**AIM: BASIC R COMMANDS**

**THEORY:**

**Introduction to R:**

**R** is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity. One of R’s strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.

R is available as Free Software under the terms of the Free Software Foundation’s

GNU General Public License in source code form. It compiles and runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and MacOS.

**R environment:**

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

* an effective data handling and storage facility,
* a suite of operators for calculations on arrays, in particular matrices,
* a large, coherent, integrated collection of intermediate tools for data analysis,
* graphical facilities for data analysis and display either on-screen or on hardcopy, and
* a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

The term “environment” is intended to characterize it as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software.

R, like S, is designed around a true computer language, and it allows users to add additional functionality by defining new functions. Much of the system is itself written in the R dialect of S, which makes it easy for users to follow the algorithmic choices made. For computationally intensive tasks, C, C++ and Fortran code can be linked and called at run time. Advanced users can write C code to manipulate R objects directly.

Many users think of R as a statistics system. We prefer to think of it as an environment within which statistical techniques are implemented. R can be extended (easily) via packages. There are about eight packages supplied with the R distribution and many more are available through the CRAN family of Internet sites covering a very wide range of modern statistics.

R has its own LaTeX-like documentation format, which is used to supply comprehensive documentation, both on-line in a number of formats and in hardcopy.

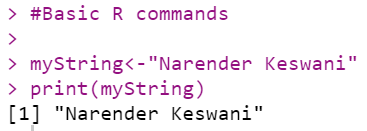
1. **PRINTING STRING:**

**SOURCE CODE:**

myString<-"Narender Keswani"

print(myString)

**OUTPUT:**



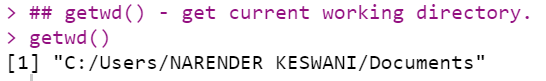
1. **GET CURRENT WORKING DIRECTORY:**

**SOURCE CODE:**

## getwd() - get current working directory.

getwd()

**OUTPUT:**



1. **GET LIST OF DIRECTORIES:**

**SOURCE CODE:**

## dir() - lists the contents of current working directory.

dir()

**OUTPUT:**



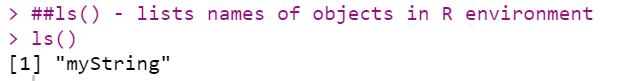
1. **GET LIST NAMES OF OBJECTS IN R ENVIRONMENT:**

**SOURCE CODE:**

##ls() - lists names of objects in R environment

ls()

**OUTPUT:**



1. **CHECKING TYPE OF OBJECT:**

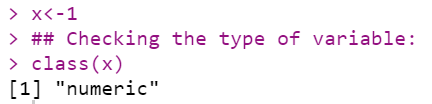
**SOURCE CODE:**

x<-1

## Checking the type of variable:

class(x)

**OUTPUT:**



1. **EXAMPLE OF AUTO-PRINTING:**

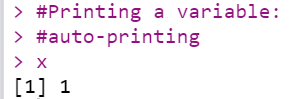
**SOURCE CODE:**

#Printing a variable:

#auto-printing

X

**OUTPUT:**



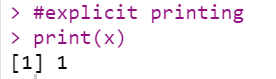
1. **EXAMPLE OF EXPLICIT PRINTING:**

**SOURCE CODE:**

#explicit printing

print(x)

**OUTPUT:**



1. **CHECK DATATYPE**
2. **CHARACTER:**

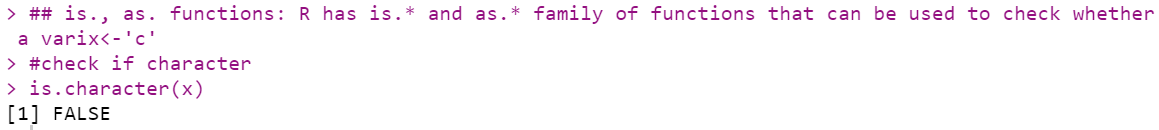
**SOURCE CODE:**

## is., as. functions: R has is.\* and as.\* family of functions that can be used to check whether a varix<-'c'

#check if character

is.character(x)

**OUTPUT:**



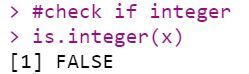
1. **INTEGER:**

**SOURCE CODE:**

#check if integer

is.integer(x)

**OUTPUT:**



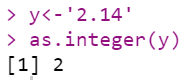
1. **CONVERT TO INTEGER:**

**SOURCE CODE:**

y<-'2.14'

as.integer(y)

**OUTPUT:**



1. **CREATE VECTOR:**
2. **USING c() FUNCTION:**

**SOURCE CODE:**

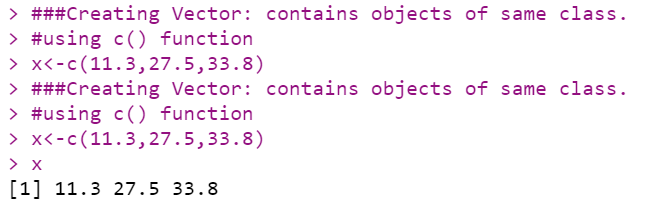
###Creating Vector: contains objects of same class.

