**R PROGRAMMING - LAB**

(COURSE CODE: 23UPCSC4E13)

A Laboratory record Submitted to Periyar University,Salem. In partial fulfilment of the Requirements for the Degree of

**MASTER OF SCIENCE**

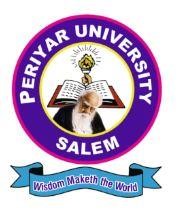
**IN**

**DATA SCIENCE**

BY

**AJITHKUMAR RK**

**REG NO:** **U23PG507DTS003**



**DEPARTMENT OF COMPUTER SCIENCE**

**PERIYAR UNIVERSITY**

PERIYAR PALKALAI NAGAR,

SALEM – 636011.

CERTIFICATE

This is to certify that the Programming Laboratory entitled **“R PROGRAMMING - LAB”** (23UPCSC4E13) is a bonafide record work done by **Mr.AJITHKUMAR RK** Register No.**U23PG507DTS003** as partial fulfilment of the requirements for a degree of MASTER OF SCIENCE IN DATA SCIENCE in the Department of Computer Science, Periyar University, Salem, during the year 2023 – 2025

Faculty In-Charge Head of the Department

Submitted for the practical examination held on ……. /……. /………………

Marks Obtained

|  |
| --- |
|  |
|  |

Internal Examiner External Examiner

**Certificate**



# R PROGRAMMING INDEX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **DATE** | **TITTLE** | **PAGE.NO** | **SIGNATURE** |
| 1 | 2.01.2024 | Installing R on Windows, and R Studio | 1 |  |
| 2 | 09.01.2024 | Data Types - R Objects and Attributes, Data Types - Vectors and Lists | 4 |  |
| 3 | 23.01.2024 | Data Structure - Data Frames, Matrices, Factors & Lists | 7 |  |
| 4 | 30.01.2024 | Functions Expression & Logical Statement in R | 13 |  |
| 5 | 06.02.2024 | Subsetting of List, Matrices, &Data frame | 19 |  |
| 6 | 13.02.2024 | Data frame functions on inbuilt Dataset | 24 |  |
| 7 | 20.02.2024 | Dplyr Function on Dataset | 29 |  |
| 8 | 27.02.2024 | Implementation the following in R studio | 33 |  |
| 9 | 5.03.2024 | Basic Plotting with R | 37 |  |
| 10 | 12.03.2024 | GGplots with R | 41 |  |
| 11 | 19.03.2024 | Working with R Markdown & pushing Code to Git | 45 |  |
| 12 | 26.03.2024 | Building R Shiny Dashboard App | 47 |  |

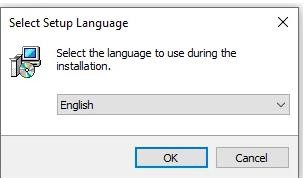
|  |  |
| --- | --- |
| **Ex. No:1** | **INSTALLING R ON WINDOWS, AND R STUDIO** |
| **Date:02.01.2024** |

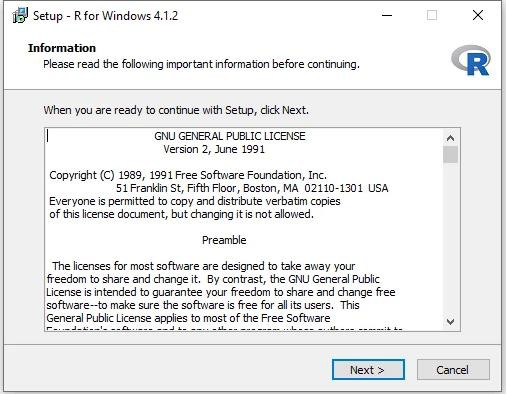
**AIM:**

To know he process for installing R on windows and R Studio.

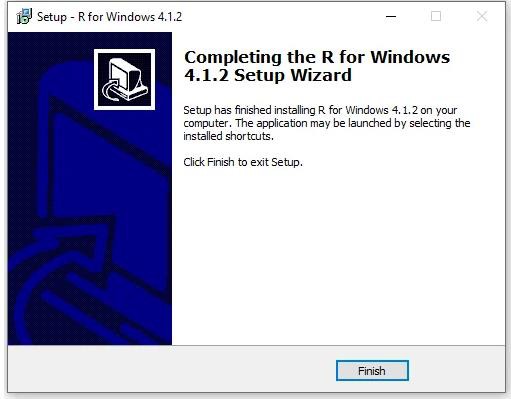
# PROCEDURE:

**To install R in windows OS:**

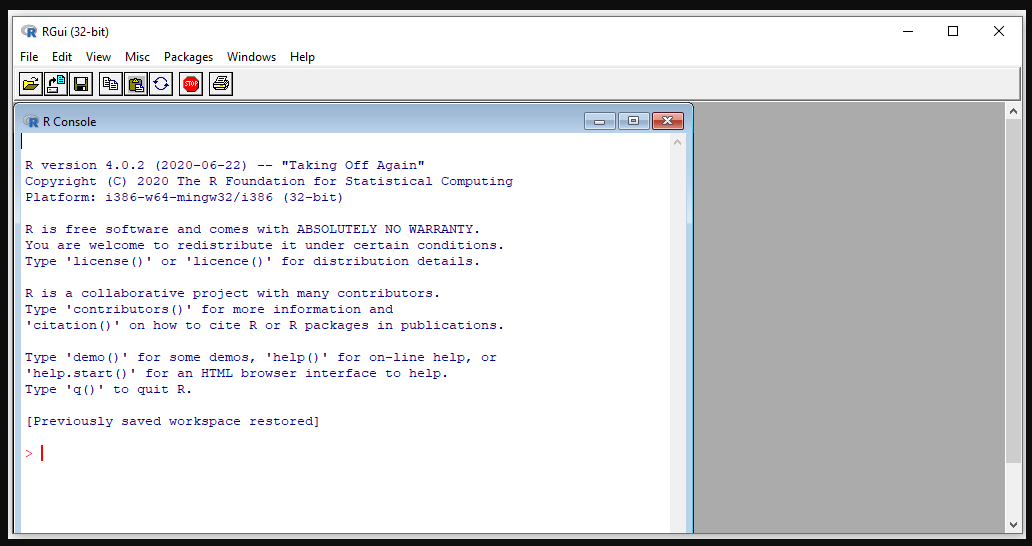
1. Go to the [CRAN](https://cran.r-project.org/) website.
2. Click on **"Download R for Windows"**.
3. Click on "install R for the first time" link to download the R executable (.exe) file.
4. Run the R executable file to start installation and allow the app to make changes to your device.
5. Select the installation language.
6. **Follow the installation instructions.**



1. **Click on "Finish" to exit the installation setup.**

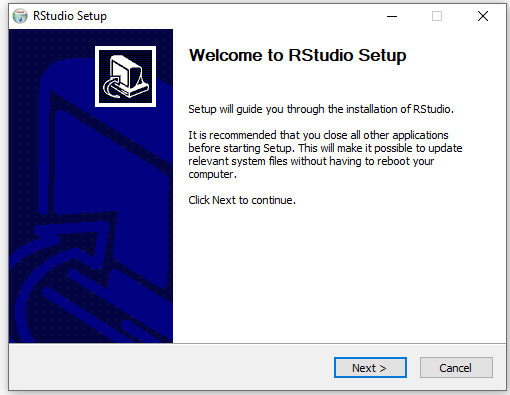


R has now been successfully installed on your Windows OS. Open the R GUI to start writing R codes.

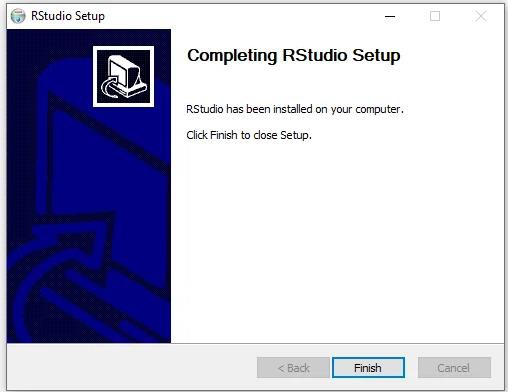


**To install RStudio Desktop on your computer, do the following:**

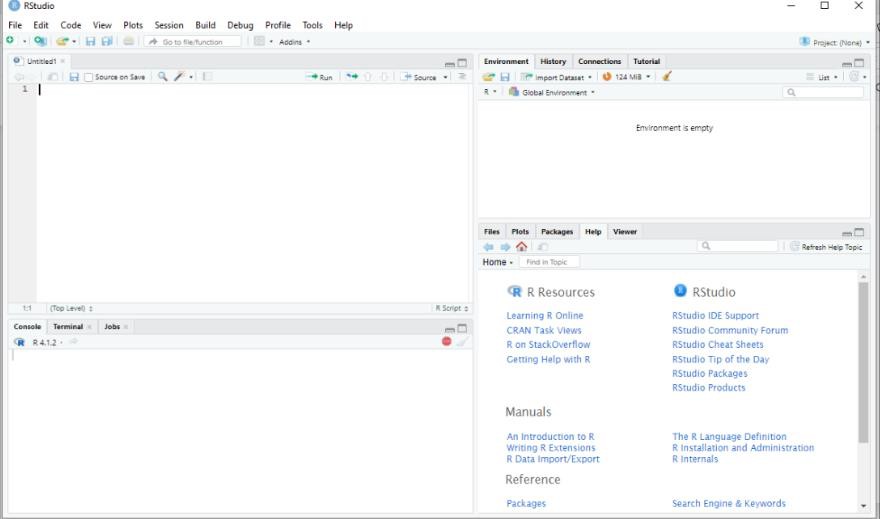
1. Go to the [RStudio](https://www.rstudio.com/) website.
2. Click on **"DOWNLOAD"** in the top-right corner.
3. Click on **"DOWNLOAD"** under the **"RStudio Open Source License"**.
4. Download RStudio Desktop recommended for your computer.
5. Run the RStudio Executable file (.exe) for Windows OS or the Apple Image Disk file (.dmg) for macOS X.



1. Follow the installation instructions to complete RStudio Desktop installation.



RStudio is now successfully installed on your computer. The RStudio Desktop IDE interface is shown in the figure below:



Another way to inteface with R using RStudio is with the *RStudio Server*. RStudio Server provides a browser-based R interface.

