|  |  |  |
| --- | --- | --- |
| logo1 | **NitteMeenakshi Institute of Technology**  (AN AUTONOMOUS INSTITUTION AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM)  (A Unit of Nitte Education Trust, Mangalore)  PB No. 6429, Yelahanka, Bangalore 560-064, Karnataka  Telephone: 080- 22167800, 22167860  Fax: 080 - 22167805 |  |

**Big Data Analytics Using R**

**Manual**

****

**Department of Computer Science & Engineering**



**Part A**

1. **a**. Create two variables, x and y of type integer using the as.integer function and assign them the values of 4 and 5. Add the x by y and store the result in a third variable named z. What kind of variable is z?

**b.** Create two variables, x and y of type integer using the as.integer function and assign them the values of 3 and 2. Divide the x by y and store the result in a third variable named z. What kind of variable is z? How this is different than the answer in the previous question?

1. **a.** Create a sequence of numbers ranging from 1 − 10 by 0.1 and assign it to the variable x.

**b.**Create a sequence of numbers from 100 down to 50 by 2 and assign it to the variable y.

1. Create vector for the following

**a.** (4*,* 6*,* 3*,* 4*,* 6*,* 3*, . . . ,* 4*,* 6*,* 3) where there are 10 occurrences of 4.

**b.** Use the function paste to create the following character vectors of length 30

("fn1", "fn2", ..., "fn30").In this case, there is no space between fn and the number

1. Calculate the following using appropriate function

**a.**



**b.**



1. **a.** Turn the vector of character items "Control", "Control", "Control", "Ear Removal", "Ear Removal", "Ear Removal", "Ear Removal", "Fake Ear Removal", "Fake Ear Removal", "Fake Ear Removal", "Fake Ear Removal" into a Factor variable and create a table from it to show the number of entries in each treatment.

**b.** Create a vector of character variables that contains 25 ”a”, 15 ”b”, and 58 ”c” instances. What is the length of this vector? Create a table from the entries.

1. Suppose

A =

1. Check that A3 = 0 where 0 is a 3 X 3 matrix with every entry equal to 0.
2. Replace the third column of A by the sum of the second and third columns.
3. Calculate the 3 x 3 matrix ATA.
4. create the following patterned matrix:



1. a. Consider the continuous function



Write a function tmpFn which takes a single argument xVec. The function should return the vector of values of the function *f*(*x*) evaluated at the values in xVec. Hence plot the function *f*(*x*) for *-*3 *< x <* 3.

**b.** Write a function which takes a single argument which is a matrix. The function should return a matrix which is the same as the function argument but every odd number is doubled.

Hence the result of using the function on the matrix

should be

1. **a.** Create three different variables, one that is numeric type and other two are vector of characters. Use these to create data frame of student.(USN,Name,Marks)

**b.** Add a new numeric data column to the existing data frame (Age). Provide summary of the data

**c.**  Display the list of student whose Age is less than 20 and Marks greater than 25

**Part B**

1. Suppose we are given xVec which represents the vector (*x*1,……..,*xn*) and yVec which represents the vector (*y*1,…….,*ym*). Suppose further that zVec represents the vector (*z*1,….,*zn*) where *z*1 = number(*yj < x*1) *z*2 = number(*yj < x*2) *….. zn* = number(*yj < xn*)

Formally, if *I* denotes the indicator function, then



**a.** By using the function outer, write a function which takes the arguments xVec and yVec and returns the vector zVec.

**b.** Repeat part (a) but use sapply instead of outer.,

**c.** Now repeat part (a) but use vapply instead of outer or sapply.

**d.** Investigate the relative speed of your solutions by using system.time.

1. **a.**  Zeller’s congruence is the formula:



where [*x*] denotes the integer part of *x*; for example [7.5] = 7.

Zeller’s congruence returns the day of the week *f* given:

*k* = the day of the month,

*y* = the year in the century

*c* = the first 2 digits of the year (the century number)

*m* = the month number (where January is month 11 of the preceding year, February is month 12 of the preceding year, March is month 1, etc.)

For example, the date 21*/*07/1963 has *m* = 5, *k* = 21, *c* = 19, *y* = 63; whilst the date 21*/*2/1963 has *m* = 12, *k* = 21, *c* = 19 and *y* = 62.

Write a function weekday(day,month,year) which returns the day of the week when given the numerical inputs of the day, month and year.