#using c() function

x<-c(11.3,27.5,33.8)

x

**OUTPUT:**



1. **USING VECTOR() FUNCTION:**

**SOURCE CODE:**

#using vector() function

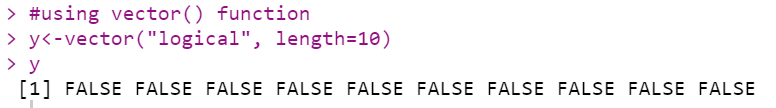
y<-vector("logical", length=10)

y

y<-c(4,5,6)

y

**OUTPUT:**





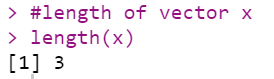
1. **FIND LENGTH OF VECTOR:**

**SOURCE CODE:**

#length of vector x

length(x)

**OUTPUT:**



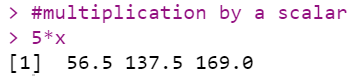
1. **ARTHIMETIC OPERATIONS:**
2. **MULTIPLICATION OF SCALAR:**

**SOURCE CODE:**

#multiplication by a scalar

5\*x

**OUTPUT:**



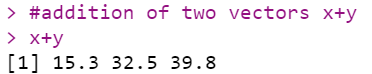
1. **ADDITION OF VECTORS:**

**SOURCE CODE:**

#addition of two vectors x+y

x+y

**OUTPUT:**



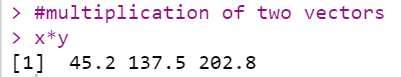
1. **MULTIPLICATION OF VECTORS:**

**SOURCE CODE:**

#multiplication of two vectors

x\*y

**OUTPUT:**



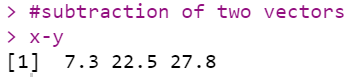
1. **SUBTRACTION OF VECTORS:**

**SOURCE CODE:**

#subtraction of two vectors

x-y

**OUTPUT:**



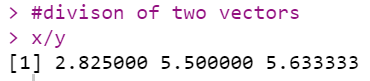
1. **DIVISION OF VECTORS:**

**SOURCE CODE:**

#divison of two vectors

x/y

**OUTPUT:**



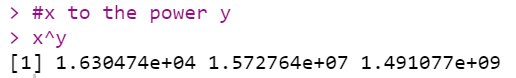
1. **POWER:**

**SOURCE CODE:**

#x to the power y

x^y

**OUTPUT:**



1. **CREATION OF MATRIX:**

**SOURCE CODE:**

###Creating Matrix: Two-dimensional array having elements of same class.

#using matrix() function

m<-matrix(c(11,12,13,55,60,65,66,72,78),nrow=3,ncol=3)

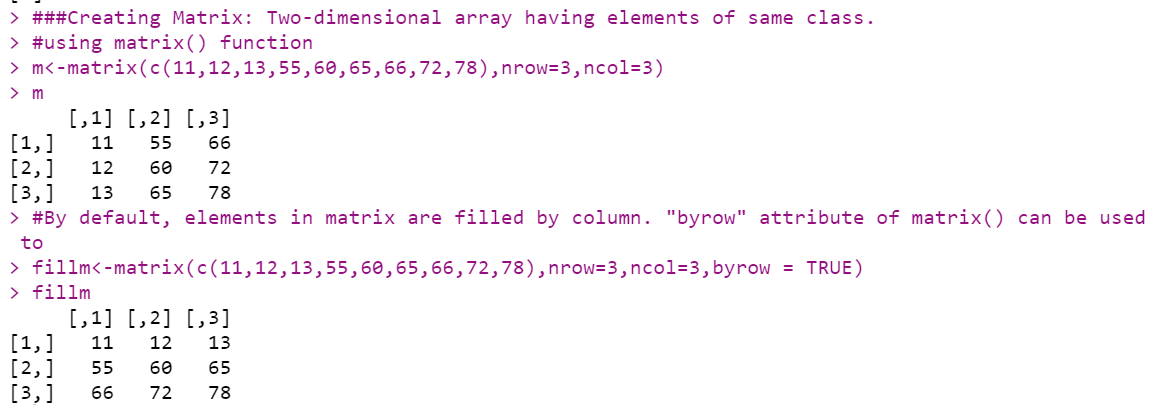
m

#By default, elements in matrix are filled by column. "byrow" attribute of matrix() can be used to

fillm<-matrix(c(11,12,13,55,60,65,66,72,78),nrow=3,ncol=3,byrow = TRUE)

fillm

**OUTPUT:**



1. **FIND DIMENSION & ATTRIBUTE OF A MATRIX:**

**SOURCE CODE:**

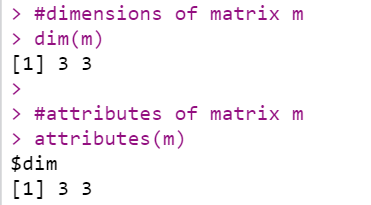
#dimensions of matrix m

dim(m)

#attributes of matrix m

attributes(m)

**OUTPUT:**



1. **CBIND() & RBIND():**

**SOURCE CODE:**

#cbinding and rbinding:

#By using cbind() and rbind() functions

x<-c(1,2,3)

y<-c(11,12,13)