## RESULT:

Thus, R and R studio was installed in windows successfully.

|  |  |
| --- | --- |
| **Ex. No:2** | **DATA TYPES - R OBJECTS AND ATTRIBUTES, DATA TYPES - VECTORS AND LISTS** |
| **Date:09.01.24** |

# AIM:

To know the Data Types - R Objects and Attributes, Data Types - Vectors and Lists

# PROCEDURE:

* 1. About Objects 2.Attributes

3.Data types

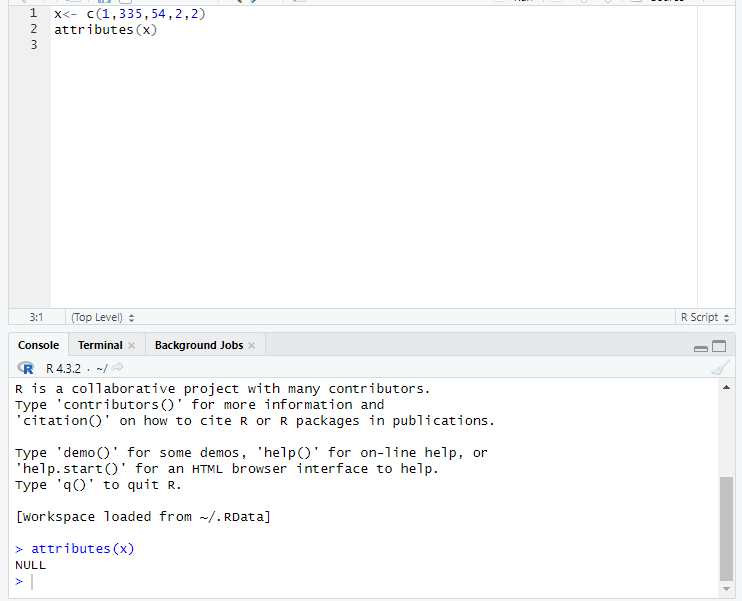
1. **About Objects :**

R has many objects :

* 1. Atomic vectors
  2. Attributes
  3. Matrices
  4. Arrays
  5. Class
  6. Coercion
  7. Lists
  8. Data Frame
  9. Loading Data
  10. Saving Data
  11. Summary

1. **Attributes:**

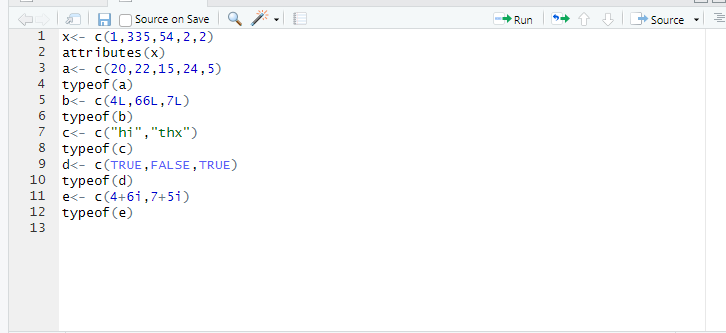
An attribute is a piece of information that you can attach to an atomic vector (or any R object). R uses NULL to represent the null set, an empty object.



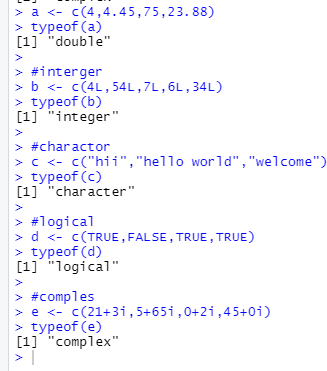
1. **Data types in R :**

R’s basic data types are character, double, integer, complex, and logical.

**INPUT:**



**OUTPUT:**



# RESULT:

Thus, the code was executed successfully and verified the results.

|  |  |
| --- | --- |
| **Ex.No:3** | **DATA STRUCTURE - DATA FRAMES, MATRICES, FACTORS & LISTS** |
| **Date:23.01.24** |

# AIM:

To Know About Data Structure - Data Frames, Matrices, Factors & Functions

# PROCEDURE:

R’s basic data structures include the vector, list, matrix, data frame, and factors.

* 1. Data Frames
  2. Matrices
  3. Factors
  4. list

1. **Data Frames**

create a new data frame with data.frame() function.

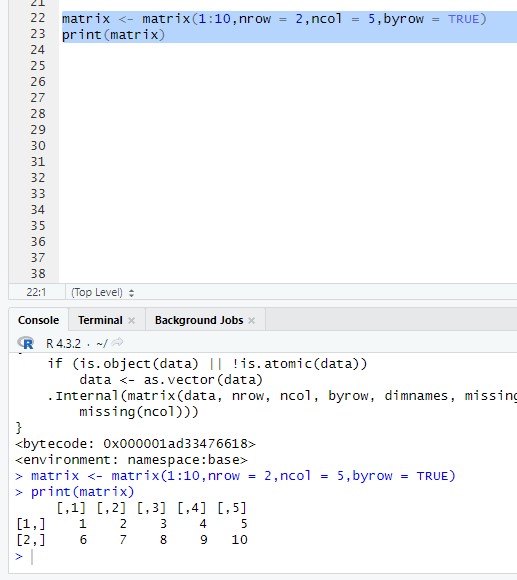
A computer screen shot of a computer

Description automatically generated

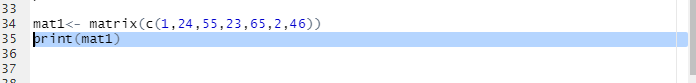
**>>> Some useful Data Frame functions**

* + head() - shows first 6 rows
  + tail() - shows last 6 rows
  + dim() - returns the dimensions of data frame (i.e. number of rows and number of columns)
  + nrow() - number of rows
  + ncol() - number of columns
  + str() - structure of data frame - name, type and preview of data in each column
  + names() or colnames() - both show the names attribute for a data frame
  + sapply(dataframe, class) - shows the class of each column in the data frame

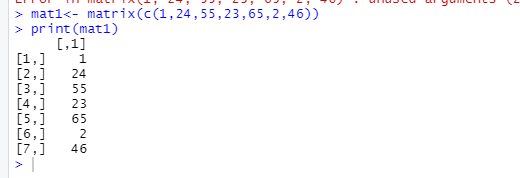
1. **Matrices**
   * A matrix is a two dimensional data set with columns and rows.
   * created with the matrix() function.



**Input:**

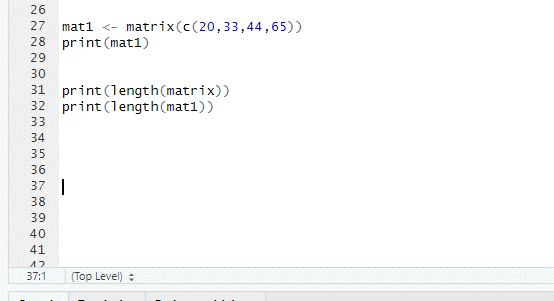


**Output:**

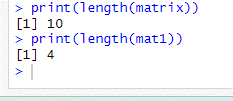


**Matrix length:**

Use the length() function to find the dimension of a Matrix Input:



Output:

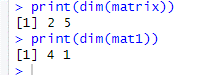


Use the dim() function to find the number of rows and columns in a Matrix Input:

A blue and white rectangular object

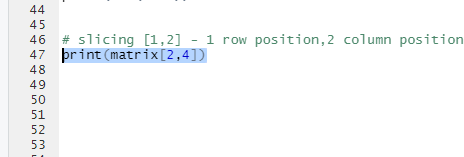
Description automatically generated

Output:



**Slicing:**

**Input:**



Output:

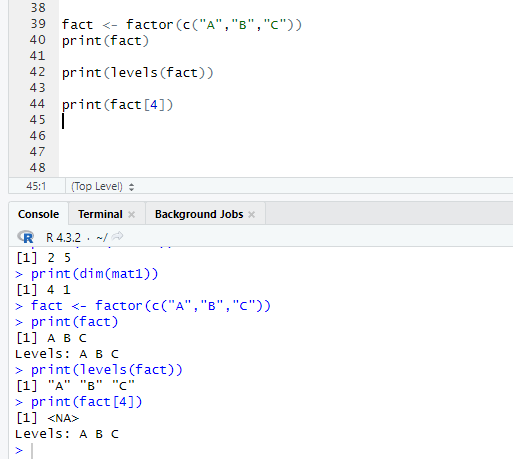


1. **Factors**

Factors are used to categorize data. Examples of factors are:

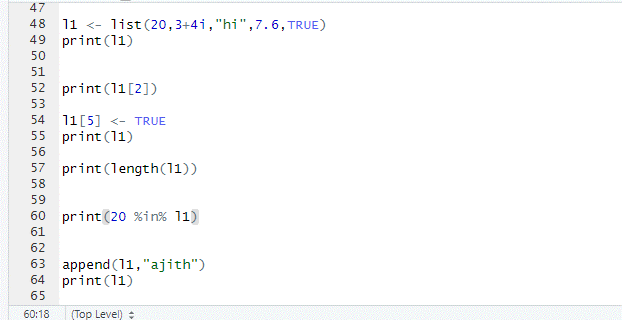
* + Demography: Male/Female
  + Music: Rock, Pop, Classic, Jazz
  + Training: Strength, Stamina

To create a factor, use the factor() function and add a vector as argument:

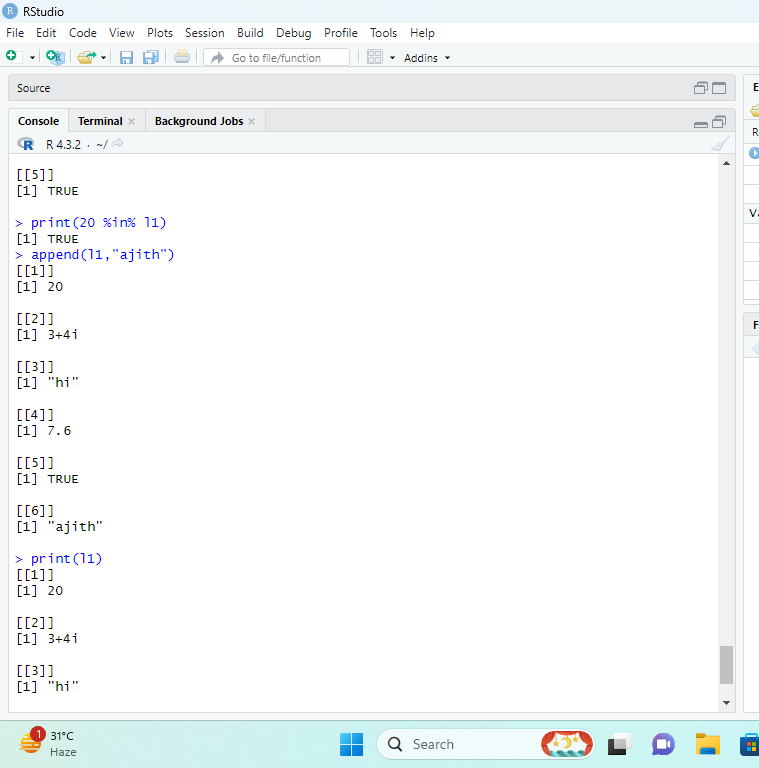


1. **list**

A list in R can contain many different data types inside it. A list is a collection of data which is ordered and changeable.

**Input:** 

**Output:**



# RESULT:

Thus, the program was executed successfully, and the result is verified.

|  |  |
| --- | --- |
| **Ex.No:4** | **FUNCTION EXPRESSIONS & LOGICAL STATEMENT R** |
| **Date:30.01.24** |

# AIM:

To write program for Functions Expressions & logical statement R.

# PROCEDURE:

1. Function Expressions
2. Logical statement
3. Comparison statement
4. Conditional statement
5. **Function Expressions:**

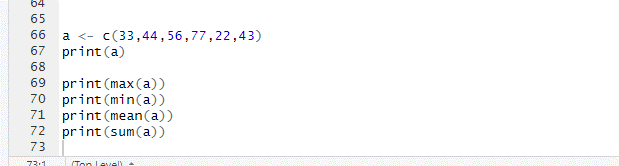
R has many **in-built** functions which can be directly called in the program without defining them first. We can also create and use our own functions referred as **user defined** functions.

**Built-in Function:**

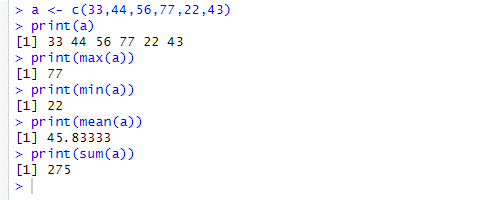
Simple examples of in-built functions are **seq()**, **mean()**, **max()**, **sum(x)** and **paste(...)** etc.

