13950	4.3

- Q42: Using the dataset, create a line graph between Product and Sales. Is there any trend?
- Q43: Using the dataset, create two sub-plots in a single plot: First, between Price and Sales and second, between Discount and Sales. Are there any similarities between the subplots?
- Q44: Using the dataset, create a plot between Price and Sales. In this plot, use a dotted green line with a triangle marker style.
- Q45: Using the dataset, create a plot between Order_ID and Sales. The plot should have axis labels, a title and a legend.
- Q46: Using the dataset, create a plot between Order_ID and Sales, as created for Q45. In this graph, annotate the highest and lowest sales.

- Q47: Make the plot as mentioned in Q46. Save this plot as a .png file first and then as a .jpg file. Make sure that the high-resolution plot is copied.
- Q48: Using the dataset, create a plot between Category and total sales in that category. Give a suitable axis label, title and legend to this plot. Annotate the bar with the highest total sales.
- Q49: Using the dataset, create a histogram for the Sales and price. Analyse the two distributions.
- Q50: Using the dataset, create a scatter plot for the Sales and rating. Add a regression line to the scatter plot. Describe the trend.

Session 7 and 8 (Cover Units 7, 8 and 9 of MCS-67)

For the following questions, create the file **Customer_SalesData.csv** showing data for the first 20 days of January 2026. Using the data given in the following table, use matplotlib and other Python libraries to perform the following questions:

OID	Date	City	Region	Age	Category	Sales (₹)	Profit (₹)	Latitude	Longitude
O01	2026-01-01	Delhi	North	25	Grocery	1500	200	28.61	77.23
O02	2026-01-03	Mumbai	West	32	Electronics	12000	1500	19.07	72.87
O03	2026-01-05	Chennai	South	28	Clothing	3500	500	13.08	80.27
O04	2026-01-07	Kolkata	East	40	Grocery	2000	250	22.57	88.36
O05	2026-01-09	Bangalore	South	35	Electronics	15000	2000	12.97	77.59
O06	2026-01-11	Jaipur	North	29	Clothing	4000	600	26.91	75.79
O07	2026-01-13	Pune	West	31	Grocery	1800	300	18.52	73.85
O08	2026-01-15	Hyderabad	South	27	Electronics	11000	1300	17.38	78.48
O09	2026-01-17	Ahmedabad	West	45	Clothing	5000	700	23.02	72.57
O10	2026-01-19	Lucknow	North	38	Grocery	2200	350	26.85	80.95

- Q51: Create a bar chart with headings to show product sales by region.
- Q52: Draw the appropriate graph type to show Sales vis-à-vis Profit.
- Q53: Create a scatter plot matrix (pair plot) for the three numeric variables - Sales, Profit, and Age. Are there any possible relationships?
- Q54: Convert the data of the 'Date' column to a datetime data type. Create a time-series line plot of sales over time.
- Q55: Use the Latitude and Longitude columns to create a visualisation of sales distribution across geographical locations.
- Q56: Represent Sales and profit using box plots.
- Q57: Create a side-by-side box plot of profit across different categories. Compare profits across categories using this box plot.

Q58: Make a scatter plot of Sales and Profit. Highlighting different categories in different colours and marker sizes. Also, add a grid.

Q59: Arrange the following plots in a 2×2 grid layout by drawing appropriate graphs for each:

1. Region-wise sales
2. Category-wise profit
3. Distribution of the Age column
4. Distribution of Sales

Q60: Add annotations to highlight minimum and maximum sales and minimum and maximum profit in the related graphs.

Q61: Create a customised plot setting, including font size, plot size, etc.

Q62: Create a line plot showing Sales with respect to the Date of sale. Add a trendline to the plot.

Q63: Create a 3D plot using Sales, Profit and Age. Visualise the plots by changing the variable on the axes.

Q64: Create a bipartite graph; you may select OID as the first set and a Category as the second set.

Q65: Build a hierarchical tree showing data as Root → Region → City → Product Category

Q66: Construct and visualise a minimum spanning tree using city distances based on latitude/longitude.

Session 9 and 10 (Cover Units 10, 11 and 12 of MCS-67)

For the following questions, create the file **Customer_PurchaseData.csv**.

Using the data given in the following table, use matplotlib and other Python libraries to perform the following questions:

ID	Age	Income	Spending Score	Region	Category	Visits	Online Time	Purchase	Rating
1	22	25000	65	North	Grocery	5	12	Yes	4.2
2	35	55000	40	West	Electronics	2	8	No	3.8
3	28	32000	70	South	Clothing	6	15	Yes	4.5
4	45	75000	30	East	Grocery	3	6	No	3.5
5	30	60000	80	South	Electronics	7	18	Yes	4.8
6	26	40000	55	North	Clothing	4	10	Yes	4.1
7	38	50000	45	West	Grocery	3	9	No	3.9
8	29	45000	75	South	Electronics	6	16	Yes	4.6
9	50	80000	35	West	Clothing	2	5	No	3.6

ID	Age	Income	Spending Score	Region	Category	Visits	Online Time	Purchase	Rating
10	33	52000	60	North	Grocery	5	11	Yes	4.3

- Q67: Train a decision tree classifier to predict Purchase. Visualise this decision tree.
- Q68: Train a Random Forest model for Purchase. Plot its important features.
- Q69: For the decision tree of Question 67, plot accuracy against the tree depth.
- Q70: Plot a histogram and univariate Kernel Density Estimate (KDE) for Income.
- Q71: Plot bivariate KDE for Income against Spending Score.
- Q72: Plot the density distribution of the Spending Score for Purchase (Yes/No). Make this distribution across the regions
- Q73: Create a pairplot for the numeric variables. Use this plot to find the possible clusters or patterns.
- Q74: Plot the regression line between Income and Spending Score.
- Q75: Make the residual plot and QQ plot for Question 74.
- Q76: Based on the data in this section, make plots that, according to you, would help managers improve purchase rating and purchases.

1.7 SUMMARY

This section covers the following points:

- It gives you a basic introduction to the Python and tools of the Python programming.
- Introduces of process of writing and running Python program.
- Provides a comprehensive list of problems in the area of data wrangling and visualisations that you can solve using pandas, matplotlib

Structure**Page No.**

- 2.0 Introduction
 - 2.1 Objectives
 - 2.2 General Guidelines
 - 2.3 Installing R & R-Studio
 - 2.4 R-Studio for Data Science
 - 2.5 Session wise list of Lab Assignments
 - 2.6 Summary
-

2.0 INTRODUCTION

This is the era of Information and communication technologies. With the enhancement of information and communication technology tools, world wide web, mobile technologies, IoT devices etc. led to creation of very large amount of heterogeneous data. Today, you are surrounded by a vast reservoir of data, which is required to be processed and analysed. R programming is one of the popular tools for performing statistical analysis of data. Please note that you may be using R to analyse the structured data most often, however, R programming can also be used to process unstructured data too. As far as size of data that can be processed at a time using R is concerned, R is claimed to process about 1 million records of data with ease. Although several applications claim to use even bigger data sets using R. On the whole, R is truly a powerful programming language for data analysis. R also has a very rich syntax for graphical presentation of the data and results.

R programming language was designed at the University of Auckland by R Ihaka and R Gentleman and is available as an open-source programming language. In this lab session, you may use open-source license RStudio Desktop version. This section introduces you to using R Studio desktop for R programming. You may refer to <https://www.r-project.org/> about R project and <https://cran.r-project.org/manuals.html> R documentation. You may also refer to <https://www.rstudio.com/products/rstudio/> website for RStudio.

This lab manual is designed to guide you through the practical aspects of learning and applying R programming for data science. It is structured to help you gradually build skills from the fundamentals to advanced techniques, making the learning experience self-paced and practice-oriented. R is a powerful language for statistical computing and data analysis, and this manual provides a hands-on approach to understand its environment, core programming concepts, and data structures.

As a pre-requisite to understand and implement the sessions given in this lab manual you need to have an understanding of concepts delivered in Units 13,14,15 and 16 of MCS-068 i.e. Predictive data analysis course.

From Unit 13 – Basics of R Programming, you will first explore the foundations such as the R environment, data types, variables, operators, and factors. You will practice writing simple programs using decision-making statements, loops, and functions. You will also work with fundamental data

structures like vectors, strings, lists, matrices, arrays, and data frames. These basics will enable you to store, manipulate, and manage data effectively, which forms the core of programming in R.

From Unit 14 – Data Interfacing and Visualisation in R, the focus shifts to working with real-world data. You will learn how to read and write data from different file formats including CSV, Excel, XML, JSON, and binary files, as well as interface R with databases and extract information from the web. You will also practice cleaning and preprocessing data to make it suitable for analysis. A key part of this unit is data visualisation, where you will learn to create bar charts, box plots, histograms, line graphs, and scatterplots, using them to communicate insights effectively.

From Unit 15 – Data Analysis and R, you will develop the ability to apply statistical methods and models. This includes performing hypothesis testing using the Chi-Square test, building linear and multiple regression models, and exploring logistic regression for classification tasks. You will also be introduced to time series analysis, learning how to identify trends, seasonality, and randomness in data observed over time and how to forecast future patterns using R's built-in functions.

From Unit 16 – Advanced Analysis using R, you will step into more sophisticated techniques that bring machine learning concepts into practice. You will construct decision trees and random forests for predictive modelling, study various classification algorithms, and apply clustering techniques such as K-Means to group similar data points. Finally, you will explore association rule mining to discover hidden relationships in large datasets, as often seen in recommendation systems.