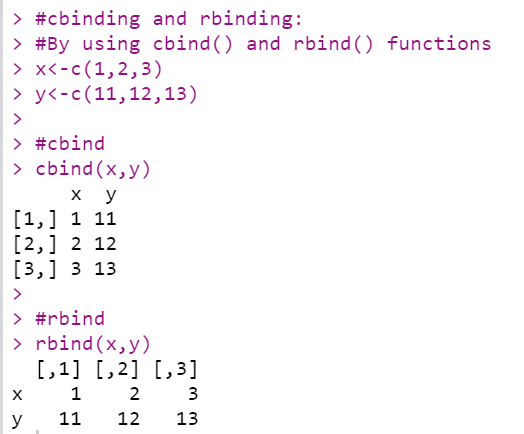
#cbind

cbind(x,y)

#rbind

rbind(x,y)

**OUTPUT:**



1. **OPERATIONS ON MATRIX:**
2. **MULTIPLICATION BY A SCALAR:**

**SOURCE CODE:**

##Matrix operations/functions:

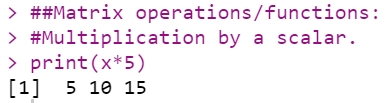
#Multiplication by a scalar.

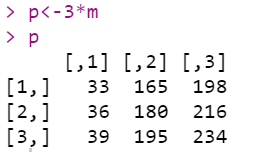
print(x\*5)

p<-3\*m

p

**OUTPUT:**





1. **ADDITION OF MATRICES:**

**SOURCE CODE:**

print(x+y)

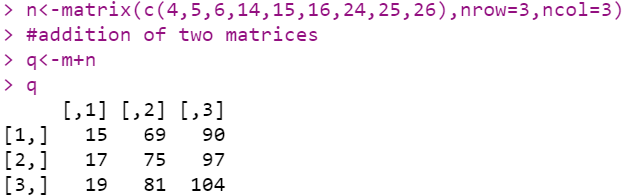
n<-matrix(c(4,5,6,14,15,16,24,25,26),nrow=3,ncol=3)

q<-m+n

q

**OUTPUT:**





1. **SUBTRACTION OF MATRICES:**

**SOURCE CODE:**

print(x-y)

**OUTPUT:**



1. **MULTIPLICATION OF MATRICES:**

**SOURCE CODE:**

print(x\*y)

**OUTPUT:**



1. **DIVISION OF MATRICES:**

**SOURCE CODE:**

print(x/y)

**OUTPUT:**



1. **MATRIX MULTIPLICATION BY USING %\*%**

**SOURCE CODE:**

o<-matrix(c(4,5,6,14,15,16),nrow=3,ncol=2)

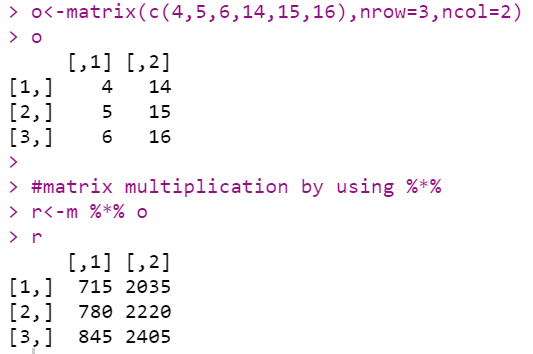
o

#matrix multiplication by using %\*%

r<-m %\*% o

r

**OUTPUT:**



1. **TRANSPOSE OF MATRIX:**

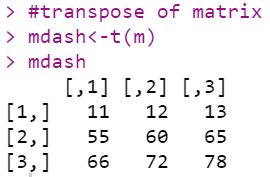
**SOURCE CODE:**

#transpose of matrix

mdash<-t(m)

mdash

**OUTPUT:**



1. **FIND DETERMINANT FROM MATRIX:**

**SOURCE CODE:**

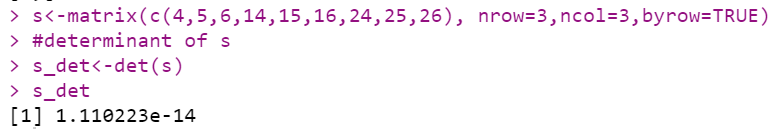
s<-matrix(c(4,5,6,14,15,16,24,25,26), nrow=3,ncol=3,byrow=TRUE)

#determinant of s

s\_det<-det(s)

s\_det

**OUTPUT:**



1. **EXAMPLE OF LIST:**

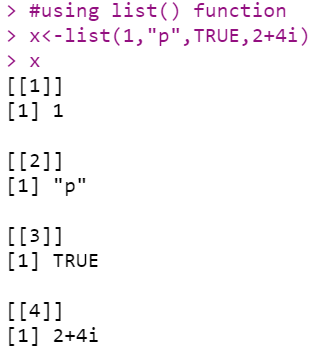
**SOURCE CODE:**

#using list() function

x<-list(1,"p",TRUE,2+4i)

x

**OUTPUT:**



1. **EXAMPLE OF FACTOR & LEVELS:**

**SOURCE CODE:**

###Factor: Represents categorical data. Can be ordered or unordered.

status<-c("low","high","medium","high","low")