They are directly called by user written programs

**Input:**



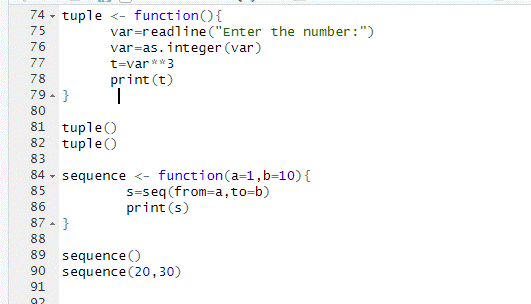
**Output:**



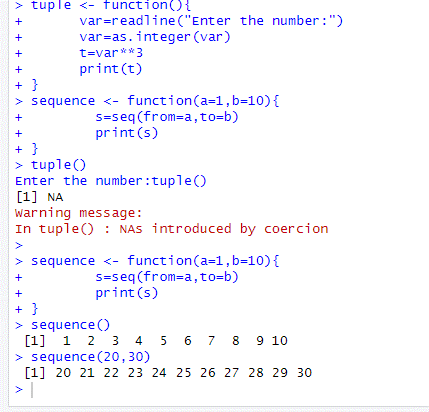
**User define functions:**

We can create user-defined functions in R. They are specific to what a user wants and once created they can be used like the built-in functions. Use the key function().

**Input:**



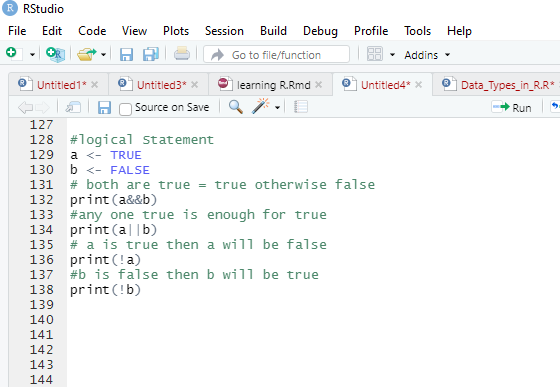
**Output:**



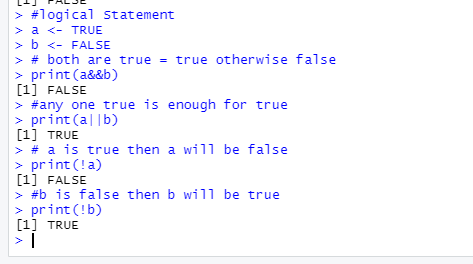
1. **logical statement:**

Logical operators are &, && - and, |, || - or,! – not.

**Input:**



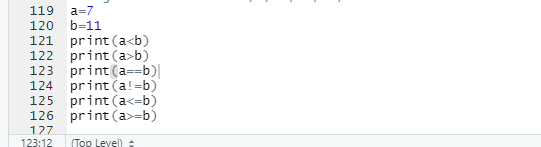
**Output:**



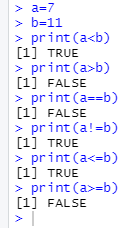
1. **comparison statement:**

Comparison operators are < - less than, > - greater than, <= - less than or equal, >= - greater than or equal, == - equal, != - not equal.

**Input:**

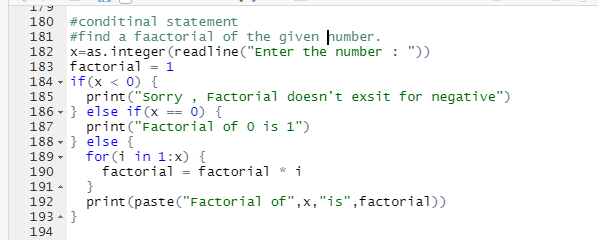


**Output:**

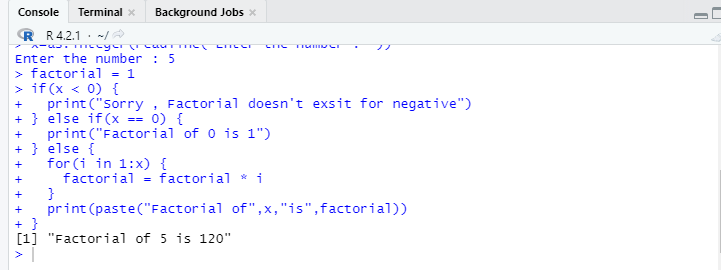


1. **conditional statement:**

**Input:**



**Output:**



## RESULT:

Thus, the function and statements of R is run successfully.

|  |  |
| --- | --- |
| **Ex. No:5** | **SUBSETTING OF LIST, MATRICES & DATA FRAME** |
| **Date:06.02.24** |

# AIM:

To subset the list, Matrices & Data Frame.

# PROCUDURE:

* 1. Subsetting list 2.subsetting matrices

3.subsetting Data Frame

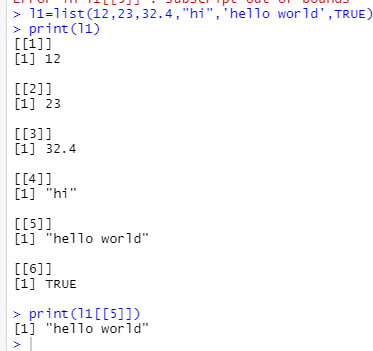
1. **Subsetting list:**
   * **Extract a single element from a list.**

We can extract using an index number. Indexing starts with 1

**Input:**



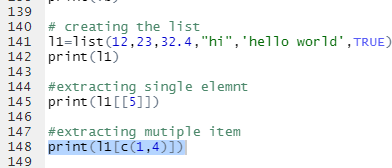
**Output:**



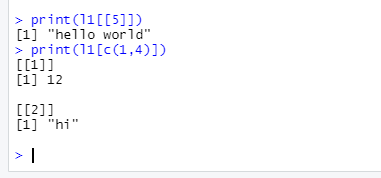
* + **Extract multiple items from list**

We can extract multiple items using the c().

**Input:**



**Output:**

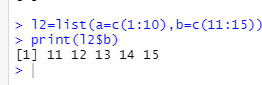


* + **Extracting with $ .**

Input:

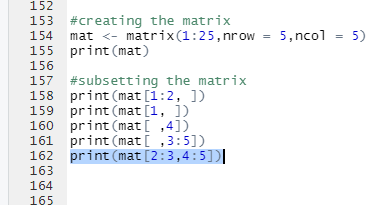


Output:

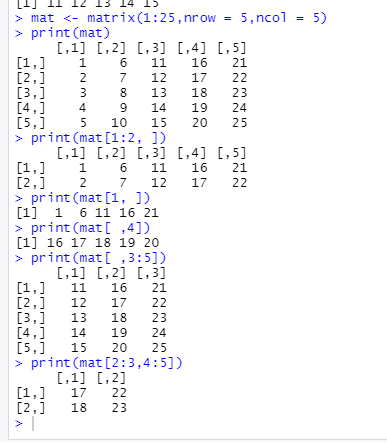


1. **subsetting the Matrices:**

**Input:**



**Output:**

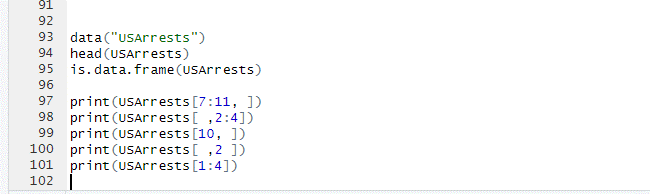


1. **subsetting of Data Frame**

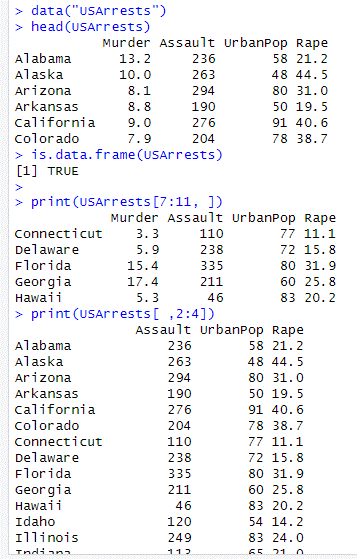
Data frames have the characteristics of both lists and matrices:

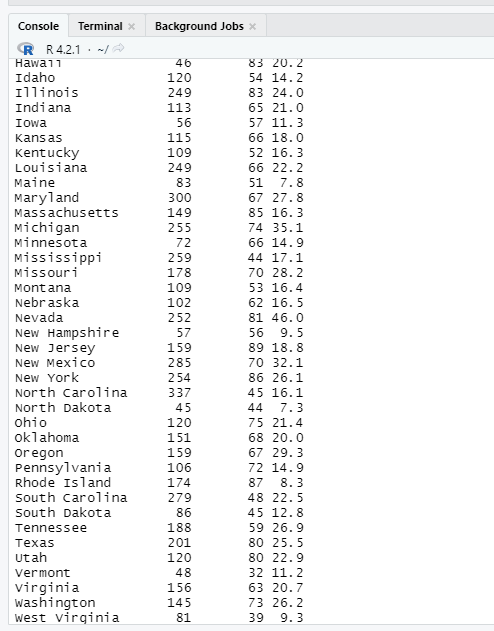
* When subsetting with a single index, they behave like lists and index the columns, so USArrestsf [1:2] selects the first two columns.
* When subsetting with two indices, they behave like matrices, so USArrests [1:3,] selects the first three *rows* (and all the columns)

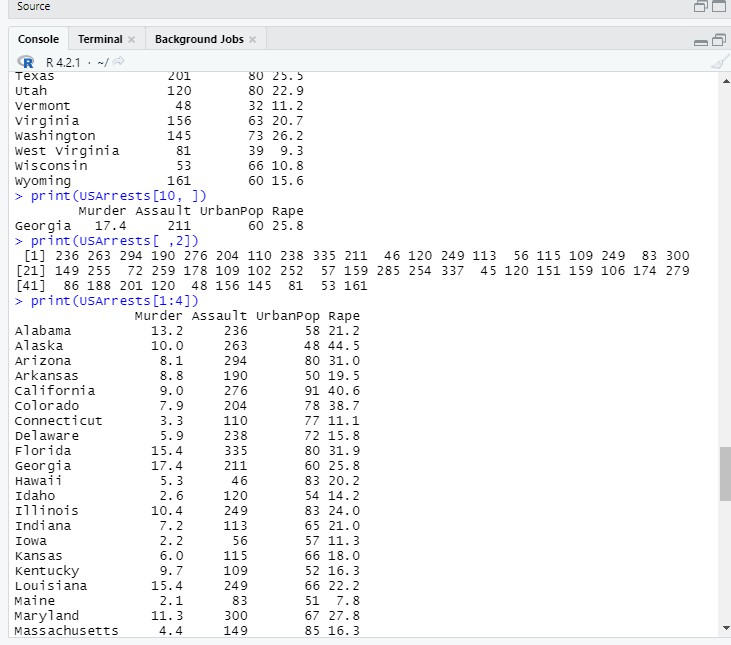
Input:



Output:







## RESUT:

Thus ,the sub setting of list , matrices and data frame are run successfully.