By working through this manual step by step, you will not only gain confidence in using R as a tool for statistical computing and data analysis but also develop problem-solving skills by applying concepts to practical exercises. The emphasis is on self-learning, where you are encouraged to experiment with commands, observe outputs, and reflect on results to strengthen your understanding.

2.1 OBJECTIVES

After completing this lab course, you will be able to:

- Develop the ability to write and execute R programs using basic constructs, data types, operators, functions, and data structures.
- Acquire skills to import, clean, preprocess, and visualise data from multiple sources including files, databases, and the web using R.
- Apply statistical techniques such as Chi-Square tests, regression models, and time series analysis for data-driven insights.
- Implement advanced analytical methods in R including decision trees, random forests, classification, clustering, and association rule mining.
- Gain hands-on experience in using R as a complete tool for statistical computing, data analysis, and predictive modelling.

2.2 GENERAL GUIDELINES

- Students should attempt all problems/assignments given in the list session wise.
- Students may seek assistance in doing the lab exercises from the concerned lab instructor. Since the assignments have credits, the lab instructor is obviously not expected to tell you how to solve these, but you may ask questions concerning the implementation of the concepts of Data Science using the various tools or any other technical problem.
- For each programming related task assigned in the respective session you should add comments above each function in the code, including the main function. This should also include a description of the function written, the purpose of the function, meaning of the argument used in the function and the meaning of the return value (if any).
- These descriptions should be placed in the comment block immediately above the relevant function source code.
- The comment block above the main function should describe the purpose of the program. Proper comments are to be provided where and when necessary, in the programming.
- The program should be interactive, general and properly documented with real Input/ Output data.
- If two or more submissions from different students appear to be of the same origin (i.e. are variants of essentially the same program), none of them will be counted. You are strongly advised not to copy somebody else's work.
- It is the responsibility of the students to create a separate directory to store all the programs, so that nobody else can read or copy.
- Observation book and Lab record are compulsory
- The list of the programs (list of programs given at the end, session-wise) is available to you in this lab manual. For each session, you must come prepared with the solution to the respective tasks, written in the Observation Book. You should utilise the lab hours for executing the tasks/programs, testing for various desired outputs and enhancements of the programs.
- As soon as you have finished a lab exercise, contact one of the lab instructors / in-charge in order to get the exercise evaluated and also get the signature from him/her on the Observation book.
- Completed lab assignments should be submitted in the form of a Lab Record in which you have to write the algorithm, program code along with comments and output for various inputs given.
- The total number of lab sessions (3 hours each) for this sections are 10 and the list of assignments is provided session-wise. It is important to observe the deadline given for each assignment.

In the subsequent section, we will learn about the installation of R and R-Studio, and there after we will learn how to work with R-Studio. So, lets begin with the process of installing R and R-studio.

2.3 INSTALLING R AND R STUDIO

First you must install R programming with its environment, which includes software features for data creation, storage, manipulation with a rich set of array-based operators, creation of graphs for data visualisation and a rich set of data analysis functions. In addition, it has the constructs of programming language including conditional statements, looping constructions, functions, etc. Like the present-day programming language, R also provides a rich set of ever increasing number of packages, which support many additional features. However, in order to use the features of R, you must first install it. You can use R website <https://cran.r-project.org/> to download R programming, as per your operating system and install R into your system.

Once you have installed R, the next task is to install the open source RStudio Desktop. For this you may visit the website :

<https://www.rstudio.com/products/rstudio/download/#download> and download the RStudio Desktop, which has open-source license. You should install this software on your computer.

2.4 R-STUDIO FOR DATA SCIENCE

Once the latest version of R and RStudio Desktop have been installed, you are ready to use RStudio Desktop version for various purposes. In this section, we discuss how you can use R-Studio for data science.

On opening the RStudio, you will get the window as shown in Figure 1. The important tab, which you should all be aware of is the help tab, as shown in Figure 1. You may notice that Global environment in Figure 1 is currently empty.

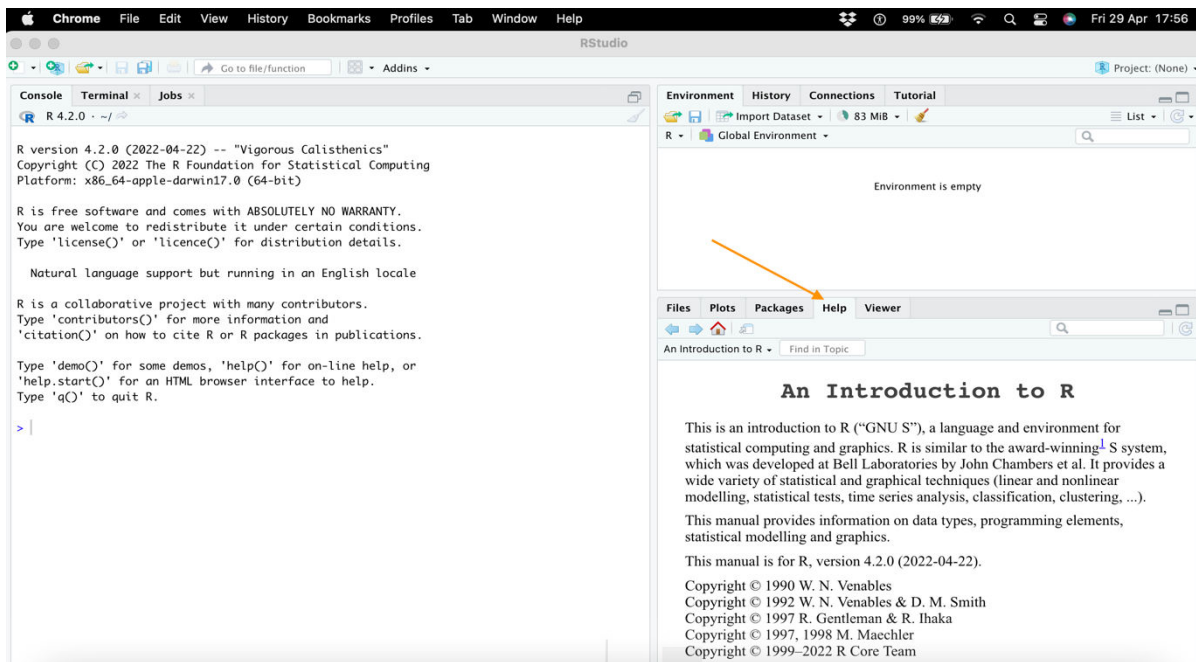


Figure 1: The RStudio Environment with focus on Help

Now, you are ready for R programming, but before that please go through the Block 4 of MCS226: Data Science and Big data, which covers some of the basic set of useful commands of R programming.

A data science project involves several steps including data creation, modification, graphical data representation, exploratory data analysis and modeling. Let us now perform some basic operations using R in R Studio Desktop, with the help of a hypothetical example. Please note this example does not cover the detailed requirements of a data science project, which also has major steps relating to feature extraction and model optimization.

Example 1: Consider a school collect following data of its students:

Enrolment Number, % attendance, Average Study Hours per week, %of marks in final

The school believes that student's performance is highly dependent of these two factors. The hypothetical data for this was collected in an MS-Excel file. Show how R can be used to perform this analysis.

Solution: For performing this analysis using a sample file PracticalData.xlsx, since it is a hypothetical example, therefore, we are using only 10 data sets for these variables. This data set is shown in the Figure 2.

Enrolment Number	Marks Percent	Percent Attendance	Weekly Study Hr
S20200001	95	91	20
S20200002	91	93	18
S20200003	62	70	6
S20200004	75	80	10
S20200005	88	92	15
S20200006	79	88	16

S20200007	55	70	3
S20200008	69	75	8
S20200009	60	70	6
S20200010	80	75	17

Figure 2: Sample Data in PracticalData.xlsx File

In order to import this data in to RStudio Desktop, you may select “Import Dataset” option followed by “From Excel...” in the drop down list, as shown in Figure 3.

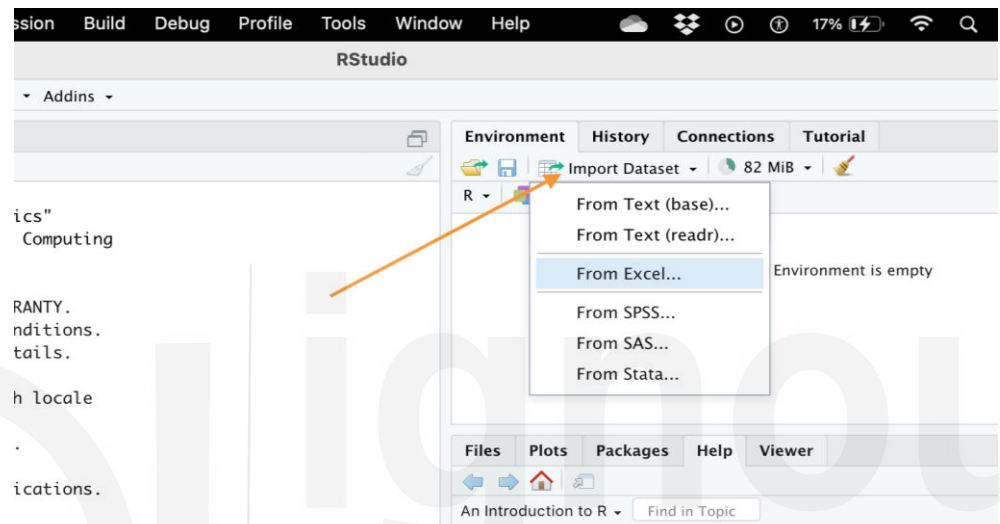


Figure 3: Selection for importing data from Excel

However, as you may be importing the data for the first time, the related package may not have been installed. Therefore, you may receive the Dialog Box, as shown in Figure 4.