Note that the value of 1 for *f* denotes Sunday, 2 denotes Monday, etc.

**b.** Does your function work if the input parameters day, month and year are vectors with the same length and with valid entries?

11. *The waiting time of the n*th *customer in a single server queue.* Suppose customers labeled

*C*0, *C*1, *. . .* ,*Cn* arrive at times t= 0,t1,….tn for service by a single server. The inter arrival times *A*1 = t1 -t 0, *. . .* , *An* = t *n-t n−*1are independent and identically distributed random variables with the exponential density



The service times *S*0, *S*1,……, *Sn* are independent and identically distributed random variables which are also independent of the inter arrival times with the exponential density



Let *Wj* denote the waiting time of customer *Cj* . Hence customer *Cj* leaves at time t *j* + *Wj* + *Sj* If this time is greater than t*j*+1 then the next customer, *Cj*+1 must wait for the time t*j* + *Wj* + *Sj j*+1. Hence we have the recurrent relation *W*0= 0

*Wj*+1= max{0*, Wj* + *Sj  - Aj*+1*}* for *j* = 0, 1, *. . .* , *n-*1

**a.** Write a function queue (n, aRate, sRate) which simulates one outcome of *Wn* where aRate denotes λ*a* and sRate denotes λ*s*. Try out your function on an example such as queue(50,2,2)

**b.** Now suppose we wish to simulate many outcomes of *Wn* in order to estimate some feature of the distribution of *Wn*. Write a function which uses a loop to repeatedly call the function in part (a) to calculate *Wn*. Then write another function which uses *sapply* (or replicate) to call the function created in part (a). Compare the speed of the two functions by using system.time.

**c.** Can we do any better? Try writing a vectorised form of the basic recurrence relation *Wj*+1 = max{0*, Wj* + *Sj* - *Aj*+1} where *Wj* is treated as a vector. Compare the speed of this new function with the two answers to the previous part.

12. **a.** The code ts(datVec, start=c(1960,3), frequency=12) creates a time series with monthly

observations (frequency=12), with first observation in March 1960 (start=c(1960,3)) and with values specified in the vector datVec.

Suppose *z*1, *z*2, ….,*zn* is a time series. Then we define the exponentially weighted moving average of this time series as follows: select a starting value *m*0 and select a discount factor *\_*. Then calculate *m*1, *m*2, *…* , *mn* recursively as follows: for *t* = 1, 2, *…* , *n*

*et* = *zt - mt−*1

*mt* = *mt−*1 + (1- δ)*et*

Write a function tsEwma(tsDat, m0=0, delta=0.7) where tsDat is a time series, m0 is the starting value *m*0 and delta is δ. The function should return *m*1, *m*2, …, *mn* in the form of a time series.

**b.** Write a function, called myListFn, which takes a single argument *n* and implements the following algorithm:

* 1. Simulate *n* independent numbers, denoted x = (*x*1, *x*2, *…* , *xn*), from the *N*(0*;* 1) distribution.
  2. Calculate the mean 
  3. If ≥0, then simulate *n* independent numbers, denoted y = (*y*1, *y*2, …, *yn*), from the exponential density with mean x.
  4. If  *<* 0, then simulate *n* independent numbers, denoted z = (*z*1, *z*2, *…* , *zn*), from the exponential density with mean -. Set y = (*y*1, *y*2, *…* , *yn*) = -z.
  5. Calculate *k* which is the number of *j* with |*yj| > |xj*|.
  6. Return the list of x, y and *k* with names xVec, yVec and count respectively.

Execute the following lines and check the format of the answers:

lapply( rep(10,4), myListFn )

sapply( rep(10,4), myListFn )

**13**. Write a program to create the csv file for storing Employee data. Containing the data

(EmpID, EmpName , DOJ, EmpCode, Dept, Desig.)