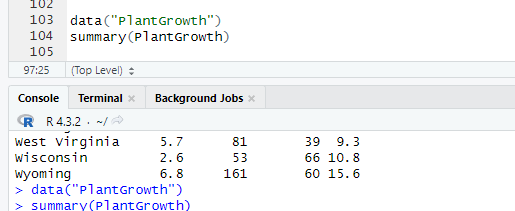
|  |  |
| --- | --- |
| **Ex. No:6** | **DATA FRAME FUNCTIONS ON INBUILT DATA SET** |
| **Date:13.02.24** |

# AIM:

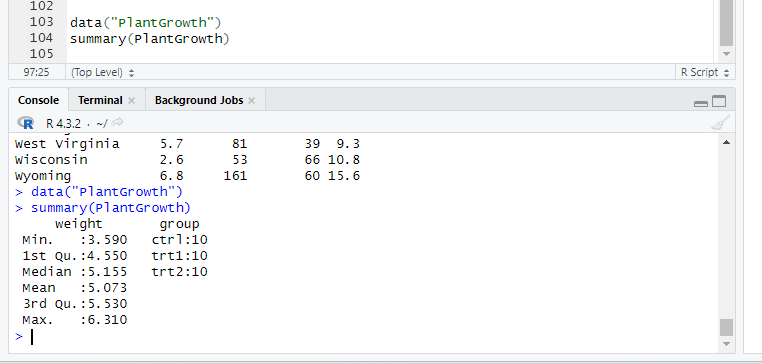
To know the functions of data frame with inbuilt data set.

PROCEDURE:

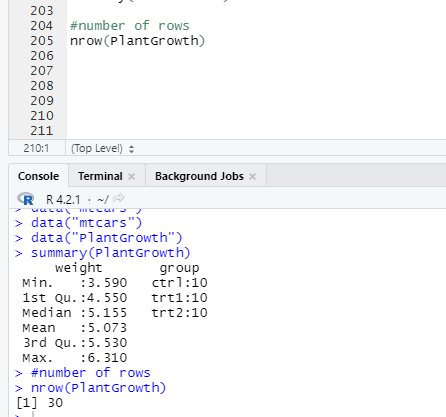
1. **data () -To load the inbuild data set**



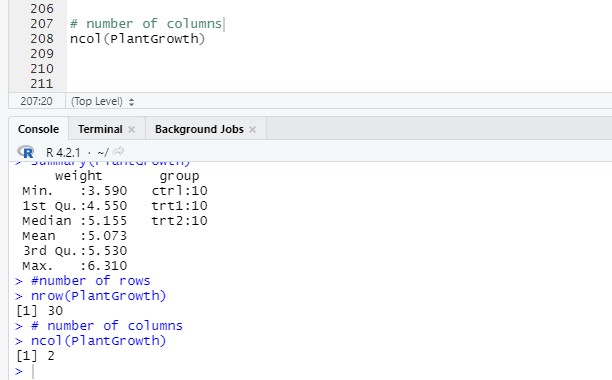
1. **summary () – to summaries the data set**



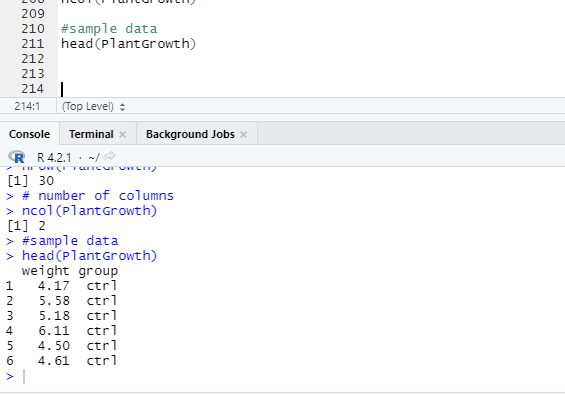
1. **nrow ()- to know the number of rows**



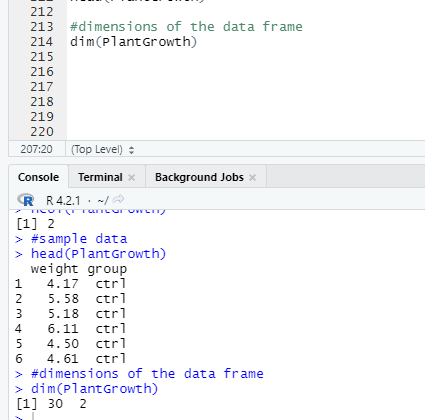
1. **ncol()- to know the number of columns in dataset**



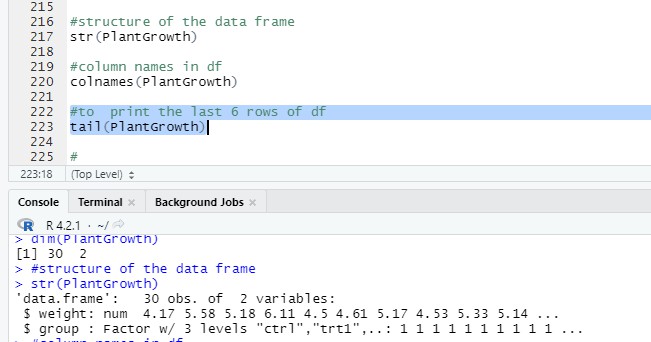
**6.head()-to print the sample data**



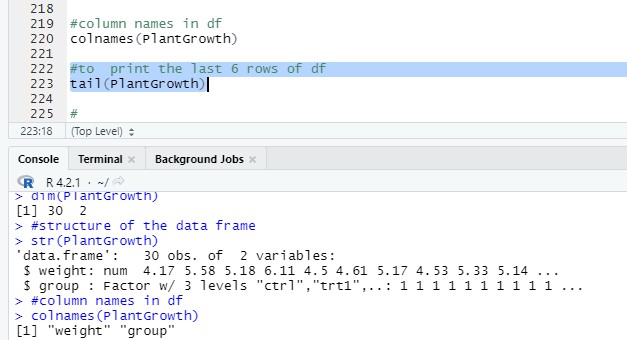
**7dim() - shows the dimensions of the data frame by row and column**



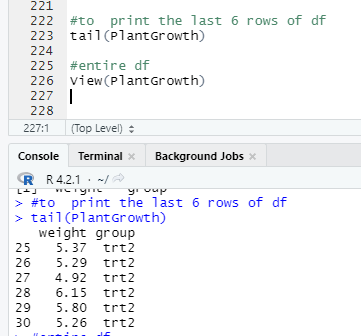
1. **str() - shows the structure of the data frame**



1. **colnnames() - shows the name of each column in the data frame**



1. **tail() - shows the last 6 rows of the data frame**



# RESULT:

Thus, the functions of data frame using are run using inbuilt dataset.

|  |  |
| --- | --- |
| **Ex. No:7** | **DPLYR FUNCTIONS ON DATA SET** |
| **Date:20.02.24** |

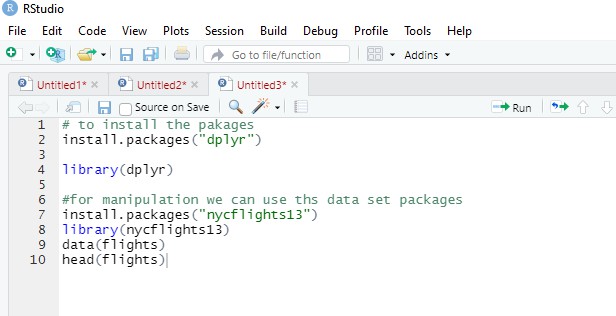
# AIM:

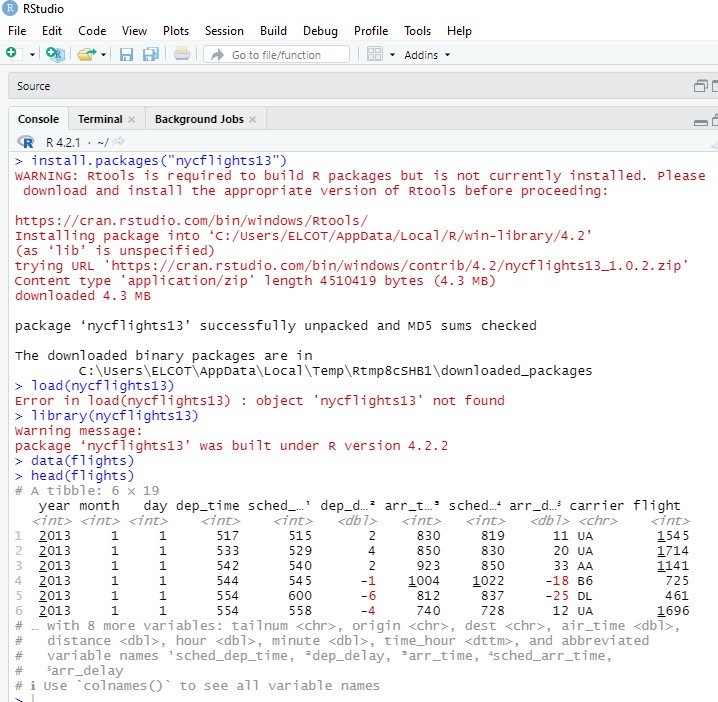
To know about Dplyr functions on data set.

# PROCEDURE:

**The dplyr package in** [**R Programming Language**](https://www.geeksforgeeks.org/introduction-to-r-programming-language/)is a structure of data manipulation that provides a uniform set of verbs, helping to resolve the most frequent data manipulation hurdles.

At First we should install the packages and load data for manipulation





[is.na()](https://rdrr.io/r/base/NA.html)

&,

[==,](https://rdrr.io/r/base/Comparison.html)









>,

[|](https://rdrr.io/r/base/Logic.html), [!](https://rdrr.io/r/base/Logic.html),

,

[between()](https://dplyr.tidyverse.org/reference/between.html)

[xor()](https://rdrr.io/r/base/Logic.html)

[near()](https://dplyr.tidyverse.org/reference/near.html)

dplyr package provides various important functions that can be used for Data Manipulation. These are:

* 1. filter(): to select records based on their values.
  2. arrange(): to reorder.
  3. select(): to select variables from the dataset
  4. mutate(): to add new variables
  5. summarize(): to condense multiple values into one.
  6. group\_by(): to break down the dataset into specified groups of rows

1. **filter(): to select records based on their values.**

function is used to subset a data frame, retaining all rows that satisfy

filter()

The

your conditions. To be retained, the row must produce a value of TRUE for all conditions. Note that when a condition evaluates to NA the row will be dropped, unlike base subsetting with [.

There are many functions and operators that are useful when constructing the expressions used to filter the data:

>= etc

1. **arrange(): to reorder.**

largely ignores grouping; you need to explicitly

arrange()

Unlike other dplyr verbs,

mention grouping variables (or use .by\_group = TRUE) in order to group by them, and functions of variables are evaluated once per data frame, not once per group.

1. **select(): to select variables from the dataset**

Tidyverse selections implement a dialect of R where operators make it easy to select variables:

* : for selecting a range of consecutive variables.
* ! for taking the complement of a set of variables.
* & and | for selecting the intersection or the union of two sets of variables.
* [c()](https://rdrr.io/r/base/c.html) for combining selections.

In addition, you can use **selection helpers**. Some helpers select specific columns:

|  |  |  |  |
| --- | --- | --- | --- |
|  | [everythin](https://tidyselect.r-lib.org/reference/everything.html) | [g()](https://tidyselect.r-lib.org/reference/everything.html) | : Matches all variables. |
| [last\_col()](https://tidyselect.r-lib.org/reference/everything.html) | : Select last variable, possibly with an offset. | |

These helpers select variables by matching patterns in their names:

[starts\_with()](https://tidyselect.r-lib.org/reference/starts_with.html): Starts with a prefix. [ends\_with()](https://tidyselect.r-lib.org/reference/starts_with.html): Ends with a suffix.

: Contains a literal string.

: Matches a regular expression.

[contains()](https://tidyselect.r-lib.org/reference/starts_with.html)











[num\_range()](https://tidyselect.r-lib.org/reference/starts_with.html): Matches a numerical range like x01, x02, x03.

[matches()](https://tidyselect.r-lib.org/reference/starts_with.html)

These helpers select variables from a character vector:

* + [all\_of()](https://tidyselect.r-lib.org/reference/all_of.html): Matches variable names in a character vector. All names must be present,

otherwise an out-of-bounds error is thrown.