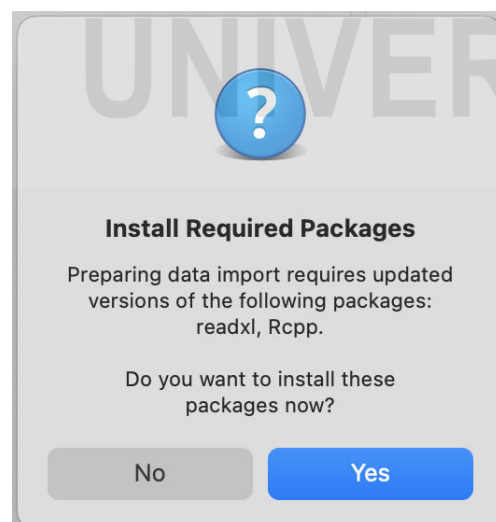


Figure 4: Dialog Box for installing packages for reading Excel file

Select “Yes”, the highlighted option in Figure 4. It will result in installation of the required packages. After the required package is installed, the dialog box, as shown in Figure 5, will appear for importing Excel data.

Import Excel Data

File/URL:
~/Library/CloudStorage/OneDrive-Personal/MCS226/PracticalData.xlsx Browse...

Data Preview:

EnrolmentNumber (character)	MarksPercent (double)	PercentAttendance (double)	WeeklyStudyHr (double)
S20200001	95	91	20
S20200002	91	93	18
S20200003	62	70	6
S20200004	75	80	10
S20200005	88	92	15
S20200006	79	88	16
S20200007	55	70	3
S20200008	69	75	8

Previewing first 50 entries.

Import Options:

Name: PracticalData Max Rows: ☒ First Row as Names
 Sheet: Default Skip: 0 ☒ Open Data Viewer
 Range: A1:D10 NA:

Code Preview:

```
library(readxl)
PracticalData <- read_excel("~/Library/CloudStorage/OneDrive-Personal/MCS226/PracticalData.xlsx")
View(PracticalData)
```

Import Cancel

Figure 5: Importing Excel Data in RStudio Desktop

You may notice that you can see the data in Data Preview. The data type of the data is also shown. For example, MarksPercent will have *double* data type. In the Import option the option “First Rows as Name” is ticked. One of the important options in import options is NA, which relates to missing data. You may go through various options from the help files. Once you press the Import button in Figure 4, the data will be imported in RStudio, as shown in Figure 6. Please notice the highlighted commands of R programming in the Console tab

The screenshot shows the RStudio interface. The Environment pane on the right displays 'PracticalData' with 10 observations and 4 variables. The Console pane at the bottom shows the following R commands:

```
> library(readxl)
> PracticalData <- read_excel("~/Library/CloudStorage/OneDrive-Personal/MCS226/PracticalData.xlsx")
> View(PracticalData)
```

The Data Viewer on the left shows the first 10 rows of the data, matching the preview in Figure 5.

Figure 6: The R commands for importing data from excel

Now, let us do one more activity before you go to performing an analysis. Let us draw the plots.

To draw the plot it may be good idea to separate the data that you may need for plotting. This can be performed using the following R code:

```
xydata <- PracticalData[, c('MarksPercent', 'PercentAttendance', 'WeeklyStudyHr')]
```

Next, you draw plot between attendance versus marks and study hours versus marks using the following code:

```
plot(x = xydata$PercentAttendance, y = xydata$MarksPercent,
      xlab = "Percentage of Attendance", ylab = "Final Percentage of Marks",
      xlim = c(70,100), ylim = c(50,100), main = "Attendance versus Marks Percentage"
    )
plot(x = xydata$WeeklyStudyHr, y = xydata$MarksPercent,
      xlab = "Weekly Hours of Study", ylab = "Final Percentage of Marks",
      xlim = c(1,20), ylim = c(50,100), main = "Study Hours versus Marks Percentage"
    )
```

Figure 7 shows these statements in R console.

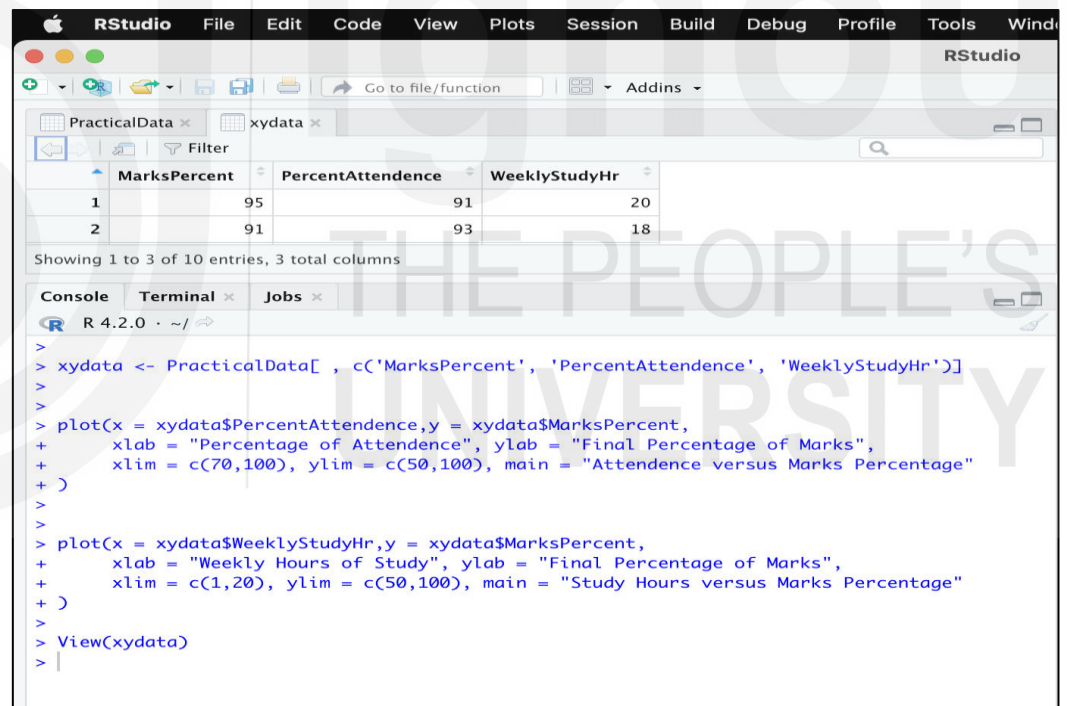
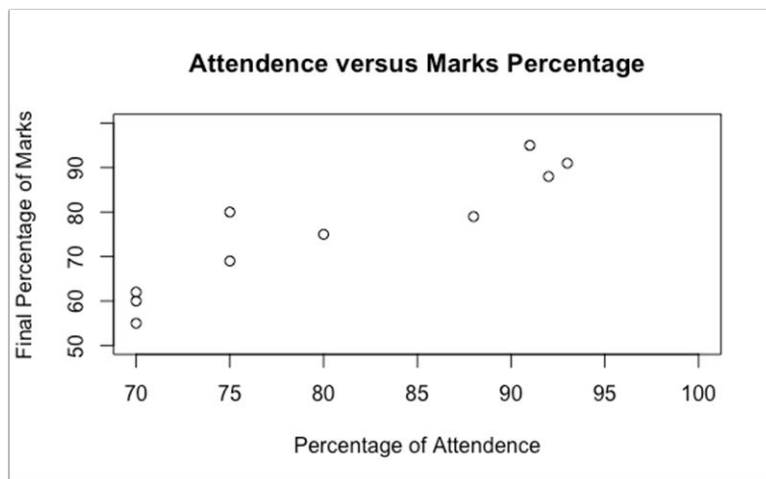
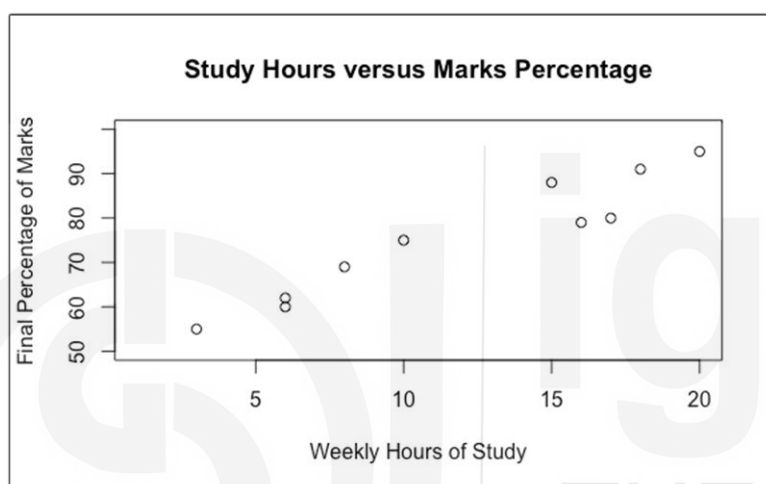


Figure 7: Extracting data and plotting Scatter plots

The output of these commands is shown in Figure 8.