1. Read the suitable number of employee details from the user.
2. Create a dataframe of Employee
3. Store the dataframe in the csv file
4. Check the difference between csv and csv2 file
5. Read the data from csv and Display the contents
6. Append a new row into the csv file

**14.** Dataset example

**a.** List the data set available in your system using suitable command

**b.** Select “mtcars” data set, find and display the number of rows and columns in that data set

**c.** Find are there more automatic (0) or manual (1) transmission-type cars in the dataset? Hint: 9th column indicate the transmission type

**d.** Get a scatter plot of ‘hp’ vs ‘weight’.

**e.** Change ‘am’, ‘cyl’ and ‘vs’ to integer and store the new dataset as ‘newmtc’.

**f.** Extract the cases where cylinder is less than 5

**15.** Time series

1. Read the data from the file sales.csv. (Need to download this from the attachment)
2. Transform the data into a time series object of the ts type (indicate that the data is monthly, and the starting period is January 1992).  
   Print the data.
3. Plot the time series. Ensure that the y axis starts from zero.
4. Use the gghistogram function from the forecast package to visually inspect the distribution of time series values. Add a kernel density estimate and a normal density function to the plot.
5. Use the decompose function to break the series into seasonal, trend, and irregular components (apply multiplicative decomposition).  
   Plot the decomposed series.

**16.** Consider “Airquality” dataset

1. Display the dimension of the dataset
2. Display the class of each fields in the data set
3. Test the missing values
4. Recode the missing values, as mean of the column values
5. Exclude the missing values

**Assessment Method**

* **Experiment Write up + Execution + Viva - 15 Marks**
* **Lab Record Writing - 10 Marks**
* **Lab Internals Test - 15 Marks**
* **Case Study - 10 Marks**

**-----------------------**

**Total = 50 Marks**

**Note:**

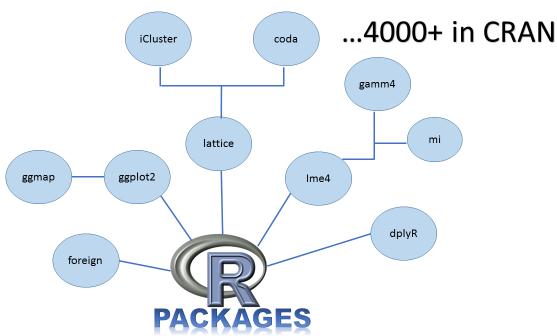
1. Case study component also need to be demonstrated for the SEE
2. One program from Part A and one program from Part B need to be executed.

**INTRODUCTION TO R**

* The R language is a terminology of S language which was designed in the 1980s by John Chambers at Bell labs and has been in widespread use in the statistical community since.
* It was grown up by Robert Gentleman and Ross Ihaka of the University of Auckland. R has been with us since 1993.
* After the four versions of ‘S’ language we came up with **R Programming tool** and **R Programming language**.
* It is concluded to adopt the syntax of the S language which has developed at Bell Laboratories

**The Development of R**

* 1991: Created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand
* 1993: First announcement of R to the public.
* 1995: R was made as free software.
* 1997: The R Core Group is formed (containing some people associated with S-PLUS). The core group controls the source code for R.
* 2000: R version 1.0.0 is released.
* 2013: R version 3.1.2 has been released on 2014-10-31.
* Latest version 3.4.1 released on 2017-6-30



**Why R essential?**

R provides you number of Datasets that can be used in Analytics and these datasets are built in and are available in packages

* Time Series Data
* Numeric Data
* Categorical Data
* Character Data
* Small Data
* Large Data