* + [any\_of()](https://tidyselect.r-lib.org/reference/all_of.html): Same as [all\_of()](https://tidyselect.r-lib.org/reference/all_of.html), except that no error is thrown for names that don't exist.

This helper selects variables with a function:

* [where()](https://tidyselect.r-lib.org/reference/where.html): Applies a function to all variables and selects those for which the function returns TRUE.

1. **mutate(): to add new variables.**

mutate() adds new variables and preserves existing ones; transmute() adds new variables and drops existing ones. New variables overwrite existing variables of the same name. Variables can be removed by setting their value to NULL.

Useful mutate functions













[+,](https://rdrr.io/r/base/Arithmetic.html) [-,](https://rdrr.io/r/base/Arithmetic.html) [log()](https://rdrr.io/r/base/Log.html), etc., for their usual mathematical meanings [lead()](https://dplyr.tidyverse.org/reference/lead-lag.html),

[dense\_rank()](https://dplyr.tidyverse.org/reference/ranking.html), [cumsum()](https://rdrr.io/r/base/cumsum.html),

[na\_if()](https://dplyr.tidyverse.org/reference/na_if.html),

[percent\_rank()](https://dplyr.tidyverse.org/reference/ranking.html), [row\_number()](https://dplyr.tidyverse.org/reference/ranking.html), [cume\_dist()](https://dplyr.tidyverse.org/reference/ranking.html), [ntile()](https://dplyr.tidyverse.org/reference/ranking.html) [cummax()](https://rdrr.io/r/base/cumsum.html), [cumany()](https://dplyr.tidyverse.org/reference/cumall.html),

[if\_else()](https://dplyr.tidyverse.org/reference/if_else.html), [recode()](https://dplyr.tidyverse.org/reference/recode.html), [case\_when()](https://dplyr.tidyverse.org/reference/case_when.html)

[coalesce()](https://dplyr.tidyverse.org/reference/coalesce.html)

[cumall()](https://dplyr.tidyverse.org/reference/cumall.html)

[cummin()](https://rdrr.io/r/base/cumsum.html),

[cummean()](https://dplyr.tidyverse.org/reference/cumall.html),

[min\_rank()](https://dplyr.tidyverse.org/reference/ranking.html),

[lag()](https://dplyr.tidyverse.org/reference/lead-lag.html)

**5.summarize(): to condense multiple values into one.**

summarise() creates a new data frame. It will have one (or more) rows for each combination of grouping variables; if there are no grouping variables, the output will have a single row summarising all observations in the input. It will contain one column for each grouping variable and one column for each of the summary statistics that you have specified.

summarise() and summarize() are synonyms.

Useful functions

* + Center: [mean()](https://rdrr.io/r/base/mean.html), [median()](https://rdrr.io/r/stats/median.html)

[mad()](https://rdrr.io/r/stats/mad.html)

* + Spread: [sd(),](https://rdrr.io/r/stats/sd.html) [IQR()](https://rdrr.io/r/stats/IQR.html),
  + Range: [min(),](https://rdrr.io/r/base/Extremes.html) [max()](https://rdrr.io/r/base/Extremes.html), [quantile()](https://rdrr.io/r/stats/quantile.html)

[n\_distinct()](https://dplyr.tidyverse.org/reference/n_distinct.html)

[nth()](https://dplyr.tidyverse.org/reference/nth.html)

[last()](https://dplyr.tidyverse.org/reference/nth.html),

[first()](https://dplyr.tidyverse.org/reference/nth.html),

* + Position: ,
  + Count: [n()](https://dplyr.tidyverse.org/reference/context.html),
  + Logical: [any()](https://rdrr.io/r/base/any.html),

[all()](https://rdrr.io/r/base/all.html)

**6.group\_by(): to break down the dataset into specified groups of rows**

Most data operations are done on groups defined by variables. group\_by() takes an existing table and converts it into a grouped table where operations are performed "by group". ungroup() removes grouping.

# RESULT:

Thus, the dplyr functions on a dataset are known now.

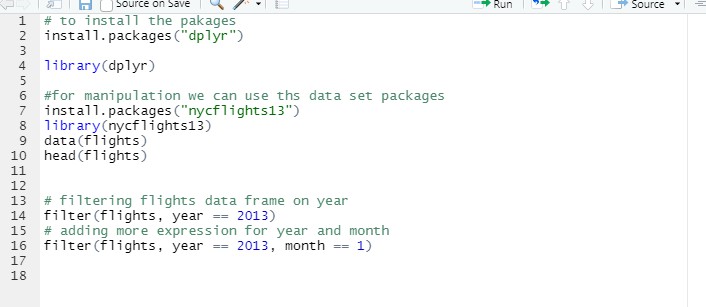
|  |  |
| --- | --- |
| **Ex.No:8** | **IMPLEMENTATION THE FOLLOWING IN R** |
| **Date:27.02.24** |

# AIM:

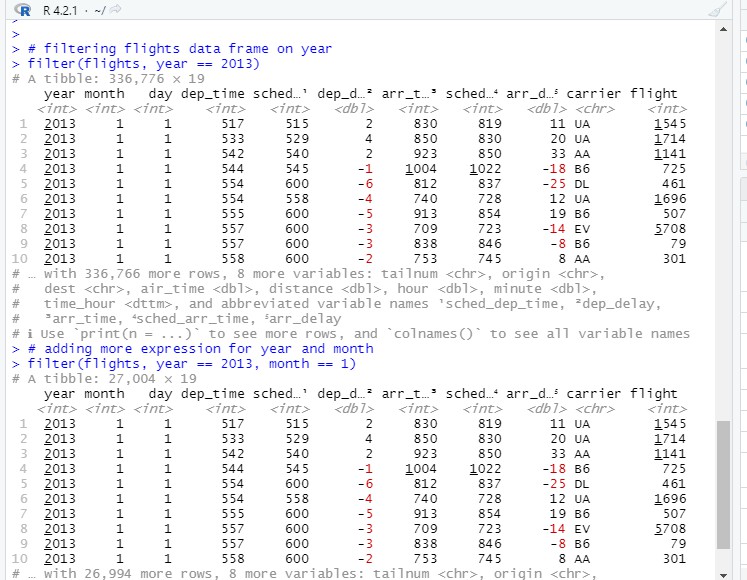
To write program for the implementation of dplyr functions on a dataset in r.

# PROCEDURE:

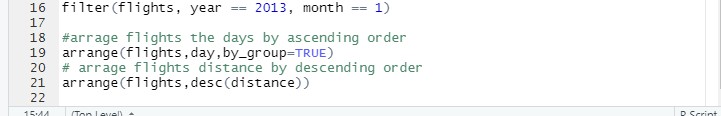
1. **filter(): Input:**



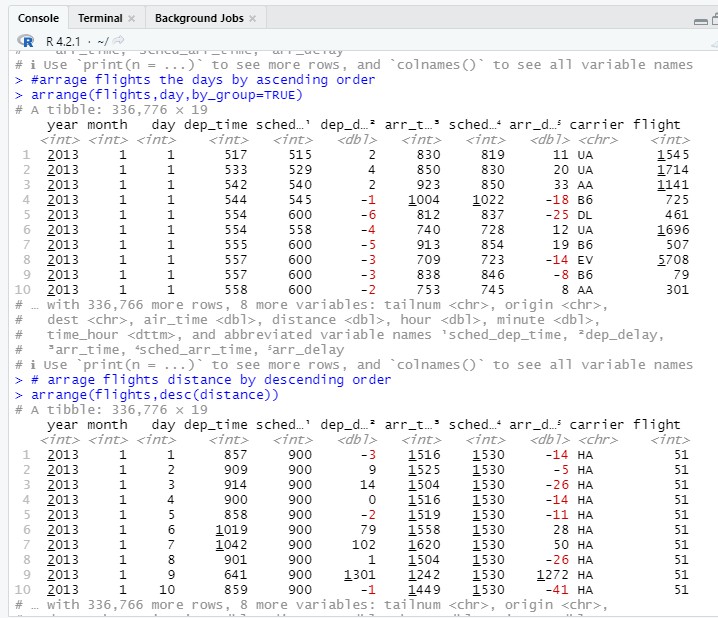
**Output:**



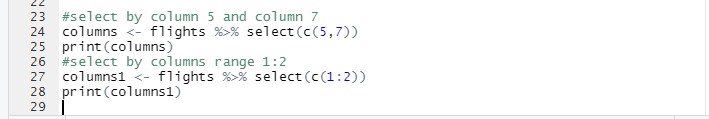
1. **arrange(): Input:**



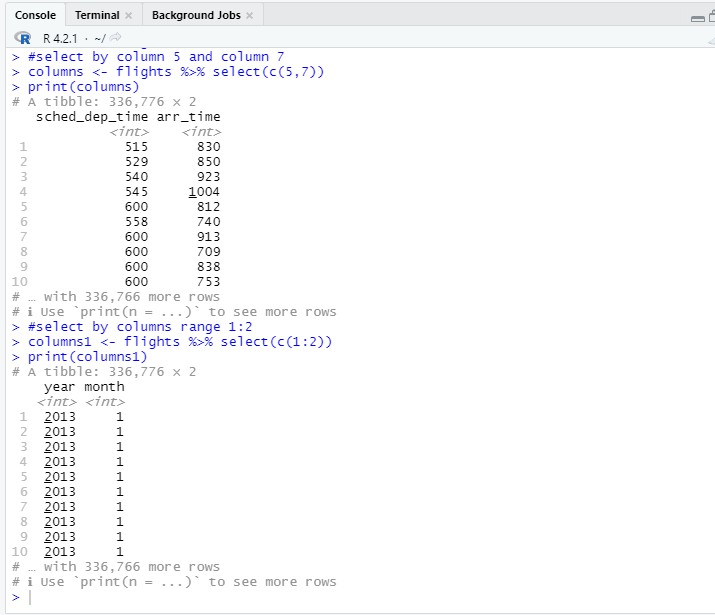
**Output:**



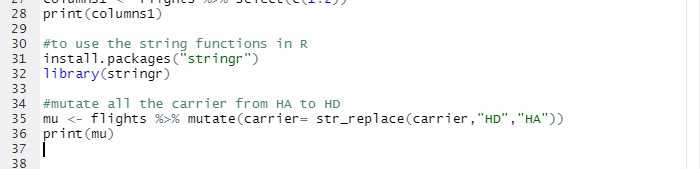
1. **select(): Input:**



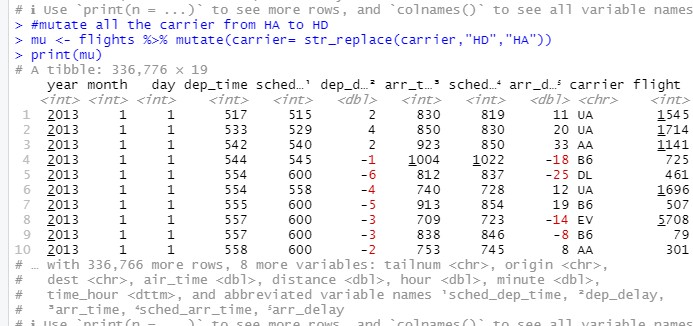
**Output:**



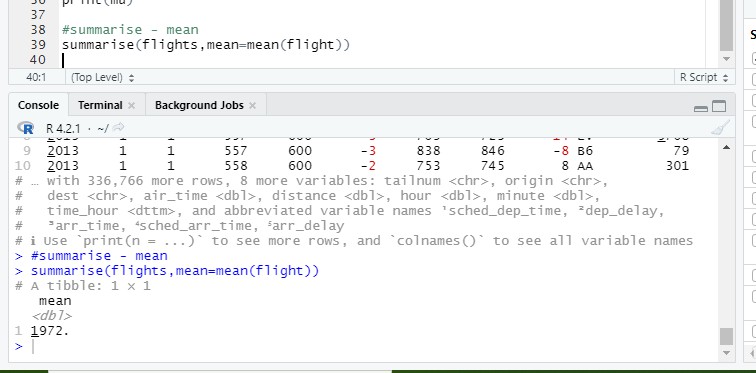
1. **mutate(): Input:**



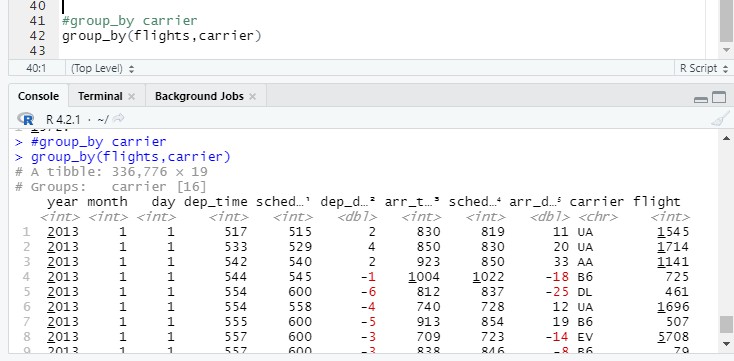
**Output:**



1. **summarise():**



1. **group\_by():**



# RESULT:

Thus ,the dplyr functions on a flights data set was implement successfully.