#using factor() function

x<-factor(status, ordered=TRUE,levels=c("low","medium","high"))

x

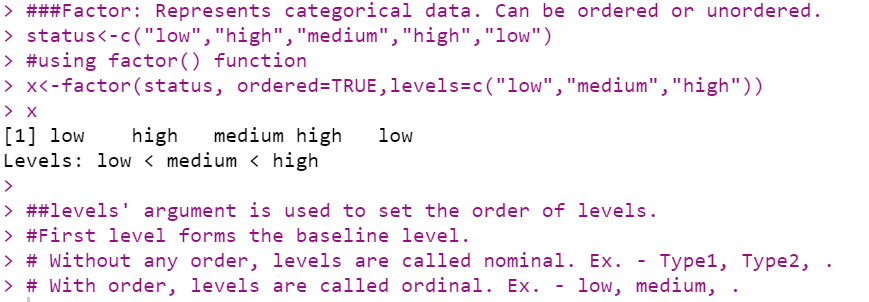
##levels' argument is used to set the order of levels.

#First level forms the baseline level.

# Without any order, levels are called nominal. Ex. - Type1, Type2, .

# With order, levels are called ordinal. Ex. - low, medium, .

**OUTPUT:**



1. **EXAMPLE OF DATAFRAME:**

**SOURCE CODE:**

###Data frame: Used to store tabular data. Can contain different classes.

student\_id<-c(1,2,3)

student\_names<-c("Ram","Shyam","Laxman")

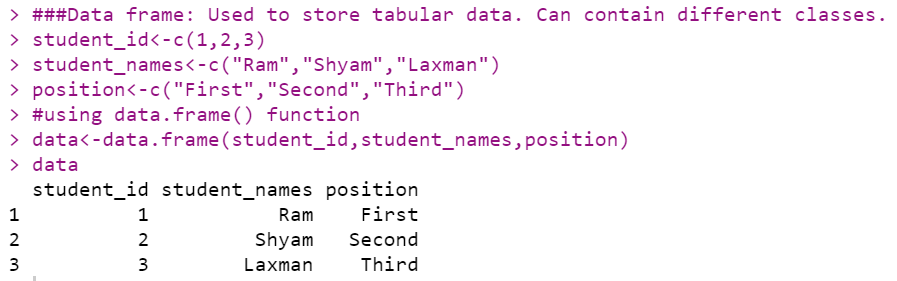
position<-c("First","Second","Third")

#using data.frame() function

data<-data.frame(student\_id,student\_names,position)

data

**OUTPUT:**



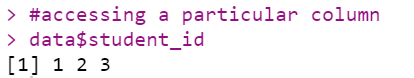
1. **FUNCTIONS OF DATAFRAME:**
2. **ACCESSING A PARTICULAR COLUMN:**

**SOURCE CODE:**

#accessing a particular column

data$student\_id

**OUTPUT:**



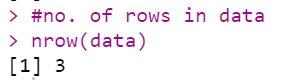
1. **NUMBER OF ROWS IN DATAFRAME:**

**SOURCE CODE:**

#no. of rows in data

nrow(data)

**OUTPUT:**



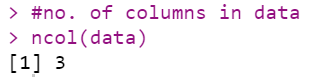
1. **NUMBER OF COLUMNS IN DATAFRAME:**

**SOURCE CODE:**

#no. of columns in data

ncol(data)

**OUTPUT:**



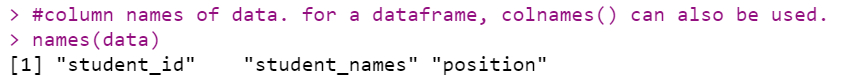
1. **GET COLUMN NAMES OF A DATAFRAME:**

**SOURCE CODE:**

#column names of data. for a dataframe, colnames() can also be used.

names(data)

**OUTPUT:**



1. **CREATE 2-DIMENSIONAL TABLE:**

**SOURCE CODE:**

###Table command is used to create a 2dimensional table in R

smoke <- matrix(c(51,43,22,92,28,21,68,22,9),ncol=3,byrow=TRUE)

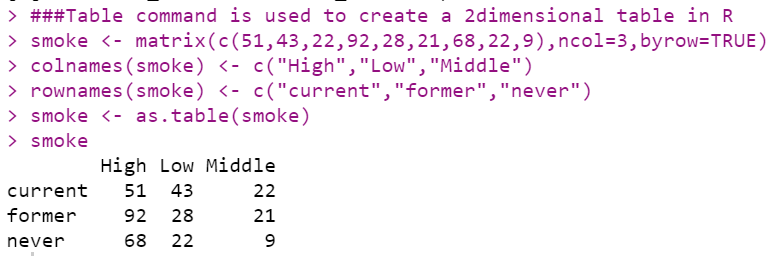
colnames(smoke) <- c("High","Low","Middle")

rownames(smoke) <- c("current","former","never")

smoke <- as.table(smoke)

smoke

**OUTPUT:**



1. **INSTALL LIBARIES:**

**SOURCE CODE:**

#install.packages("package\_name")

library(XLConnect)