(a) Attendance versus Marks



(b) Hours versus Marks

Figure 8: Scatter Plots of data

Figure 8 suggests that there is possibility of relationship between marks of the students with the class attendance and hours of study. You can also plot pairs of these three variables together using the R code:

```
pairs(~ PercentAttendance+MarksPercent+WeeklyStudyHr, data = xydata,
      main = "Scatter Plot Matrix")
```

This will produce the following output:

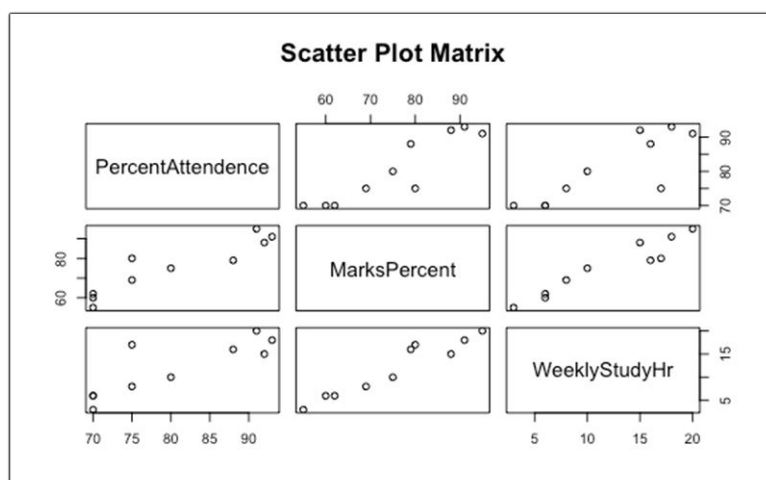


Figure 9: The Scatter plots for all the pairs

You may observe that the scatter plots of Figure 8 and Figure 9 show possibilities of linear regression among the data. Therefore, let us use linear regression for making the model for this problem. In this case, we will use multiple linear regression, to demonstrate how multiple regression can be performed using R programming.

You need to define the response variable or dependent variable, which for the present model would be the marks percentage of the students. The two independent or exploratory variables being used for this model are attendance and study hours. Thus, the basic multiple linear regression proposed model is:

$$\text{MarksPercent} = a * \text{PercentAttendance} + b * \text{WeeklyStudyHr} + c$$

Please note that c is the intercept in the model given above. You need to determine the value of a , b and c using the multiple linear regression.

First, you create the data, as done earlier.

```
xydata <- PracticalData[, c('MarksPercent', 'PercentAttendance', 'WeeklyStudyHr')]
summary(xydata)
```

The result of summary statistics of `xydata` is shown in Figure 10.

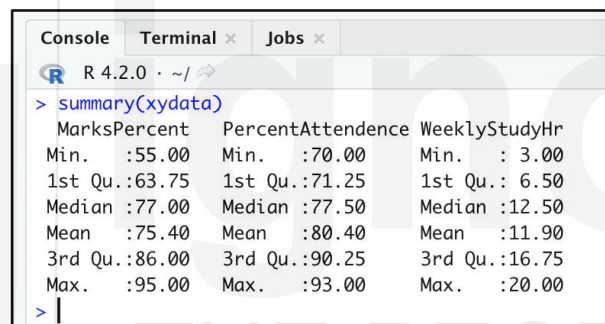


Figure 10: Statistical summary of x - y data

You may notice that Figure 10 shows the summary of each of the column of `xydata` variable. This summary shows minimum value, the first quartile value, the median, the mean, 3rd quartile and then the maximum value of the data. You may please note that for all the three attributes the mean and median values are close, which represents that data may be of normal distribution.

Next, you may use `lm()` function of R programming (Please refer to Block 4 Course MCS226).

```
ParameterLMR = lm (MarksPercent ~ PercentAttendance+WeeklyStudyHr, data=xydata)
```

The statement given above generates the multiple regression model –

$$\text{MarksPercent} = a * \text{PercentAttendance} + b * \text{WeeklyStudyHr} + c$$

The model coefficients and other details of the model are stored in the variable `ParameterLMR`. To display the summary of this model, you use the `summary` function to display this variable, as shown in Figure 11.

```
> ParameterLMR = lm (MarksPercent ~ PercentAttendance+WeeklyStudyHr, data=xydata)
> summary(ParameterLMR)

Call:
lm(formula = MarksPercent ~ PercentAttendance + WeeklyStudyHr,
    data = xydata)

Residuals:
    Min       1Q   Median       3Q      Max
-6.4962 -0.9043  0.4864  1.9889  2.5773

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    14.3005    12.7533   1.121  0.29914
PercentAttendance  0.5450     0.1955   2.788  0.02699 *
WeeklyStudyHr    1.4522     0.3165   4.589  0.00252 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.081 on 7 degrees of freedom
Multiple R-squared:  0.9608,    Adjusted R-squared:  0.9496
F-statistic: 85.73 on 2 and 7 DF,  p-value: 1.195e-05
```

Figure 11: Multiple Regression Result

The output first shows the value of residual, which measures the distance or difference between the actual value, also called observed value, and the predicted value by the regression model. The residuals are represented as 5-point summary. You can observe the minimum difference being about -6.5% and maximum being 2.58%. Next, please observe the critical t-values and p-values (Pr) for all the coefficients. It may be noted that the degree of freedom of this regression is 7 (also shown in the Figure 11). You may compute the critical t-values for this degree of freedom. Please note that the regression model is:

$$\text{MarksPercent} = 0.5450 * \text{PercentAttendance} + 1.4522 * \text{WeeklyStudyHr} + 14.3005$$

Next, please note that the probability (Pr) or p-value of Intercept is about 0.3, which is > 0.05 , implying that the data is not providing statistically significant evidence against the null hypothesis about Intercept (the null hypothesis about intercept is that intercept should be 0). However, the other two coefficients have the p-value as 0.027 and .0025, which are statistically significant and provides evidence that there is a significant relationship between the response variables and explanatory variables. The adjusted r-squared variable, which is normally used instead of r-squared value in case of multiple regression, shows that about 94.96% of variation of response variable can be explained by the explanatory or independent variables. However, it is not a measure of goodness of regression model. The p-value associated with the F-statistics for this regression model rejects the null hypothesis that all the coefficients in this regression model should be zero. However, due to a very high level of standard error in the intercept value leads us to situation, which requires you to explore better models.

You may try a linear regression between MarksPercent (response or dependent variable) and WeeklyStudyHr (explanatory or independent variable). The regression model for this would be:

$$\text{MarksPercent} = a * \text{WeeklyStudyHr} + b$$

You may use the following R code for performing the regression:

```
SingR = lm (MarksPercent ~ WeeklyStudyHr, data=xydata)
```

summary(SingR)

The result of these two operations would be:

```

R 4.2.0 ~|
> SingR = lm (MarksPercent ~ WeeklyStudyHr, data = xydata)
> summary(SingR)

Call:
lm(formula = MarksPercent ~ WeeklyStudyHr, data = xydata)

Residuals:
    Min       1Q   Median       3Q      Max
-6.5887 -2.0608  0.6867  2.2021  5.7990

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    49.293      3.073   16.041 2.29e-07 ***
WeeklyStudyHr    2.194      0.233    9.415 1.33e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.187 on 8 degrees of freedom
Multiple R-squared:  0.9172,    Adjusted R-squared:  0.9069
F-statistic: 88.64 on 1 and 8 DF,  p-value: 1.329e-05

>
> |

```

Figure 12: Result of Regression

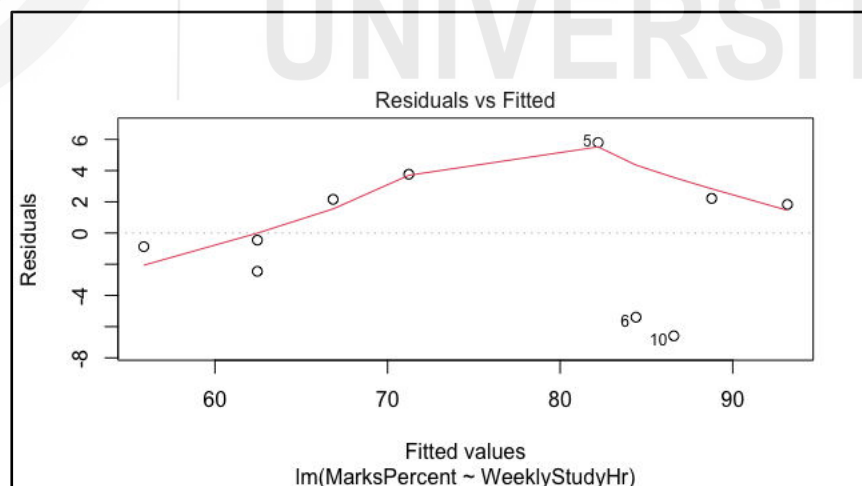
You may observe the Pr values for both Intercept and explanatory variable is very low. In addition, the p-value for F-statistics is also very low. Thus, this regression model can be used to predict student marks. The model is represented by following linear regression equation:

$$\text{MarksPercent} = 2.194 * \text{WeeklyStudyHr} + 49.293$$

In addition, you may also draw various plots related to regression using R. The following function call will show you various plots related to regression:

```
plot(SingR)
```

Some of these plots are shown in Figure 13.



