Experiment No: 05

AIM: Write a program TO LOAD AND READ THE DATA , specify file name from the subfolders

Objectives:

- ➔ Students will be able to implement program to load and read the data in R
- ➔ Students will be able to specify file name from the sub-folders in R

Software Requirements: R Studio, VS Code

Theory:

R is a popular programming language that's widely used for statistical computing and data analysis. One of its key strengths lies in its ability to work with data from various sources, including the popular CSV (Comma-Separated Values) format.

R provides several ways to load and read data from CSV files. The most common function is read.csv from the base package "utils." This function takes the file path as an argument and, by default, assumes the first row contains column names and the comma (",") acts as the field separator. You can customize these settings by specifying additional arguments like header and sep respectively.

Functions and Syntax of Reading files in R:

The read.csv function in R offers a variety of options:

- ➔ header: Control if the first row contains column names (default: TRUE). Setting it to FALSE treats all rows as data
- ➔ sep: Specify the field separator (default: ","). This allows handling files with different delimiters, like tabs ("\t") or semicolons (";")
- ➔ nrows: Restrict the number of rows read (useful for large files or preliminary exploration)
- ➔ skip: Skip a specific number of header rows (useful for files with multiple header rows)
- ➔ colClasses: Define the data type for each column (e.g., colClasses = c(character(), numeric(), logical()))
- ➔ na.strings: Define strings to be treated as missing values (e.g., na.strings = c("NA", "N/A"))

The most common use case for read.csv in R is for statistical analysis. You can import various datasets, from scientific experiments or surveys, stored in CSV format for statistical analysis. The data can then be used for tasks like:

- ➔ Calculating summary statistics: Finding measures like mean, median, and standard deviation.
- ➔ Creating visualizations: Plotting histograms, scatterplots, and other charts to explore data patterns.
- ➔ Performing hypothesis testing: Testing hypotheses about relationships between variables.

Code and Output:

```
# Reads data from csv
data <- read.csv("stud.csv")

# Prints contents of csv
print(head(data,10))

#add new entry
new_data <- data.frame('Student No' = 202, 'Average Marks' = 79)

#Combining Dataframes
datac <- rbind(data,new_data)

# Writing to dataset
write.csv(datac, file = "new_file.csv")

print(tail(datac,10))
```

```
> source("d:\\Documents\\codes\\SBL R\\e5.r", encoding = "UTF-8")
   Student.No Average.Marks
1           1            87
2           2            92
3           3            78
4           4            65
5           5            90
6           6            55
7           7            83
8           8            76
9           9            69
10         10            94
     Student.No Average.Marks
192         192            83
193         193            80
194         194            93
195         195            59
196         196            98
197         197            49
198         198            71
199         199            79
200         200            90
201         202            79
```

Conclusion:

In this way in R programming,we explored the essential functions read.csv and write.csv. The read.csv function allows us to import data from structured CSV files directly into R, enabling further exploration, manipulation, and modeling. Conversely, write.csv allows us to save processed data from R into CSV format, facilitating data sharing, collaboration, and integration with external tools or reporting systems. In the context of data analysis these two functions are crucial to the R programming language as they allow the system to read and write to multiple datasets which contain the information required for statistical analysis