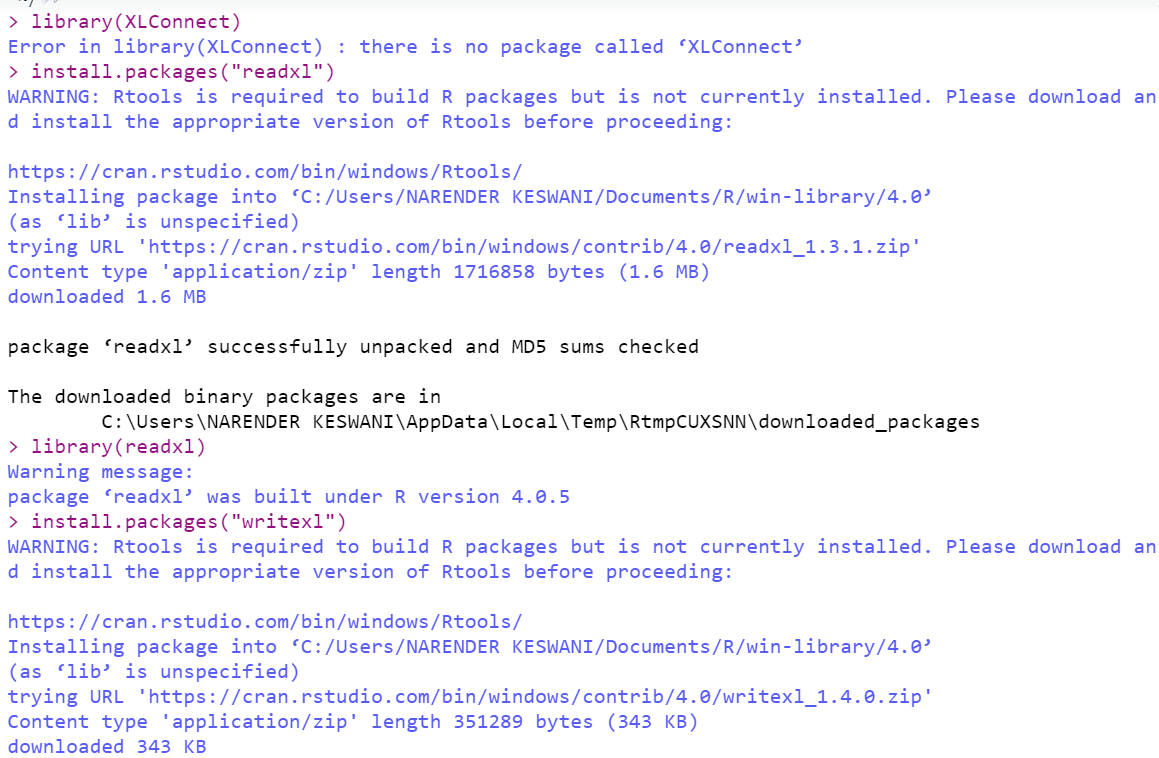
install.packages("readxl")

library(readxl)

install.packages("writexl")

library(writexl)

**OUTPUT:**



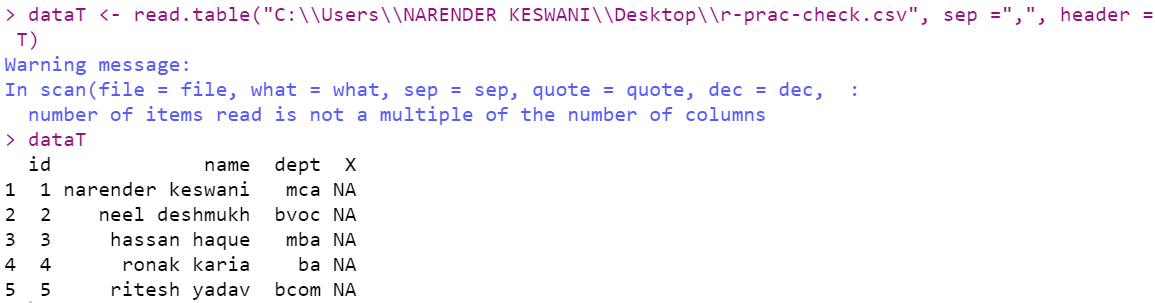
1. **EXAMPLES OF CSV:**
2. **READING DATA FROM CSV:**

**SOURCE CODE:**

dataT <- read.table("C:\\Users\\NARENDER KESWANI\\Desktop\\r-prac-check.csv", sep =",", header = T)

dataT

**OUTPUT:**



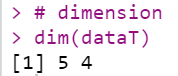
1. **GET DIMENSIONS OF CSV FILE:**

**SOURCE CODE:**

# dimension

dim(dataT)

**OUTPUT:**



1. **HEAD & TAIL:**

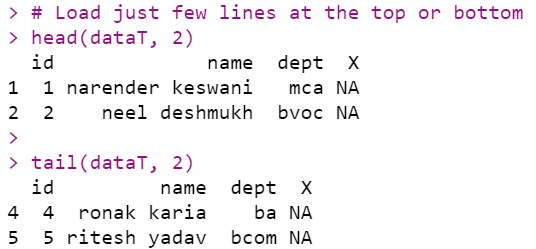
**SOURCE CODE:**

# Load just few lines at the top or bottom

head(dataT, 2)

tail(dataT, 2)