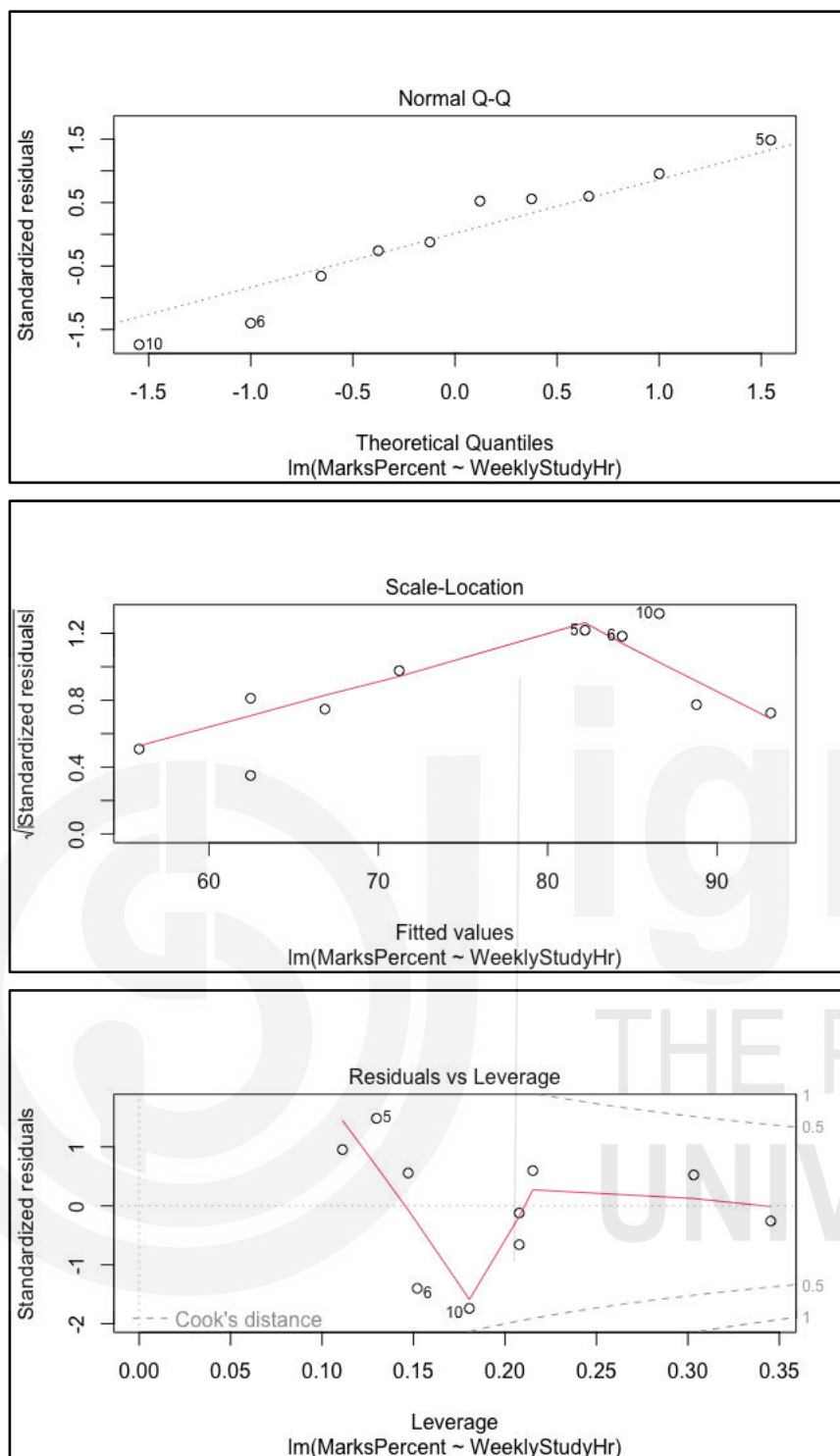


Figure 13: The plots of Linear regression

The plots in Figure 13 mainly represent characteristics of residuals in the fitted model. You may refer to the further readings for more details on these plots. However, you may notice that data items 6 and 10 are the ones, where you have maximum standard error. You may observe that both these students are putting large number of study hours, yet not able to score marks as high as the students who are putting almost similar number of hours of study.

In general, a data science project would require you to create, clean, prepare data, followed by exploratory data analysis, which will give you insights of data. This will lead you to final analysis of data and reporting your results.

In the next section, we present a list of problems, which you should attempt during your laboratory sessions.

2.5 SESSION WISE LIST OF LAB ASSIGNMENTS

Let's try solving a different kind of a problems related to Predictive Analysis.

Important Note:

1. To perform the tasks assigned in the respective sessions, you are required to make necessary assumptions (may be related to data) wherever required.
2. Please Go Through MCS-068 Block-4 (Units 13,14,15 and 16).
3. **Session 1 & 2 are based on Unit 13 – Basics of R Programming** – You are required to refer to Block-4 of MCS-068 for properly answering these sessions
4. **Session 3 & 4 are based on Unit 14 – Data Interfacing and Visualisation in R** – You are required to refer to Block-4 of MCS-068 for properly answering these sessions
5. **Session 5,6 & 7 are based on Unit 15 – Data Analysis and R** – You are required to refer to Block-4 of MCS-068 for properly answering these sessions
6. **Session 8,9 & 10 are based on Unit 16 – Advanced Analysis using R** – You are required to refer to Block-4 of MCS-068 for properly answering these sessions

Session 1 & 2 are based on Unit 13 – Basics of R Programming
(refer to Block-4 Unit-13 of MCS-068)

Lab. Session 1: Basics of R Programming

Problem – 1: Data Types and Variables with Real-Life Data Analysis

Problem Statement:

A car showroom company wants to analyse the details of different cars such as mileage, engine type, brand name, and transmission category using R programming. The company needs to import the dataset, identify different data types, perform data conversions, apply filtering conditions, and manipulate text data for better business analysis and reporting.

Objective:

The objective of this experiment is to understand and apply different data types in R using a real-life car dataset and perform various operations for data analysis.

Activities to be Performed:

1. Import the car dataset from a CSV file or use the built-in mtcars dataset in R.
2. Store car details such as mileage (mpg) as numeric data, car names as character data, and transmission type as factor data.
3. Identify the type and structure of variables using typeof(), class(), and str().
4. Convert variables into suitable formats using as.numeric(), as.character(), and as.factor().

5. Perform arithmetic calculations such as finding average mileage and total engine capacity.
6. Apply logical conditions to filter cars having mileage greater than 20 using operators like `mpg > 20`.
7. Manipulate character data by combining car names using `paste()`, extracting specific letters using `substr()`, and converting names to uppercase using `toupper()`.

Expected Outcome:

After completing this experiment, you will be able to work with real-life datasets, identify and convert different data types, apply arithmetic and logical operations, and perform character manipulation effectively using R programming for practical data analysis applications.

Problem – 2: Working with Factors for Student Data Organisation

Problem Statement:

A college administration wants to analyse student information based on categories such as gender, course stream, and grade classification using R programming. The data needs to be organised into categories, analysed based on factor levels, and visualised graphically for better understanding and decision-making.

Objective:

The objective of this experiment is to understand the use of factors in R for organising categorical data, analysing factor levels, and visualising category-wise summaries.

Activities to be Performed:

1. Create a vector containing categorical student data such as gender (Example: "Male", "Female").
2. Convert the categorical data into factors using the `factor()` function.
3. Use the `levels()` function to identify and display different categories present in the dataset.
4. Count the number of students in each category using the `table()` function.
5. Reorder factor levels according to importance or preference, such as arranging grades from "Excellent" to "Average".
6. Rename factor levels for better readability and clarity.
7. Generate bar plots using `barplot()` to visually represent the distribution of categories such as male vs female students or stream-wise student count.

Expected Outcome:

After completing this experiment, you will be able to organise categorical data using factors, analyse category levels and frequencies, reorder and rename levels effectively, and visualise categorical information using graphical representations in R programming.

Problem – 3: Creating, Manipulating, and Summarising Matrices and Arrays for Sales Analysis in a Retail Store

Problem Statement:

A retail supermarket wants to analyse its product sales data across different branches and time periods using R programming. The management needs to store sales information in matrices and arrays, perform calculations, update records, extract specific details, and visualise sales trends for better business decision-making.

Objective:

The objective of this experiment is to understand the creation and manipulation of matrices and arrays in R and perform operations for analysing real-life sales data.

Activities to be Performed:

1. Create a matrix using the `matrix()` function to store product sales data of different store branches.
2. Generate a 3×3 matrix representing sales of three products across three branches.
3. Access specific rows, columns, or individual sales values using indexing methods such as `matrix[row, column]`.
4. Modify matrix values to update incorrect sales entries or newly updated records.
5. Add new product or branch sales information using `rbind()` and `cbind()`.
6. Create a three-dimensional array using the `array()` function to represent sales data across:
 - Different store branches
 - Multiple days
 - Different product categories
7. Extract a specific slice of data, such as sales details of one branch for a particular day.
8. Perform basic operations such as:
 - Total sales branch-wise using `rowSums()`
 - Average sales product-wise using `colMeans()`
 - Matrix multiplication using `%*%` for sales analysis calculations
9. Generate statistical summaries of the sales dataset using the `summary()` function.
10. Visualise sales performance using heatmaps or graphical representations through the `image()` function.