|  |  |
| --- | --- |
| **Ex.No:9** | **BASIC PLOTS IN R** |
| **Date:05.03.24** |

# AIM:

To Write Program to Basic Plots In R.

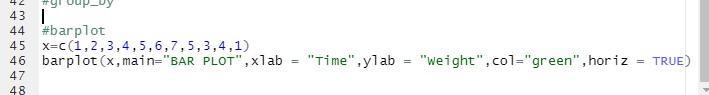
# PROCEDURE:

**R language** is mostly used for statistics and data analytics purposes to represent the data graphically in the software. To represent those data graphically, charts and graphs are used in R. There are hundreds of charts and graphs present in R. For example, bar plot, box plot, mosaic plot, dot chart, coplot, histogram, pie chart, scatter graph, etc.

1. **Barplot:**

Bar plot or Bar Chart in R is used to represent the values in data vector as height of the bars. The data vector passed to the function is represented over y-axis of the graph. Bar chart can behave like histogram by using **table()** function instead of data vector.

**Input:**



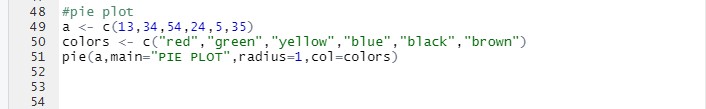
**Output:**



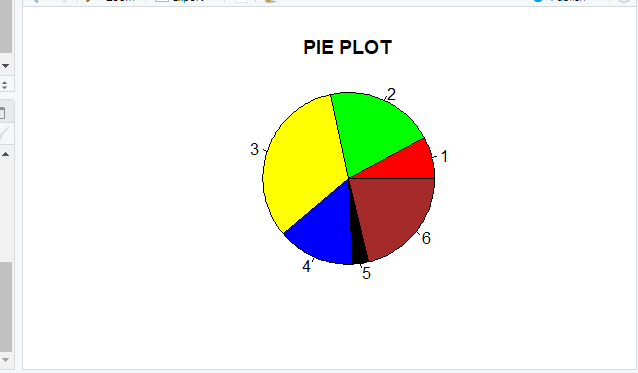
1. **pie chart:**

Pie chart is a circular chart divided into different segments according to the ratio of data provided. The total value of the pie is 100 and the segments tell the fraction of the whole pie. It is another method to represent statistical data in graphical form and **pie()** function is used to perform the same.

**Input:**



**Output:**



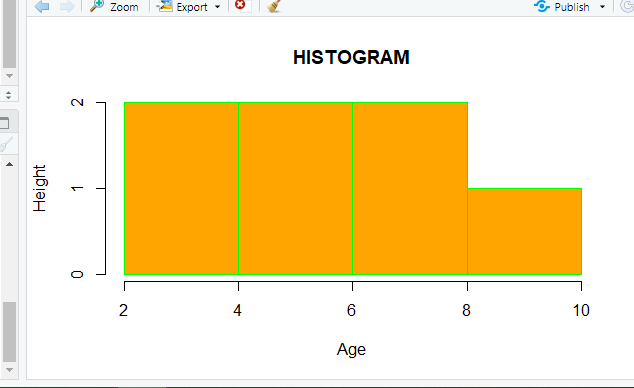
1. **Histogram:**

Histogram is a graphical representation used to create a graph with bars representing the frequency of grouped data in vector. Histogram is same as bar chart but only difference between them is histogram represents frequency of grouped data rather than data itself.

**Input:**



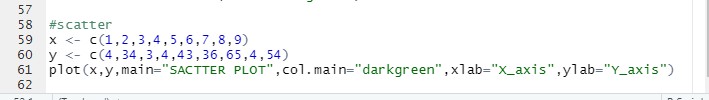
**Output:**



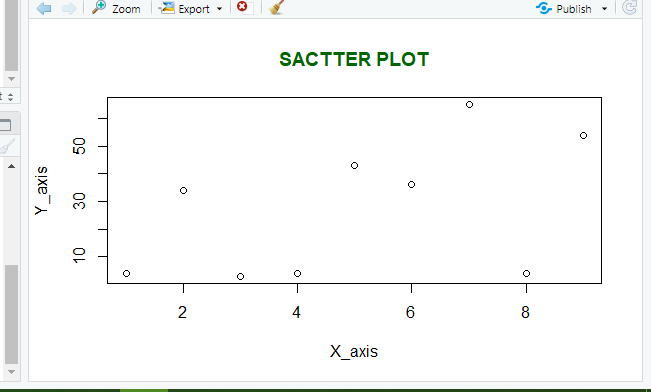
1. **Scatter plot:**

A Scatter plot is another type of graphical representation used to plot the points to show relationship between two data vectors. One of the data vectors is represented on x-axis and another on y-axis.

**Input:**



**Output:**

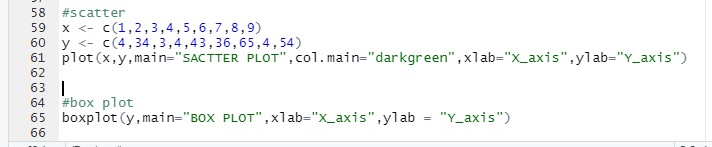


1. **Box plot:**

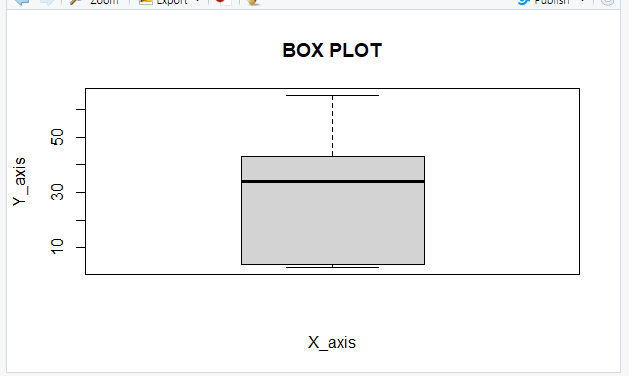
Box plot shows how the data is distributed in the data vector. It represents five

values in the graph i.e., minimum, first quartile, second quartile(median), third quartile, the maximum value of the data vector.

**Input:**



**Output:**



## RESULT:

Thus some of the basic plots of R , are was executed successfully.

|  |  |
| --- | --- |
| **Ex.No:10** | **GGPLOTS WITH R** |
| **Date:12.03.24** |

# AIM:

To write program for ggplots with R.

# PROCEDURE:

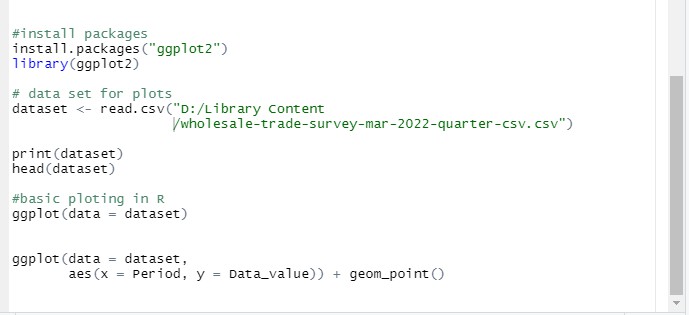
**Ggplot packages new version is ggplot2 package in R Programming**

**Language** also termed as **Grammar of Graphics** is a free, open-source, and easy-to-use visualization package widely used in [R](https://www.geeksforgeeks.org/introduction-to-r-programming-language/). It is the most powerful visualization package written by Hadley Wickham.

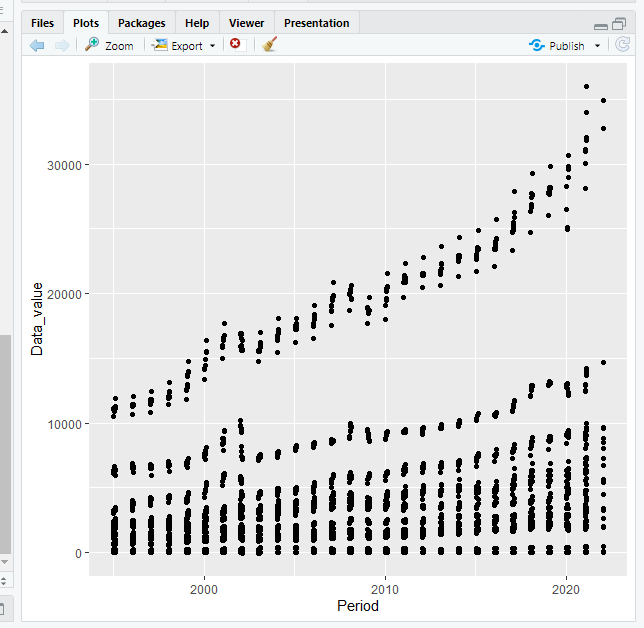
* **Data:** The element is the data set itself
* **Aesthetics:** The data is to map onto the Aesthetics attributes such as x-axis, y-axis, color, fill, size, labels, alpha, shape, line width, line type
* **Geometrics:** How our data being displayed using point, line, histogram, bar, boxplot
* **Facets:** It displays the subset of the data using Columns and rows
* **Statistics:** Binning, smoothing, descriptive, intermediate
* **Coordinates:** the space between data and display using Cartesian, fixed, polar, limits
* **Themes:** Non-data link