**OUTPUT:**



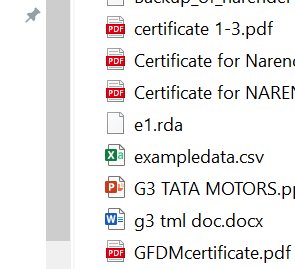
1. **WRITING DATA TO CSV FILE:**

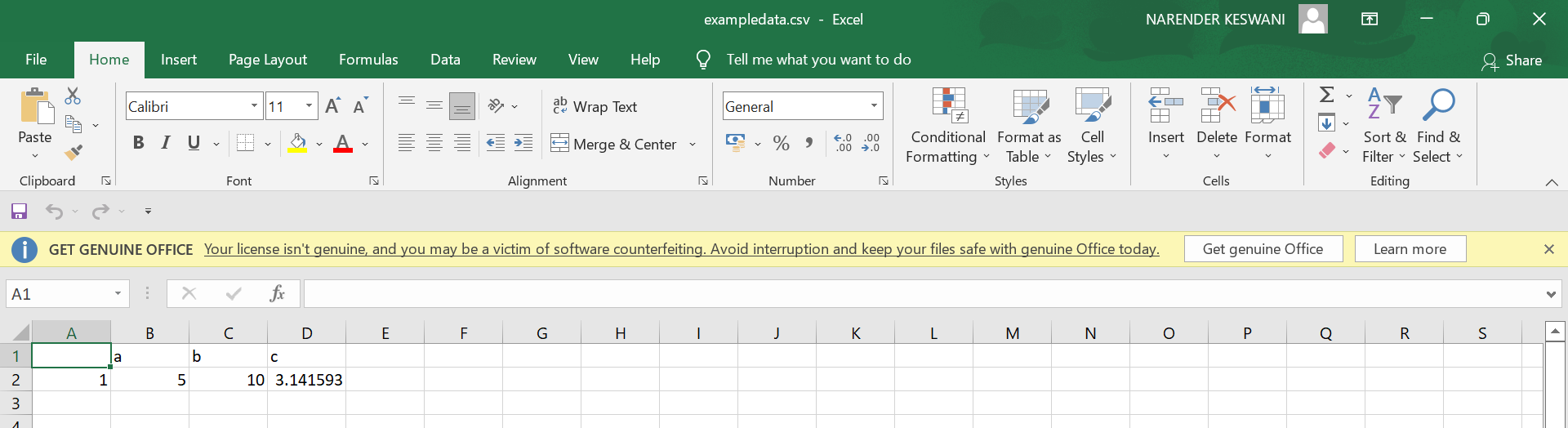
**SOURCE CODE:**

z <- data.frame(a = 5, b = 10, c = pi)

write.csv(z,file="exampledata.csv")

**OUTPUT:**





1. **READING AND WRITING DATA FROM EXCEL USING XLCONNECT:**

**SOURCE CODE:**

install.packages('Rcpp')

library(Rcpp)

#Reading and writing data from Excel using XLConnect

dataX <- XLConnect:: readWorksheetFromFile("C:\\Users\\NARENDER KESWANI\\Downloads\\01 Contoso Employee Info.xlsx",sheet=1)

dataX

# Following is called Subsetting - It will print rows from 1 to 2 and all columns

dataY<- dataX[1:2,]

dataY

#Reading and writing data from Excel using readXL and writeXL

data2 <- read\_excel("C:\\Users\\NARENDER KESWANI\\Downloads\\01 Contoso Employee Info.xlsx", sheet = "1")

data2

data3<- data2[1:5,]

write\_xlsx(data3, "e2.xlsx")

# create an empty data frame

data <- data.frame(Name=character(), Age=numeric())

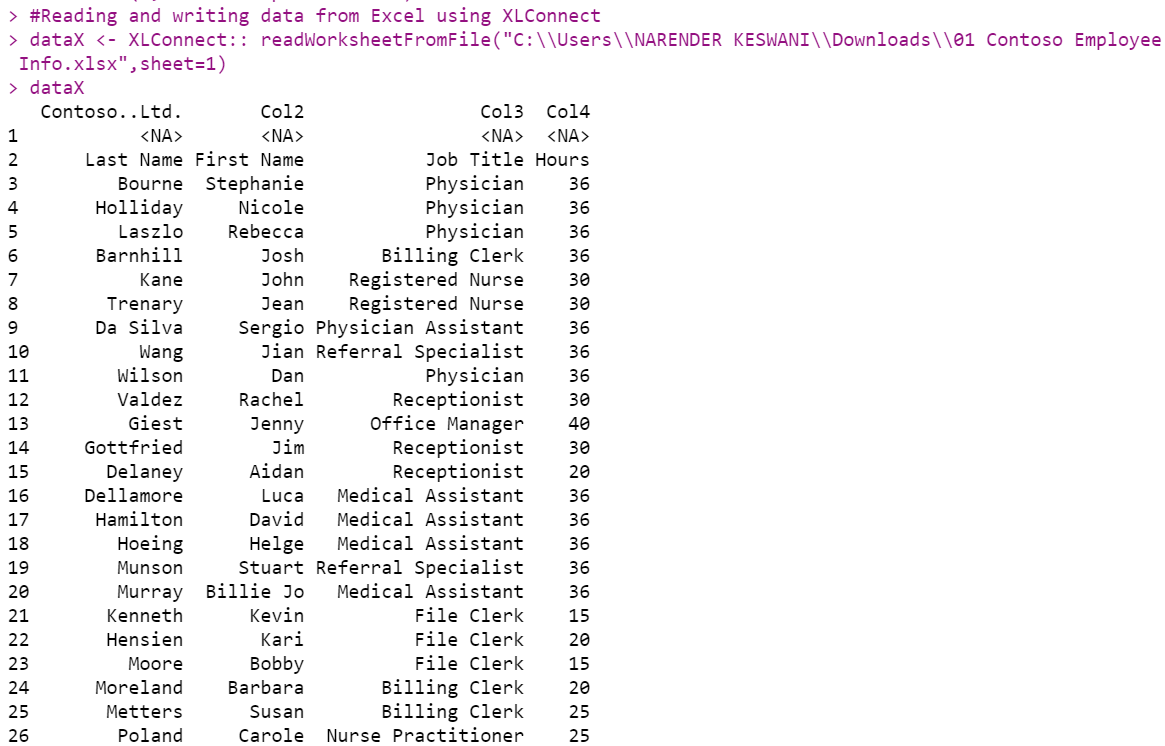
# load interface and assign edited values to data back - uncomment following