Expected Outcome:

After completing this experiment, you will be able to create and manipulate matrices and arrays in R, perform mathematical and statistical operations, update and extract multidimensional data, and visualise real-life business data effectively for analytical purposes.

Problem – 4: Constructing and Analysing Employee Management Data Using Data Frames

Problem Statement:

A multinational company wants to manage and analyse employee records using R programming. The HR department needs to store employee details such as employee ID, employee name, department, salary, and joining date in

a structured format. The organisation also wants to analyse salaries, identify department-wise employee distribution, update employee information, and visualise salary patterns for better workforce management and decision-making.

Objective:

The objective of this experiment is to understand the creation and manipulation of data frames in R and perform data analysis operations on structured employee management data.

Activities to be Performed:

1. Create a data frame containing employee information with the following columns:
 - EmployeeID (numeric)
 - Name (character)
 - Department (factor)
 - Salary (numeric)
 - JoiningDate (date stored as character)
2. Use the summary() function to generate statistical summaries of employee records.
3. Check for missing or incomplete employee data using functions such as is.na().
4. Extract subsets of employee data based on specific conditions, such as:
 - Employees having salary greater than ₹50,000
 - Employees belonging to a particular department like IT or HR
5. Select specific rows or columns from the employee data frame for analysis.
6. Add a new column such as TaxPaid calculated from employee salary details.
7. Modify employee records by updating information such as changing the department of an employee.
8. Perform department-wise data analysis using:
 - aggregate() function
 - dplyr package functions
9. Calculate:
 - Average salary of employees in each department
 - Total number of employees department-wise
10. Create graphical visualisations such as:
 - Bar plots showing department-wise employee count
 - Boxplots representing salary distribution across departments

Expected Outcome:

After completing this experiment, you will be able to create and manipulate data frames in R, perform filtering and aggregation operations, update structured records, analyse employee data efficiently, and generate graphical insights for real-life organisational data analysis.

Lab. Session 2: Basics of R Programming

Problem – 5: Working with Strings and Vectors for Student Performance Analysis

Problem Statement:

A school wants to analyse student performance records using R programming. The system should store student marks, names, and pass/fail status using vectors and perform string manipulation for generating personalised student messages. The school also wants to perform numerical analysis on marks and identify students scoring above a specified threshold for scholarship consideration.

Objective:

The objective of this experiment is to understand the creation, manipulation, and analysis of strings and vectors in R for real-life student data processing tasks.

Activities to be Performed:

1. Create different types of vectors to store student information such as:
 - Numeric vector for marks or scores
 - Character vector for student names
 - Logical vector for pass/fail status
2. Store student marks, names, and result status using vectors such as:
 - scores
 - names
 - passed
3. Perform string operations on student names and messages using:
 - paste() or paste0() for generating personalised result messages
 - substr() for extracting initials or portions of student names
 - tolower() and toupper() for changing text case formats
4. Perform vector operations such as:
 - Element-wise addition or subtraction of marks
 - Finding maximum marks, minimum marks, total marks, and average marks using vector functions
5. Apply logical operations to identify students scoring above a particular cutoff, such as marks greater than 80.
6. Access specific student records or ranges of marks using vector indexing.
7. Use the length() function to determine the total number of students present in the dataset.
8. Generate personalised appreciation messages for students scoring above the scholarship threshold using string concatenation functions.

Expected Outcome:

After completing this experiment, you will be able to create and manipulate vectors and strings in R, perform numerical and logical operations on real-life student datasets, process textual information efficiently, and generate meaningful outputs for data analysis and reporting purposes.

Problem – 6: Working with Lists for School Student Management System

Problem Statement:

A school management system wants to maintain complete student records using R programming. The system should store different types of information such as student names, age, subjects, marks, grades, and teacher details in a structured format. Since the information contains multiple data types, the school decides to use lists for storing and managing hierarchical student and school data efficiently.

Objective:

The objective of this experiment is to understand the creation and manipulation of lists in R for storing heterogeneous and hierarchical real-life data.

Activities to be Performed:

1. Create a list representing a student profile containing:
 - Student name
 - Age
 - List of subjects
 - Subject-wise scores
2. Store different data types together in a single list using the `list()` function.
3. Access student details using:
 - `$` operator for named elements (Example: `student$Name`)
 - Index-based access using `[[ ]]` notation
4. Modify the student list by:
 - Adding new elements such as grade, attendance, or section
 - Updating existing student information
5. Create nested lists to represent complete school data, including:
 - Multiple students
 - Teacher details
 - Different classes or departments
6. Organise hierarchical data structures such as:
 - School → Students → Subjects → Scores
7. Retrieve and display information of specific students, subjects, or teachers from nested lists.
8. Calculate and summarise average marks of students based on their subject scores.
9. Display complete student performance records in a structured and readable format.

Expected Outcome:

After completing this experiment, you will be able to create and manipulate lists in R, manage heterogeneous and nested real-life data structures, access and modify list elements efficiently, and organise hierarchical information for practical data management applications.

Problem – 7: Data Structures for Hospital Patient Management System**Problem Statement:**

A hospital wants to manage and analyse patient records using R programming. The hospital needs to store patient test results, ward-wise patient monitoring data, and staff information using matrices, arrays, and data frames for effective healthcare management and reporting.

Objective:

The objective of this experiment is to gain practical experience in creating, manipulating, and analysing matrices, arrays, and data frames using real-life hospital management data.

Activities to be Performed:

1. Create a matrix to store medical test results of patients across different test categories and access or update specific patient records.
2. Create a three-dimensional array to store patient health monitoring data based on:
 - Different hospital wards
 - Days of observation
 - Morning and evening health readings
3. Extract health records for a specific ward or day using array indexing.
4. Create a data frame containing hospital staff information such as:
 - Staff ID
 - Name
 - Department
 - Salary
5. Perform department-wise salary analysis using the `aggregate()` function.
6. Generate summaries and visualisations such as:
 - Patient test analysis using `summary()`
 - Department-wise salary distribution using bar plots or boxplots

Expected Outcome:

After completing this experiment, you will be able to create and manipulate matrices, arrays, and data frames in R, analyse structured healthcare data, perform data operations efficiently, and generate meaningful visualisations for real-life hospital management applications.

Session 3 & 4 are based on Unit 14 – Data Interfacing and Visualisation in R (refer to Block-4 Unit 14 of MCS-068)

Lab. Session 3: Data Interfacing and Visualisation in R**Problem – 8: Data Acquisition and Integration for E-Commerce Customer Analytics System****Problem Statement:**

An e-commerce company wants to analyse customer behaviour and sales performance using data collected from multiple sources. Customer details, purchase records, product reviews, preferences, transaction history, and online product information are stored in different file formats and databases. The

company wants to integrate all these datasets into a single system using R programming to generate customer-wise sales reports and business insights.

Objective:

The objective of this experiment is to understand how to acquire, integrate, and analyse data from multiple real-world sources such as CSV, Excel, XML, JSON, databases, and web services using R programming.

Activities to be Performed:

1. Import customer information such as Customer ID, Name, and Email from a CSV file.
2. Load product sales records from an Excel file containing Product ID, Customer ID, and Purchase Amount.
3. Read customer product reviews stored in XML format and extract review details.
4. Import customer preference data from a JSON file containing preferred product categories and notification settings.
5. Establish a connection with a MySQL database and retrieve customer transaction history including transaction ID, purchase date, and transaction amount.
6. Fetch additional online product information or reviews from a web source available in XML format using packages such as RCurl, XML, or rvest.
7. Merge all datasets using common fields such as Customer ID and Product ID to create a unified customer analytics dataset.
8. Handle missing or inconsistent values during the data integration process.
9. Generate customer-wise reports showing:
 - Customer Name
 - Email Address
 - Total Purchase Amount
 - Latest Product Review
10. Create summaries and basic analysis reports for customer profiling and sales analysis.

Expected Outcome:

After completing this experiment, you will be able to import and integrate data from multiple file formats and databases, retrieve web-based data, perform data consolidation and cleaning, and generate meaningful business reports using R programming for real-life e-commerce analytics applications.

Lab. Session 4: Data Interfacing and Visualisation in R

Problem – 9: Data Analysis and Visualization for Retail Store Sales Management System

Problem Statement:

A retail supermarket chain wants to analyse its customer sales data to understand purchasing behaviour, product category performance, customer demographics, and monthly sales trends. However, the sales dataset contains missing values, incorrect category labels, and abnormal sales entries. The company wants to clean and preprocess the data using R programming and generate visual reports for better business decision-making.

Objective:

The objective of this experiment is to perform data cleaning, preprocessing, analysis, and graphical visualisation of retail sales data using R programming.

Activities to be Performed:*Part A: Data Cleaning and Pre-processing*

1. Load the dataset retail_sales.csv containing:
 - Product ID
 - Product Category
 - Sale Amount
 - Sale Date
 - Customer Age
 - Customer Gender
2. Inspect the structure and summary of the dataset using functions such as str() and summary().
3. Identify and correct structural errors such as:
 - Incorrect data types
 - Mislabelled product categories
4. Detect and handle outliers in:
 - Sale Amount
 - Customer Age
5. Identify missing values and apply suitable techniques such as deletion or imputation based on data requirements.