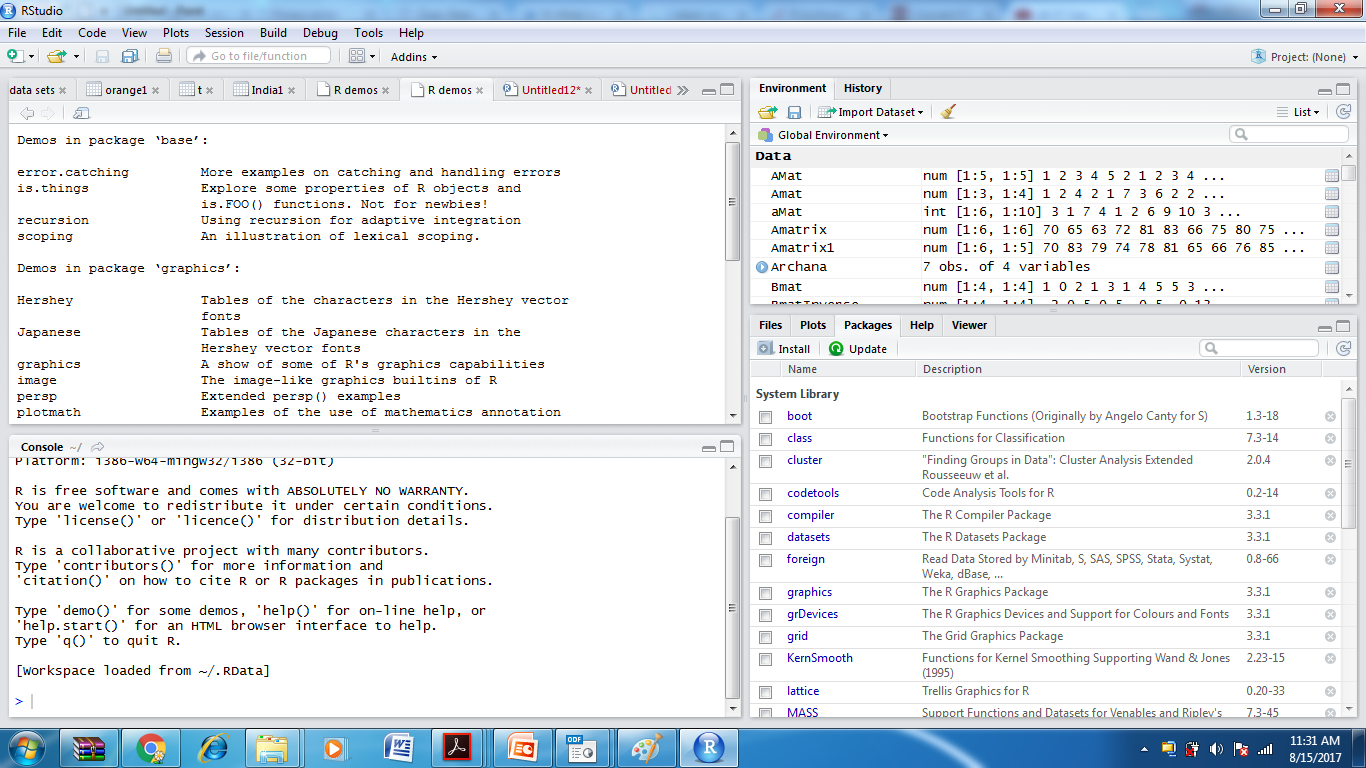
**What is R?**

* R is rather a programming language
* Limited user-friendly interfaces for data analysis
* Is object oriented and almost non declarative
* Similar to programming languages like Fortran, C, Java, Python
* The source code for the R software environment is written primarily in C, Fortran, and R

**Features of R**

* Provides Statistical and Graphical techniques
  + linear and nonlinear modelling
  + classical statistical tests
  + time-series analysis,
  + classification, clustering
  + Others
  + Available through additional packages
* easily extensible through functions and extensions
* C, C++, and Fortran code can be linked and called at run time
* Object-oriented, growing user base, scripting features
* Free and open-source

**R Studio –IDE**

****

**Working with R**

* Download & Install R: http://www.r-project.org/
* Download & Install R Studio:
* Materials
  + https://www.r-project.org/

**R Basics**

Basic Data types and how to create them

Operators and how to use them

Variable naming and how to create, manipulate and destroy

* R is functional language
* Each command given on the command prompt is either

Expression

1. + 5

sqrt(30)

Assignment

X <- 4 + 5

1. -> Y

Functions

User Defined

Built in ( Take the help)

Variables

Similar conventions used as in other programming languages can be declared as and when required

* 1. **Data Types**
     1. R calls data types /categorizes data types as ***classes***
     2. The variables are assigned with R-Objects and the data type of the R-object becomes the data type of the variable.
     3. Frequently used data types in R are
        1. Vectors
        2. Lists
        3. Matrices
        4. Arrays
        5. Factors
        6. Data Frames
           1. **Vectors**
  + There are six data types of these ***atomic*** vectors, also termed as six classes of vectors.
  + The other R-Objects are built upon the atomic vectors.

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Example** |  |
| Logical | TRUE, FALSE |  |
| Numeric | 12.3,5.0,0.999 | Numbers , including decimals |
| Integer | 12L | Integers |
| Complex | 5+9i | Complex() |
| Character | ‘a’, ’Hello’ | Data within the quotes |
| Raw | ‘Hello’ | Raw() – used to create  Raw numbers are represented as a two digit sequence of hex numbers. Valid hex digits include 0 − 9 as well as a, b, c, d, e, & f. |

* **Constants**

|  |  |
| --- | --- |
| **Name** | **Description** |
| pi | Mathematical functions ¶ |
| NULL | Absence of type |
| Nan | Not a Number |
| Infinity | ∞ as well as -∞ |
| NA | Used to represent missing data |

* + - * 1. **Lists**
  + can contain many different types of elements inside it like vectors, functions and even another list inside it.