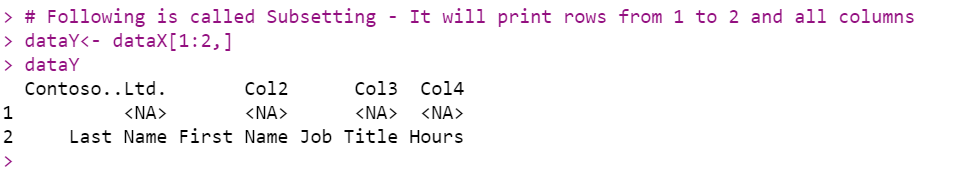
data <- edit(data)

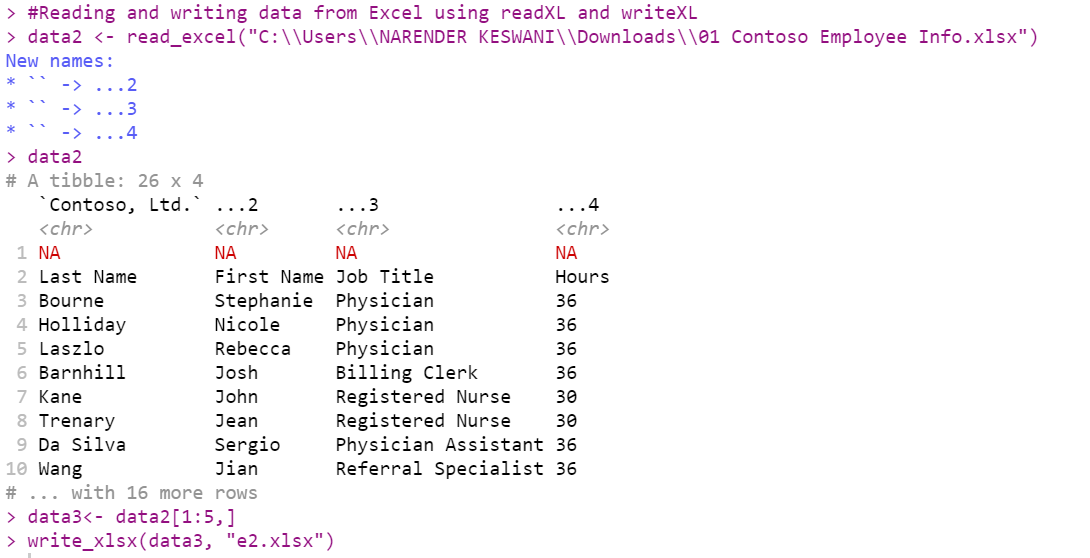
#print those values

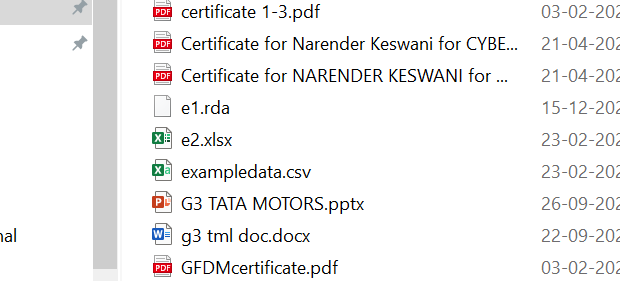
data

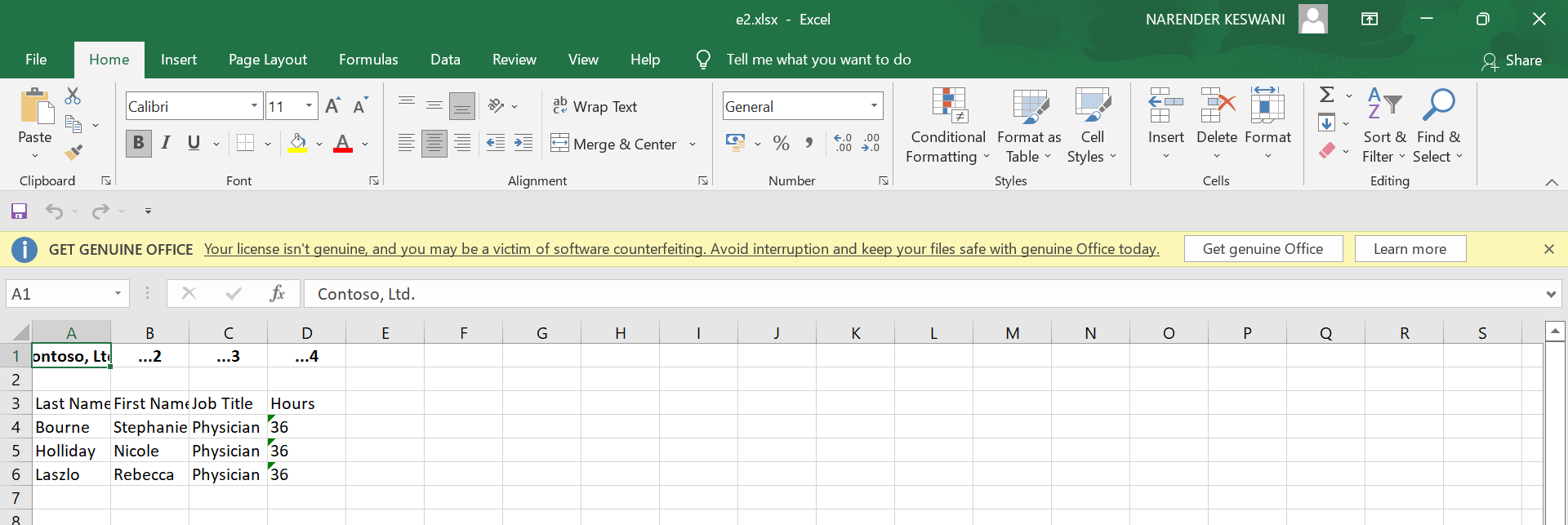