Part B: Data Visualization

6. Create a bar chart showing total sales for each product category.
7. Generate a box plot to analyse the distribution of sales amounts across categories.
8. Create a histogram representing the age distribution of customers.
9. Plot a line graph showing monthly sales trends over time.
10. Generate a scatterplot to study the relationship between customer age and purchase amount.
11. Interpret each graph and identify business insights such as:
 - Best-performing product categories
 - Customer purchasing patterns
 - Monthly sales growth trends
 - Presence of extreme sales values

Expected Outcome:

After completing this experiment, you will be able to clean and preprocess real-life retail sales data, identify and handle missing values and outliers, generate various graphical visualisations in R, and interpret business insights from the analysed data effectively.

Lab. Session – 5: Data Analysis and R

Problem – 10: Chi-Square Test for Car Safety Feature Analysis

Problem Statement:

A car manufacturing company wants to analyse whether the type of car influences the availability of airbags in vehicles. The company uses the Cars93 dataset from the MASS package in R to study the relationship between different car categories and airbag availability. The management wants to determine whether both variables are statistically associated for improving vehicle safety planning and marketing decisions.

Objective:

The objective of this experiment is to apply the Chi-Square test in R to determine whether there is a significant association between two categorical variables i.e. different car categories and airbag availability.

Activities to be Performed:

1. Load the MASS package and import the Cars93 dataset into R.
2. Extract the variables:
 - Type (Type of car)
 - Airbags (Availability of airbags)
3. Create a contingency table showing the frequency distribution between car type and airbag availability.
4. Apply the Chi-Square test using the `chisq.test()` function.
5. Analyse and interpret:
 - Chi-Square statistic
 - p-value
6. Draw a conclusion regarding whether car type and airbag availability are dependent or independent variables.

Expected Outcome:

After completing this experiment, you will be able to perform the Chi-Square test in R, analyse relationships between categorical variables, and interpret statistical results. Students will conclude that if the p-value is less than 0.05, there is a statistically significant association between car type and airbag availability.

Problem – 11: Linear Regression for Fitness Center Health Analysis

Problem Statement:

A fitness center wants to analyse the relationship between the height and weight of its members to better understand body fitness patterns and provide personalized health recommendations. The management wants to develop a predictive model using R programming that can estimate a person's weight based on their height using linear regression analysis.

Objective:

The objective of this experiment is to learn how to build and interpret a simple linear regression model in R for predicting one variable based on another variable.

Activities to be Performed:

1. Create or import a dataset containing:
 - Height of fitness center members
 - Weight of fitness center members
2. Build a simple linear regression model using:
`lm(weight ~ height)`
3. Display and analyse the regression summary to interpret:
 - Regression coefficients
 - Relationship between height and weight
 - Model significance
4. Predict weight values for new height measurements using the `predict()` function.
5. Plot:
 - Scatterplot of height and weight data
 - Regression line
 - Residual plots for model analysis
6. Interpret the regression model and evaluate how accurately height predicts weight.

Expected Outcome:

After completing this experiment, you will be able to build and interpret a simple linear regression model in R, understand the influence of height on weight, generate predictive equations, and evaluate model performance using graphical and statistical analysis.

Lab. Session – 6: Data Analysis and R

Problem – 12: Multiple Regression for Automobile Fuel Efficiency Analysis

Problem Statement:

An automobile manufacturing company wants to analyse how different vehicle characteristics affect fuel efficiency. The company uses the `mtcars` dataset in R to study the impact of engine displacement, horsepower, and vehicle weight on car mileage (`mpg`). The goal is to develop a predictive model that helps the company design fuel-efficient vehicles and understand which factors most strongly influence mileage.

Objective:

The objective of this experiment is to apply multiple regression analysis in R to predict car mileage using multiple independent variables.

Activities to be Performed:

1. Load the `mtcars` dataset in R containing vehicle performance details.
2. Select the following variables:
 - `mpg` → Fuel efficiency (miles per gallon)
 - `disp` → Engine displacement
 - `hp` → Horsepower
 - `wt` → Vehicle weight

3. Build a multiple linear regression model using:
`lm(mpg ~ disp + hp + wt, data = mtcars)`
4. Generate the regression summary and interpret:
 - Regression coefficients
 - p-values
 - Significance of predictors
 - Overall model fit
5. Formulate the regression equation for predicting mileage.
6. Predict mileage values for new vehicle data using the `predict()` function.
7. Analyse how engine displacement, horsepower, and weight influence fuel efficiency.

Expected Outcome:

After completing this experiment, you will be able to apply multiple regression analysis in R, interpret the influence of multiple predictors on fuel efficiency, generate predictive regression equations, and evaluate the performance of regression models for real-life automobile data analysis.

Problem – 13: Logistic Regression for Vehicle Transmission Classification

Problem Statement:

An automobile company wants to predict whether a car should be classified as having a manual or automatic transmission based on vehicle characteristics such as horsepower, weight, and number of cylinders. The company uses the `mtcars` dataset in R to develop a classification model that helps in understanding how vehicle specifications influence transmission type selection.

Objective:

The objective of this experiment is to apply logistic regression in R for binary classification and analyse how different vehicle parameters affect transmission type.

Activities to be Performed:

1. Load the `mtcars` dataset containing automobile specifications.
2. Select the following variables:
 - `am` → Transmission type (Manual/Automatic)
 - `hp` → Horsepower
 - `wt` → Vehicle weight
 - `cyl` → Number of cylinders
3. Build a logistic regression model using:
`glm(am ~ hp + wt + cyl, data = mtcars, family = binomial)`
4. Generate and interpret the model summary, including:
 - Regression coefficients
 - p-values
 - Significant predictor variables
 - Odds ratio interpretation
5. Compute predicted probabilities for transmission type classification.
6. Classify vehicles as manual or automatic based on predicted probabilities.
7. Compare predicted transmission types with actual transmission data to evaluate model accuracy.

Expected Outcome:

After completing this experiment, you will be able to apply logistic regression for binary classification in R, interpret classification model outputs, analyse significant predictor variables, compute predicted probabilities, and evaluate model accuracy using real-life automobile data. Students will also observe that vehicle weight often plays a major role in predicting transmission type.

Lab. Session – 7: Data Analysis and R

Problem – 14: Time Series Analysis for Stock Market Trend Analysis

Problem Statement:

A financial analytics company wants to study stock market behaviour to identify long-term trends, seasonal patterns, and irregular fluctuations in stock index values. The company uses the EuStockMarkets dataset in R and focuses on analysing the DAX stock market index. The goal is to decompose stock market data into meaningful components and understand how past market performance influences future trends for investment analysis and forecasting.

Objective:

The objective of this experiment is to perform time series analysis in R by creating time series objects, extracting trend and seasonal components, and analysing lagged relationships in stock market data.

Activities to be Performed:

1. Load the EuStockMarkets dataset in R and extract the DAX stock market index values.
2. Convert the DAX data into a time series object using the ts() function.
3. Analyse the behaviour of the time series and identify whether the data is stationary or non-stationary.
4. Apply the decompose() function to separate the time series into:
 - Trend component
 - Seasonal component
 - Irregular/error component
5. Use the stl() function for advanced seasonal-trend decomposition and smoother analysis.
6. Create lag variables using lag() or DataCombine::slide() to study the effect of previous stock values on future market movements.
7. Visualise and interpret the trend, seasonality, and lag patterns in the stock market data.

Expected Outcome:

After completing this experiment, you will be able to perform time series analysis in R, understand trend and seasonal behaviour in stock market data, apply decomposition techniques, generate lagged variables, and analyse how historical market patterns influence future stock trends.

Problem – 15: Integrated Data Analysis for Automobile Industry Decision Support System

Problem Statement: An automobile manufacturing company wants to improve vehicle safety, fuel efficiency, transmission design, and market performance analysis using data analytics. The company has appointed a data analyst to study vehicle specifications, safety features, customer-oriented vehicle performance, and automobile market trends using R programming. The analyst must apply statistical and predictive techniques to help the company make better decisions related to vehicle manufacturing, product planning, and market forecasting.

Objective: The objective of this experiment is to perform integrated data analysis in R using hypothesis testing, regression analysis, classification techniques, and time series analysis for solving real-life automobile industry problems.

Activities to be Performed:

1. Chi-Square Test – Vehicle Safety Analysis

- Use the Cars93 dataset from the MASS package.
- Analyse whether airbag availability depends on the type of car.
- Apply the `chisq.test()` function to test the association between:
 - Car Type
 - Airbag Availability
- Interpret whether luxury or sports vehicles provide better safety features compared to regular cars.

2. Linear Regression – Vehicle Performance Analysis

- Use automobile performance data containing:
 - Vehicle Weight
 - Fuel Consumption
- Build a simple linear regression model using: `lm(mpg ~ wt)`
- Analyse how vehicle weight affects fuel efficiency.
- Predict mileage values for vehicles with different weights.

3. Multiple Regression – Fuel Efficiency Prediction

- Use the `mtcars` dataset.
- Predict fuel efficiency (mpg) using multiple vehicle parameters:
 - Engine displacement (disp)
 - Horsepower (hp)
 - Vehicle weight (wt)
- Apply: `lm(mpg ~ disp + hp + wt, data = mtcars)`
- Identify which automobile features have the strongest impact on mileage.