>A= list(c(2,5,4), 21.4, 5+8i)

> print(A)

[[1]]

[1] 2 5 4

 [[2]]

[1] 21.4

[[3]]

[1] 5+8i

* (Vector all data are of the same data type, Lists store the data of different data types)
  + - * 1. **Matrices**
  + Matrices are 2-dimensional vectors
  + Created using the default constructor matrix() function
  + passing it a number for nrow and ncol
  + M = matrix( c('a','a','b','c','b','a'), nrow = 2, ncol = 3, byrow = TRUE)

> M [,1] [,2] [,3]

[1,] "a" "a" "b"

[2,] "c" "b" "a“

* + Default will be by column

**d. Arrays**

* 1. Any numbers of dimensions
  2. takes a *dim* attribute which creates the required number of dimension
  3. To create array of 2X3, with elements 1,2,3, dimension 3 times

> A<- array(c(1,2,3), dim=c(2,3,3))

> A

, **, 1**

**[,1] [,2] [,3]**

**[1,] 1 3 2**

**[2,] 2 1 3**

**, , 2**

**[,1] [,2] [,3]**

**[1,] 1 3 2**

**[2,] 2 1 3**

**, , 3**

**[,1] [,2] [,3]**

**[1,] 1 3 2**

**[2,] 2 1 3**

1. **Factors**
   1. Used to categorize the data and store as levels
   2. Useful in statistical modelling

> A<- c(2,3,5,34,3,2,56,34)

> Afact= factor(A)

> Afact

[1] 2 3 5 34 3 2 56 34

Levels: 2 3 5 34 56

1. **Data Frames**
   1. Same as Spreadsheet
   2. Data frames are used to store tabular data
   3. They are represented as a special type of list where every element of the list has to have the same length
   4. Each element of the list can be thought of as a column and the length of each element of the list is the number of rows
   5. Unlike matrices, data frames can store different classes of objects in each column (just like lists); matrices must have every element be the same class

* Example of creating Data frame

> id= c(1:4)

> Name= c("Asha","Geetha", "Rita", "Raj")

table1= data.frame(id,Name)

> table1 id Name

1 1 Asha

2 2 Geetha

3 3 Rita

4 4 Raj

Marks<<- c(25,28,30,22)

> (table1=data.frame(id,Name,Marks))

id Name Marks

1 1 Asha 25

2 2 Geetha 28

3 3 Rita 30

4 4 Raj 22

**II. Operators**

* Arithmetic Operator (Numeric Operator)
  + + Addition
  + - Subtraction
  + \* Multiplication
  + / Division
  + ^ Exponentioation
* Assignment Operator
  + ->
  + <-
  + <<-
  + =
* Relational and Logical Operator
  + >
  + <
  + ==
  + !=
  + &
  + |
  + !
* Special Operator
  + %% Mod operation
  + %/% Integer Division
  + %\*% Matrix Multiplication
  + : To build sequence
  + %in% Value Matching

**Functions/Commands**

**Flow Control**

1. If
2. If else
3. Continue
4. Break, Next
5. Switch
6. Repeat
7. For
8. while

**R data interfaces**

1. **CSV file**
   * 1. Reading

data <- read.csv("input.csv")

print(data)

2. Writing

data <- read.csv("input.csv")

retval <- subset(data, as.Date(start\_date) > as.Date("2014-01-01"))

write.csv(retval, "output.csv“)

1. **Excel File**

R can read directly from these files using some excel specific packages. Few such packages are - XLConnect, xlsx, gdata etc. We will be using xlsx package. R can also write into excel file using this package.

Reading

data <- read.xlsx("input.xlsx", sheetIndex = 1)

print(data)

**3. Binary Files**

Sometimes, the data generated by other programs are required to be processed by R as a binary file. Also R is required to create binary files which can be shared with other programs.

R has two functions **WriteBin()** and **readBin()** to create and read binary files.

Syntax

writeBin(object, con)

readBin(con, what, n )

**4.XML file**

You can read a xml file in R using the "XML" package. This package can be installed using following command.

install.packages("XML")

# Also load the other required package. library("methods")

# Give the input file name to the function.

result <- **xmlParse**(file = "input.xml")

# Print the result. print(result)

standard ASCII text

**5. JSON file**

JSON file stores data as text in human-readable format. Json stands for JavaScript Object Notation. R can read JSON files using the rjson package.