**OUTPUT:**

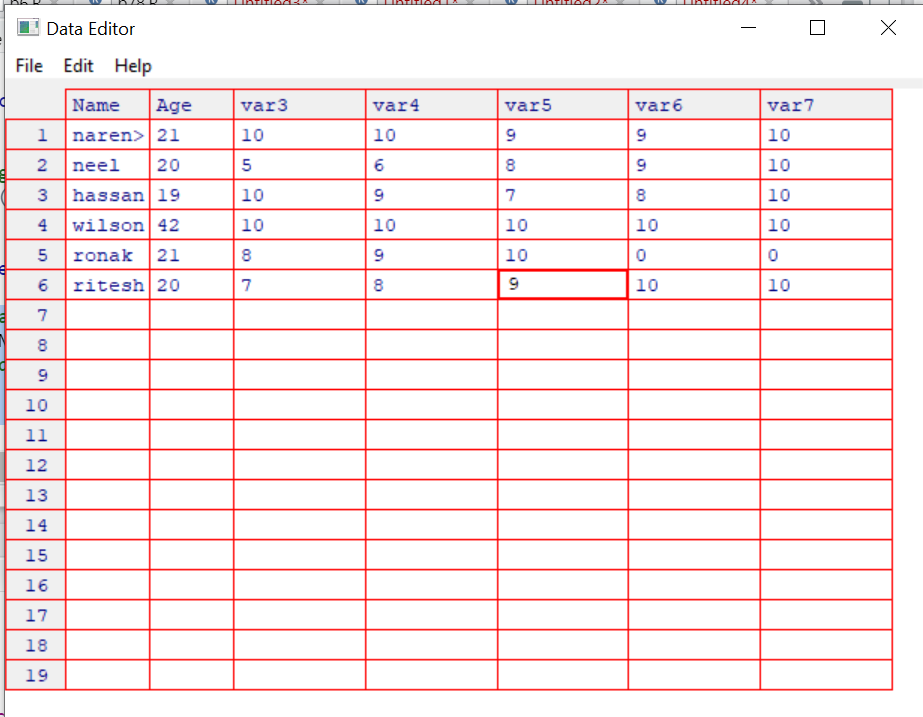


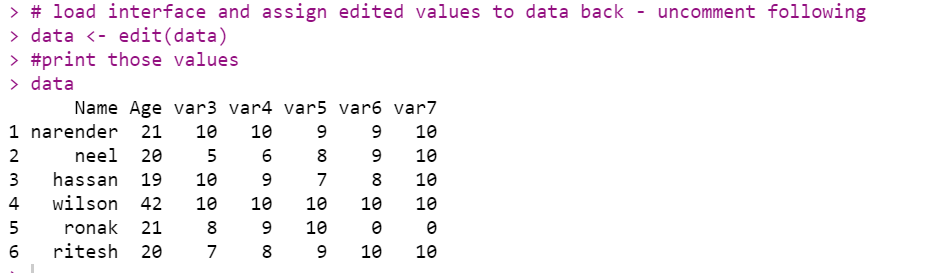












**CONCLUSION:**

From this practical, I have learned the basics of R programming.