## PROGGRAMMING:

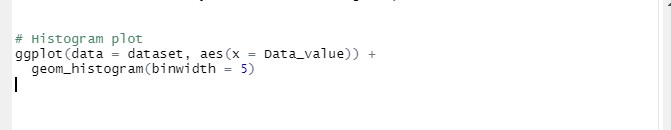
1. **Input:**



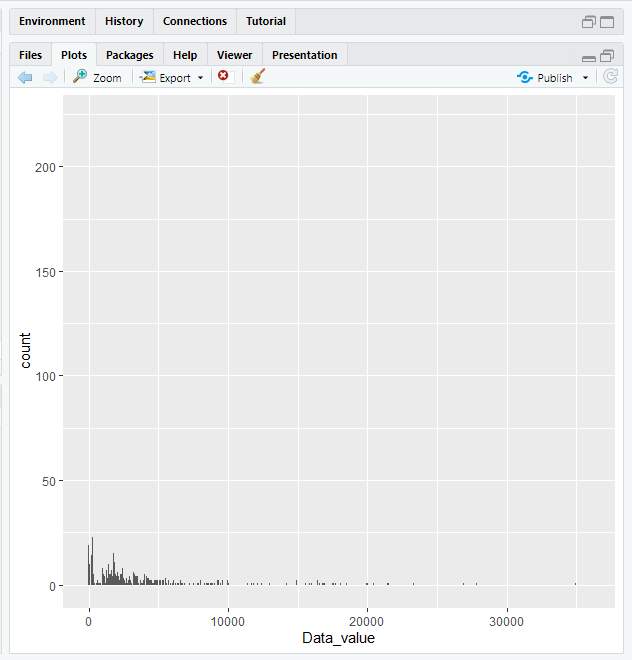
**Output:**



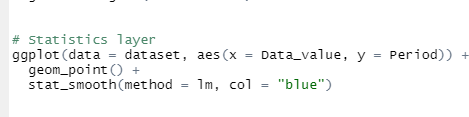
1. **Input:**



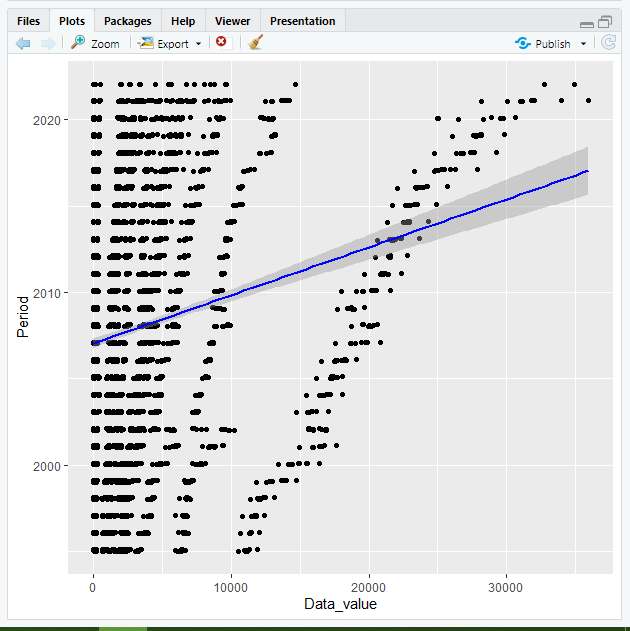
**Output:**



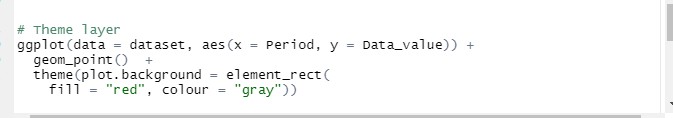
1. **Input:**



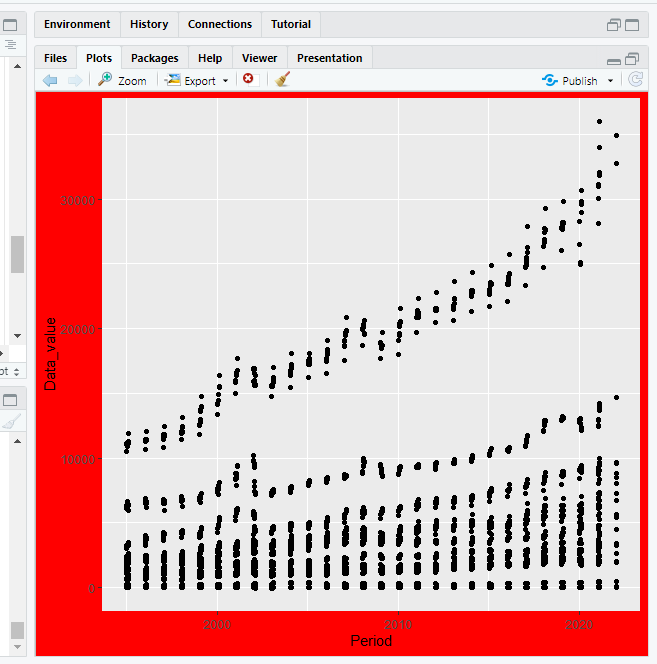
**Output:**



1. **Input:**



**Output:**



## RESULT:

Thus, some the ggplots are run successfully and the result are verified.

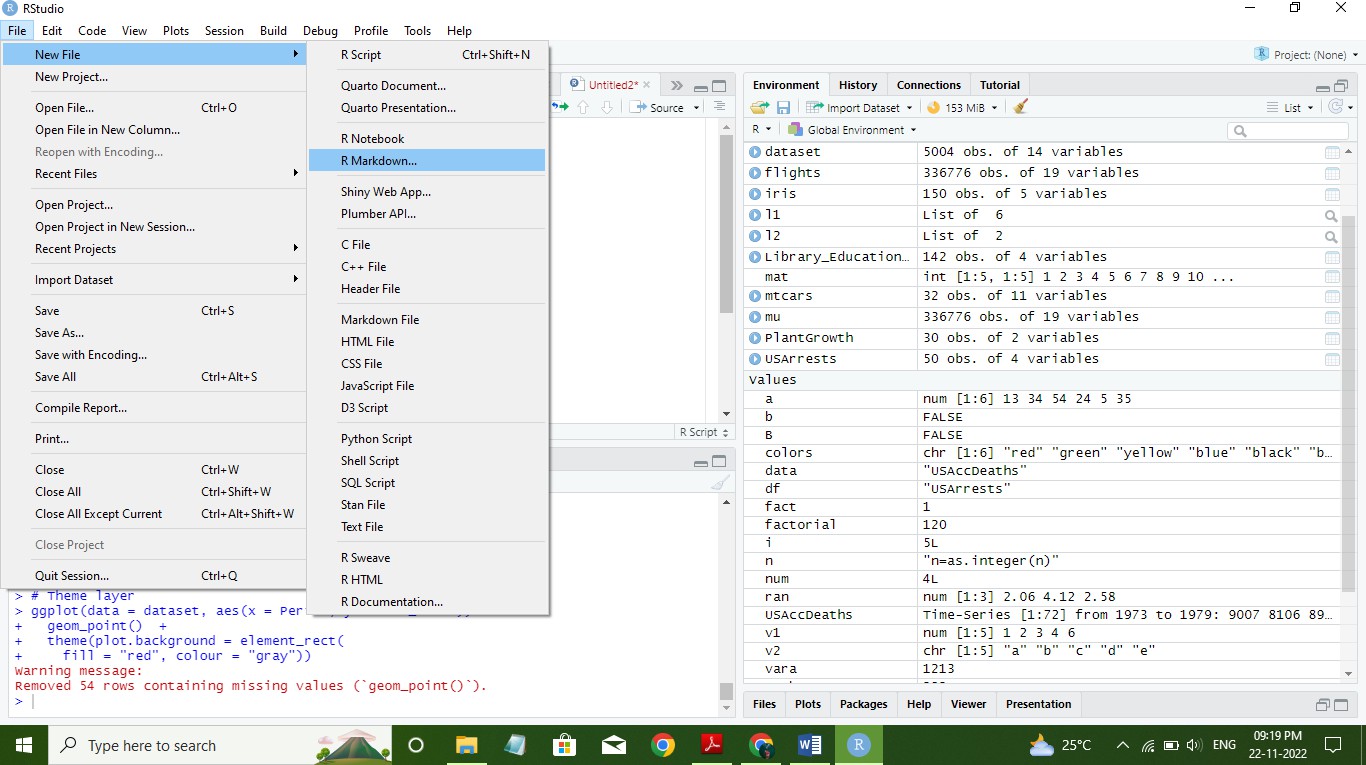
|  |  |
| --- | --- |
| **Ex. No:11** | **WORKING WITH R MARKDOWN & PUSHING CODE TO GIT** |
| **Date:19.3.24** |

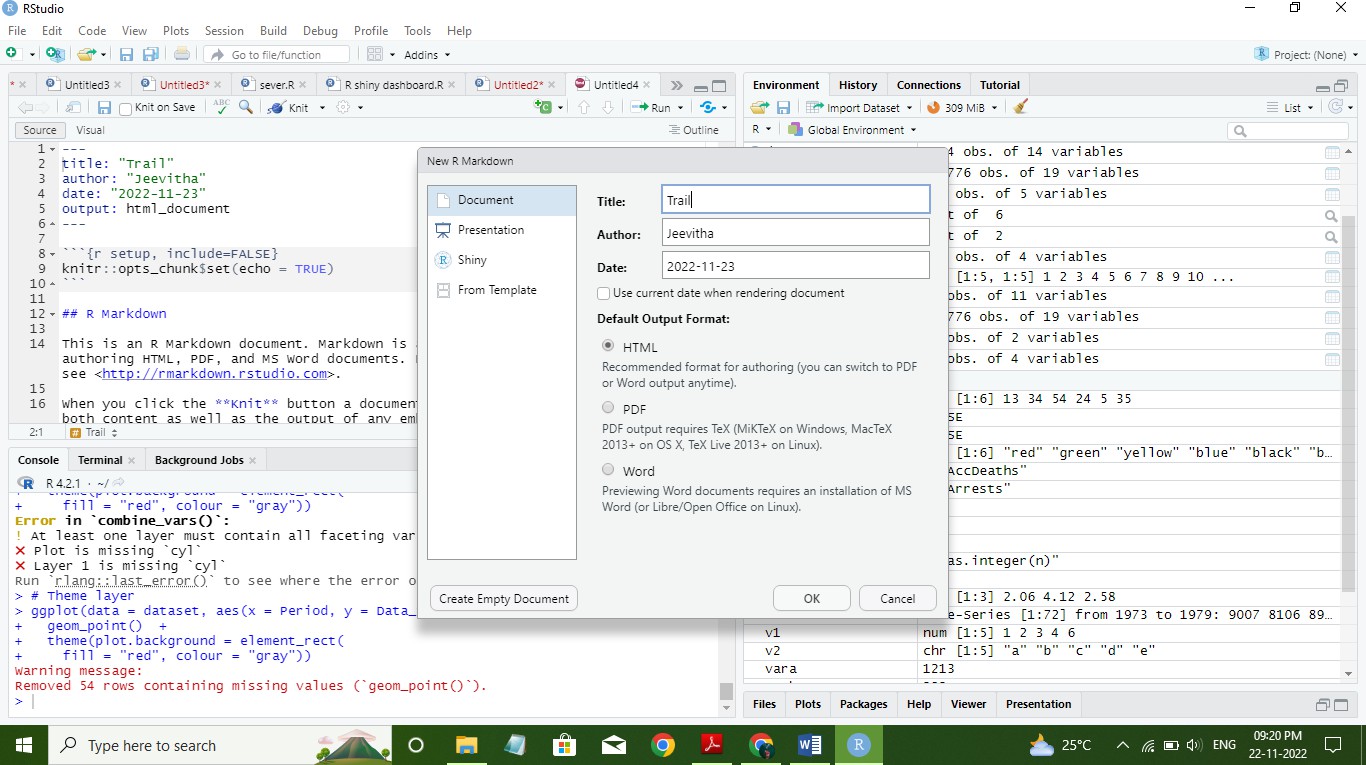
# AIM:

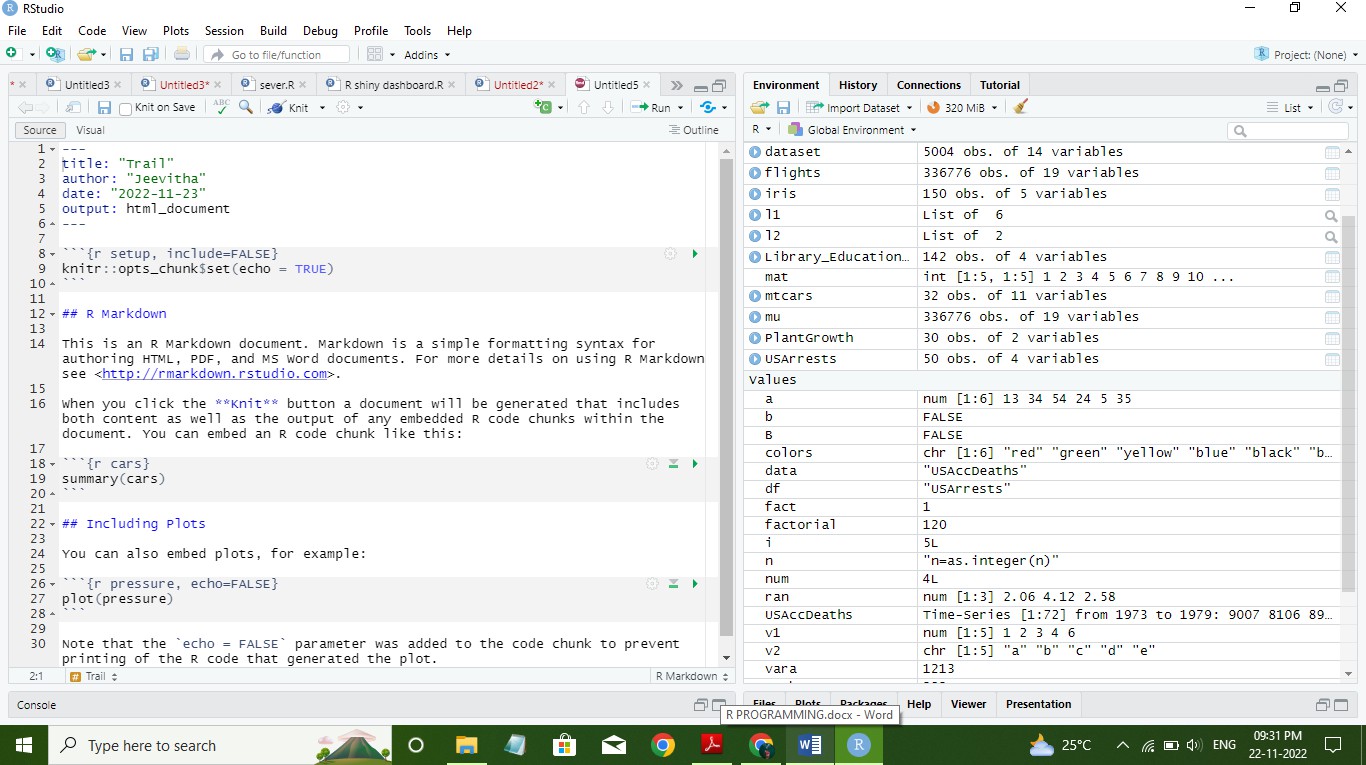
To write program Working with R Markdown & pushing Code to Git.

# PROCEDURE:

[**R Markdown**](https://rmarkdown.rstudio.com/)is a great package for [RStudio](https://www.rstudio.com/). You can create (or knit) an html file using R Markdown which will let you add text, snippets of code, and plots. Additionally, you can run R code in R Markdown and have the output as part of the html document.







## RESULT:

Thus, the program was run for creating the R markdown.

|  |  |
| --- | --- |
| **Ex. No:12** | **BUILDING R SHINY DASHBOARD APP** |
| **Date:26.03.24** |

# AIM:

To write program for create basic app using shiny dashboard application.

# PROCEDURE:

Shiny is an open source R package from the team at [RStudio, PBC](https://shiny.rstudio.com/). RStudio built Shiny to provide an elegant and easy-to-use web framework for developing web applications in R. Shiny enables R users to create incredible apps, interactive maps, and dashboards. And you don’t need advanced web dev skills to build it!

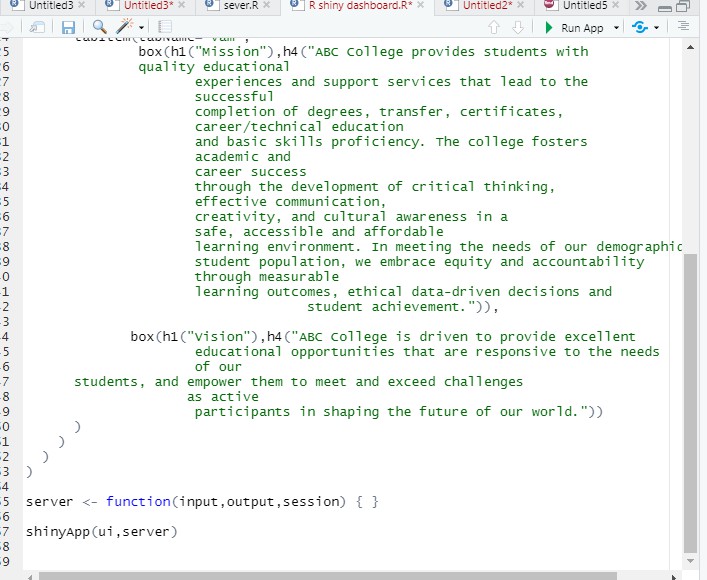
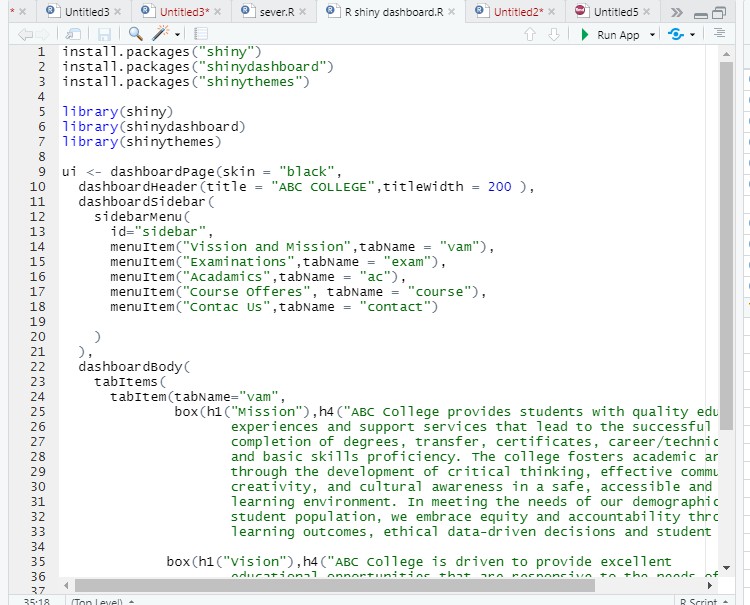
Shiny dashboards let you access a complete web application framework within the R environment. Easily turn your work in R, analyses and visualizations, machine learning models, and more into web applications that deliver value to businesses. As a complete application, end- users don’t need an understanding of R to use it. Deliver a complete, easy-to-use, and interactive product that improves the way you do business.

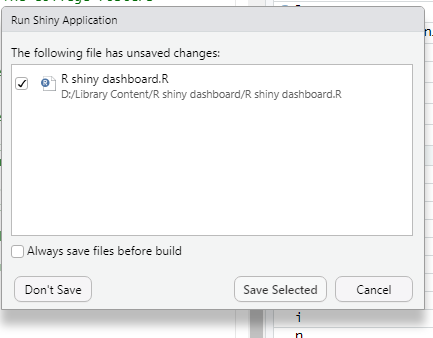
Shiny’s web framework enables easy customization of the dashboard using custom HTML, CSS, SCSS, Javascript, and so on. This level of customization lets you create a unique, branded dashboard that’s not possible with other BI software suites. Add colors, logos, fonts, and more that better represent your business.

Shiny is open source and cost-friendly compared to its counterparts like Power BI and Tableau. You can explore a comprehensive look comparing [Power BI to Shiny](https://appsilon.com/powerbi-vs-r-shiny/) and [Tableau to](https://appsilon.com/tableau-vs-r-shiny/) [Shiny](https://appsilon.com/tableau-vs-r-shiny/) on the Appsilon blog.

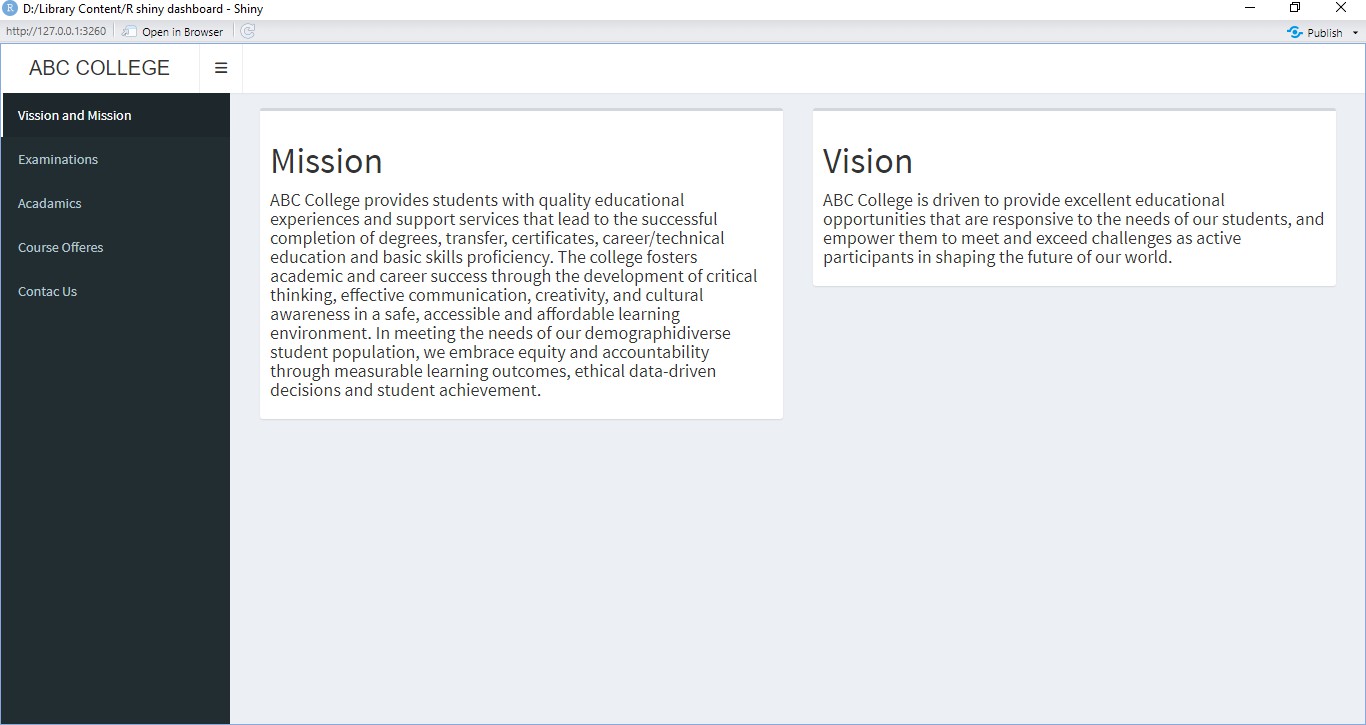
## PROGRAMMING:

**Input:**





**Output:**



## RESULT:

Thus, using the r shiny dashboard created simple basic app.