# Load the package required to read JSON files. library("rjson")

# Give the input file name to the function.

result <- fromJSON(file = "input.json")

# Print the result.

print(result)

**6. Web Data**

The following packages are required for processing the URL’s and links to the files.

install.packages("RCurl")

install.packages("XML")

install.packages("stringr")

install.packages("plyr")

getHTMLLinks() to gather the URLs of the files

download.file() to save the files to the local system

**7. Databases**

RMySQL" which provides native connectivity between with MySql database

install.packages("RMySQL")

Connecting R to MySql

# Create a connection Object to MySQL database.

# We will connect to the sampel database named "sakila" that comes with MySql installation. mysqlconnection = dbConnect(MySQL(), user = 'root', password = '', dbname = 'sakila', host = 'localhost')

# List the tables available in this database. dbListTables(mysqlconnection)

dbSendQuery()- used to query the database

The query gets executed in MySql and the result set is returned using the R fetch()

**Data Visualization**

6 different kind of graphs can be plotted

1. Pie Chart
2. Bar Chart
3. Box plot
4. Histogram
5. Line graph
6. Scatter Plot

**References**

1. <http://blog.easylearning.guru/r-programming-the-present/>
2. Iigsaw Academy, Analytics for Professionals, -Introduction to R
3. An overview of R: Text Analytics, Ashraf Uddin, PhD Scholar, Dept. of Computer Science South Asian University, New Delhi <https://sites.google.com/site/ashrafuddininfo/>
4. <https://www.youtube.com/watch?v=eDrhZb2onWY>
5. <https://www.tutorialspoint.com/r/r_data_types.htm>
6. https://www.tutorialspoint.com/r/r\_csv\_files.htm

**Solutions**

**Part(A)**

1. Perform the required operations using the methods – as.integer () and

Class()

1. Perform the required operations using methods –seq()
2. Perform the required operations using methods
   * 1. Rep()
     2. Paste()
3. Perform the required operations using methods
   * 1. Sum()
     2. Cumprod()
     3. Seq()
4. Perform the required operations using methods
   * 1. C()
     2. Factor()
     3. Nlevels()
     4. Length()
5. Perform the required operations using methods
   * 1. Matrix()
     2. Operator %\*%
     3. Crossprod()
     4. Outer()
6. Perform the required operations using methods
   * 1. Seq()
     2. Plot()
7. Perform the required operations using methods
   * 1. Plot()
     2. Dataframe()
     3. Write function

**Solutions**

**Part (B)**

1. Perform the required operations using methods
   * 1. Outer()
     2. Sapply()
     3. Vapply()
     4. System.time()
2. Write function weekday(day,month,year)
3. Perform the required operations using methods
   * 1. Rnorm() togenerate series of waiting time and service time
     2. Write function on max()
     3. Use sapply()
4. a. Perform the required operations using methods
5. rnorm() to generate data
6. ts() to generate time series data

b. Perform the required operations using methods

1. rnorm() to generate data
2. ts() to generate time series data
3. mean()
4. rexp()
5. sign()
6. sum()
7. Abs()
8. Rep()
9. Lapply()
10. Sapply()
11. Steps to be followed
    1. Use vector() method to declare the employee details
    2. Use data.frame() method to convert multiple vector to a dataframe
    3. Use method write.csv() to create csv file
    4. Verify the csv & csv2 file types
    5. Use read.csv() to read from csv file
    6. Use write.table() to append the record in csv
12. Perform the required operations using methods
    1. Data()
    2. Head()
    3. Nrow()
    4. Ncol()
    5. Data.frame()
    6. Scatter.smooth()
    7. Hist()
    8. As.integer()
13. Perform the required operations using methods
    * + 1. Read from a csv file use read.csv()
        2. Use ts()
        3. Plot()
        4. Use method gghistogram() available in package ‘forecast’
        5. Use method decompose() and set type as multiplicative

Plot the decomposed series using the method plot()

1. Perform the required operations using methods
   * + - 1. Read data from airquality data set
         2. Use method dim() to know the dimension of data set

Use sapply() to check the class of each field

* + - * 1. To check missing values method is.na()
        2. Check the missing values in each column , replace it with mean of the data. Use mean()
        3. Use na.omit()