4. Logistic Regression – Transmission Type Classification

- Use the `mtcars` dataset.
- Predict whether a vehicle has:
 - Manual transmission
 - Automatic transmission
- Use predictor variables:
 - Horsepower (hp)

- Weight (wt)
- Number of cylinders (cyl)
- Apply logistic regression using: `glm(..., family = binomial)`
- Analyse which specifications determine transmission type.

5. Time Series Analysis – Automobile Market Trend Analysis

- Use the EuStockMarkets dataset and analyse the DAX automobile-related market trend data.
- Convert stock index values into a time series object using `ts()`.
- Analyse:
 - Long-term market trend
 - Seasonal fluctuations
 - Short-term irregular variations
- Apply:
 - `decompose()`
 - `stl()`
 - `lag()`
- Study how historical automobile market performance influences future market trends and investment decisions.

Expected Outcome: After completing this experiment, you will be able to:

- Perform Chi-Square testing for analysing categorical relationships in vehicle safety data.
- Apply regression techniques for predicting vehicle fuel efficiency.
- Use logistic regression for automobile transmission classification.
- Perform time series analysis to study automobile market trends and seasonal behaviour.
- Interpret statistical and predictive results for real-life automobile industry decision-making.

Session 8,9 & 10 are based on Unit 16 – Advanced Analysis using R

(refer to Block-4 Unit 16 of MCS-068)

Lab. Session – 8: Advanced Analysis using R

Problem – 16: Decision Tree Analysis for Language Learning Assessment System

Problem Statement:

An educational institute wants to identify whether a student is a native English speaker or a non-native speaker based on student characteristics such as age, reading ability, and other learning-related attributes. The institute plans to use decision tree classification in R programming to automatically classify students and support language training and assessment programs.

Objective:

The objective of this experiment is to understand how decision trees are used for supervised classification, data partitioning, and rule-based decision-making in R programming.

Activities to be Performed:

1. Load the party package and import the readingSkills dataset in R.

2. Analyse the following variables:
 - age
 - shoeSize
 - score (reading score)
 - nativeSpeaker (classification output)
3. Build a decision tree model using:
`ctree(nativeSpeaker ~ age + shoeSize + score, data = readingSkills)`
4. Visualise the generated decision tree and analyse:
 - Decision nodes
 - Splitting conditions
 - Classification paths
5. Interpret how variables such as reading score and age influence native speaker classification.
6. Test the model using sample student values and predict whether the student is likely to be a native or non-native speaker.

Expected Outcome:

After completing this experiment, you will be able to build and visualise decision tree models in R, understand how classification rules are generated, interpret tree structures and decision paths, and apply supervised learning techniques for real-life educational data analysis and student classification systems.

Problem – 17: Random Forest Classification for Language Proficiency Assessment System

Problem Statement:

An educational institute wants to improve the accuracy of its student language classification system. Previously, the institute used a decision tree model to classify students as native or non-native English speakers based on factors such as age, shoe size, and reading score. To achieve more accurate and stable predictions, the institute now plans to use the Random Forest algorithm and compare its performance with the existing decision tree model.

Objective:

The objective of this experiment is to understand ensemble learning and build a Random Forest classification model in R for improving prediction accuracy and identifying important features influencing student language classification.

Activities to be Performed:

1. Load the randomForest package and import the readingSkills dataset in R.
2. Build a Random Forest classification model using:
`randomForest(nativeSpeaker ~ age + shoeSize + score, data = readingSkills)`
3. Generate and analyse the confusion matrix to evaluate model accuracy and classification performance.
4. Compare the prediction accuracy of:
 - Decision Tree model
 - Random Forest model
5. Analyse feature importance using the MeanDecreaseGini measure to identify the most influential variables affecting classification.
6. Interpret how factors such as reading score, age, and shoe size contribute to predicting whether a student is a native or non-native speaker.

Expected Outcome:

After completing this experiment, you will be able to build and evaluate Random Forest classification models in R, understand the concept of ensemble learning and bagging, compare classification performance with decision trees, interpret confusion matrices, and identify important predictor variables for real-life educational data analysis applications.

Lab. Session – 9: Advanced Analysis using R**Problem – 18: Flower Species Classification for Botanical Research System****Problem Statement:**

A botanical research institute wants to develop an automated flower classification system to identify different species of iris flowers based on their physical characteristics. The institute plans to use machine learning techniques in R programming to classify flowers into three species — Setosa, Versicolor, and Virginica — using measurements such as sepal length, sepal width, petal length, and petal width. The researchers also want to compare the performance of Logistic Regression and Naive Bayes classifiers for accurate species prediction.

Objective:

The objective of this experiment is to understand supervised classification techniques and compare the performance of Logistic Regression and Naive Bayes classifiers using real-life flower classification data.

Activities to be Performed:

1. Load the iris dataset containing flower measurement details and species information.
2. Split the dataset into:
 - Training dataset
 - Testing dataset
3. Build a Logistic Regression classification model using flower features such as:
 - Sepal Length
 - Sepal Width
 - Petal Length
 - Petal Width
4. Build a Naive Bayes classification model using:
naiveBayes(Species ~ ., data = iris)
5. Predict flower species using both classification models.
6. Generate and analyse confusion matrices to compare classification accuracy and model performance.
7. Interpret which classification technique provides better prediction results for flower species identification.

Expected Outcome:

After completing this experiment, you will be able to apply supervised classification techniques in R, understand multi-class classification problems, build Logistic Regression and Naive Bayes models, evaluate model accuracy using confusion matrices, and compare classifier performance for real-life botanical research applications.

Problem – 19: Flower Grouping Analysis Using K-Means Clustering

Problem Statement:

A botanical research institute wants to automatically group different iris flowers based on their physical characteristics without using predefined species labels. The researchers aim to identify natural groupings of flowers using machine learning techniques in R programming. The clustering results will then be compared with the actual flower species to evaluate how effectively the algorithm identifies similar flower groups.

Objective:

The objective of this experiment is to understand unsupervised learning techniques and apply K-Means clustering in R for grouping similar flower species based on their measurements.

Activities to be Performed:

1. Load the iris dataset and select numerical flower measurement attributes:
 - Sepal Length
 - Sepal Width
 - Petal Length
 - Petal Width
2. Apply K-Means clustering using:
`kmeans(iris[,1:4], centers = 3, nstart = 20)`
3. Generate and visualise flower clusters using:
 - `clusplot()`
 - `ggplot2`
4. Compare the generated clusters with the actual iris species labels using a confusion matrix.
5. Analyse similarities and overlaps between clusters, especially among:
 - Versicolor
 - Virginica

Expected Outcome:

After completing this experiment, you will be able to apply K-Means clustering in R, understand unsupervised learning concepts, visualise cluster formation, and compare clustering results with actual class labels. Students will also observe that K-Means can successfully identify natural flower groupings, although some overlap may occur between similar species such as Versicolor and Virginica.

Lab. Session – 10: Advanced Analysis using R

Problem – 20: Market Basket Analysis for Supermarket Product Recommendation System

Problem Statement:

A supermarket chain wants to analyse customer purchasing behaviour to identify products that are frequently bought together. The management plans to use association rule mining in R programming to discover hidden buying patterns from transaction data. These insights will help the supermarket improve product placement, cross-selling strategies, and recommendation systems for customers.

Objective:

The objective of this experiment is to understand association rule mining concepts such as support, confidence, and lift, and apply the Apriori algorithm in R for market basket analysis.

Activities to be Performed:

1. Load the arules and arulesViz packages in R.
2. Import the Groceries transaction dataset containing customer purchase records.
3. Apply the Apriori algorithm using:
`apriori(Groceries, parameter = list(support = 0.01, confidence = 0.3))`
4. Generate and inspect the top association rules sorted by lift value.
5. Analyse product combinations that are frequently purchased together, such as:
 - Bread → Butter
 - Milk → Cereal
6. Visualise association rules using the `plot()` function for better interpretation of customer buying patterns.
7. Interpret the meaning of:
 - Support
 - Confidence
 - Lift

Expected Outcome:

After completing this experiment, you will be able to perform association rule mining in R, apply the Apriori algorithm, interpret support, confidence, and lift values, and identify frequently purchased product combinations. You will also understand how market basket analysis is used in supermarkets and e-commerce platforms to build recommendation systems and improve sales strategies.

2.6 SUMMARY

Predictive Data Analysis Lab is based on R Programming, and it focuses on developing practical skills in analysing data and building predictive models using R. This section is based on the concepts covered in MCS-068: Predictive Data Analysis and is designed in a hands-on, self-learning format through 10 lab sessions. Each lab session helps you to apply theoretical concepts to real datasets, making the learning process more practical and understandable.

This section begins with basic data handling and gradually moves towards advanced analytical techniques. The lab sessions cover important topics such as similarity and distance measures, outlier detection methods, and different types of predictive models. You get practical exposure on regression techniques, time series analysis, and forecasting methods. You also learn supervised learning models like K-Nearest Neighbours, Naïve Bayes, decision trees, and support vector machines, along with unsupervised learning methods such as clustering and association rules.

The section also emphasizes model evaluation, where to use performance metrics like accuracy, precision, recall, and ROC curves to assess the effectiveness of the models. In addition, you were given a strong understanding of R programming, including data manipulation, visualisation, and statistical analysis.

Overall, this section helps you to move from basic data understanding to making predictions and informed decisions. It builds confidence in using R for real-world data analysis and prepares you for practical applications in data science and analytics.