PRACTICAL – 5

Aim: Introduction to R programming and Data acquisition.

Theory:

R is a Programming Language that is mostly used for machine learning, data analysis, and statistical computing. It is an interpreted language and is platform independent that means it can be used on platforms like Windows, Linux, and macOS.

Why Learn R Programming Language?

- R programming is used as a leading tool for machine learning, statistics, and data analysis.
- R is an open-source language that means it is free of cost and anyone from any organization can install it without purchasing a license.
- It is available across widely used platforms like windows, Linux, and macOS.
- R programming language is not only a statistic package but also allows us to integrate with other languages (C, C++). Thus, you can easily interact with many data sources and statistical packages.
- Its user base is growing day by day and has vast community support.
- R Programming Language is currently one of the most requested programming languages in the Data Science job market that makes it the hottest trend nowadays.

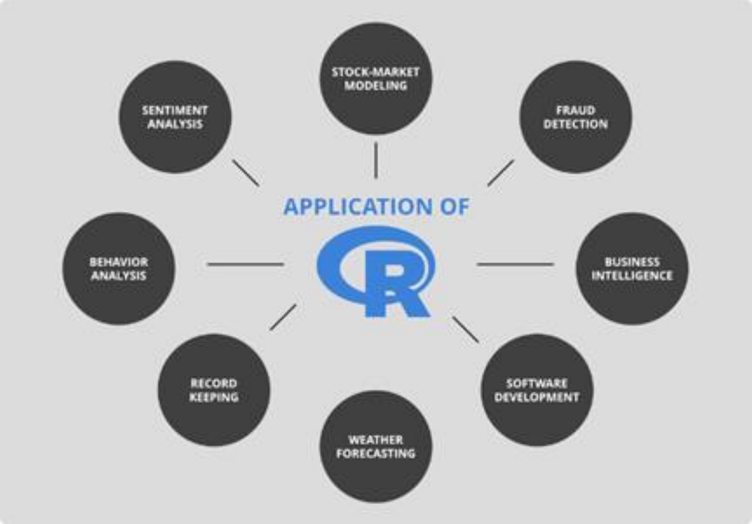
Key Features and Applications

Some key features of R that make the R one of the most demanding job in data science market are:

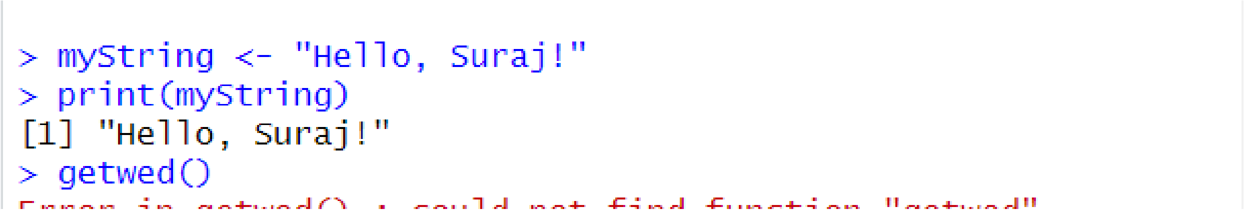
- **Basic Statistics:** The most common basic statistics terms are the mean, mode, and median. These are all known as “Measures of Central Tendency.” So using the R language we can measure central tendency very easily.
- **Static graphics:** R is rich with facilities for creating and developing various kinds of static graphics including graphic maps, mosaic plots, biplots, and the list goes on.

- **Probability distributions:** Using R we can easily handle various types of probability distribution such as Binomial Distribution, Normal Distribution, Chi-squared Distribution, and many more.
- **R Packages:** One of the major features of R is it has a wide availability of libraries. R has CRAN(Comprehensive R Archive Network), which is a repository holding more than 10,0000 packages.
- **Distributed Computing:** Distributed computing is a model in which components of a software system are shared among multiple computers to improve efficiency and performance. Two new packages ddR and multidplyr used for distributed programming in R were released in November 2015.

Application of R:

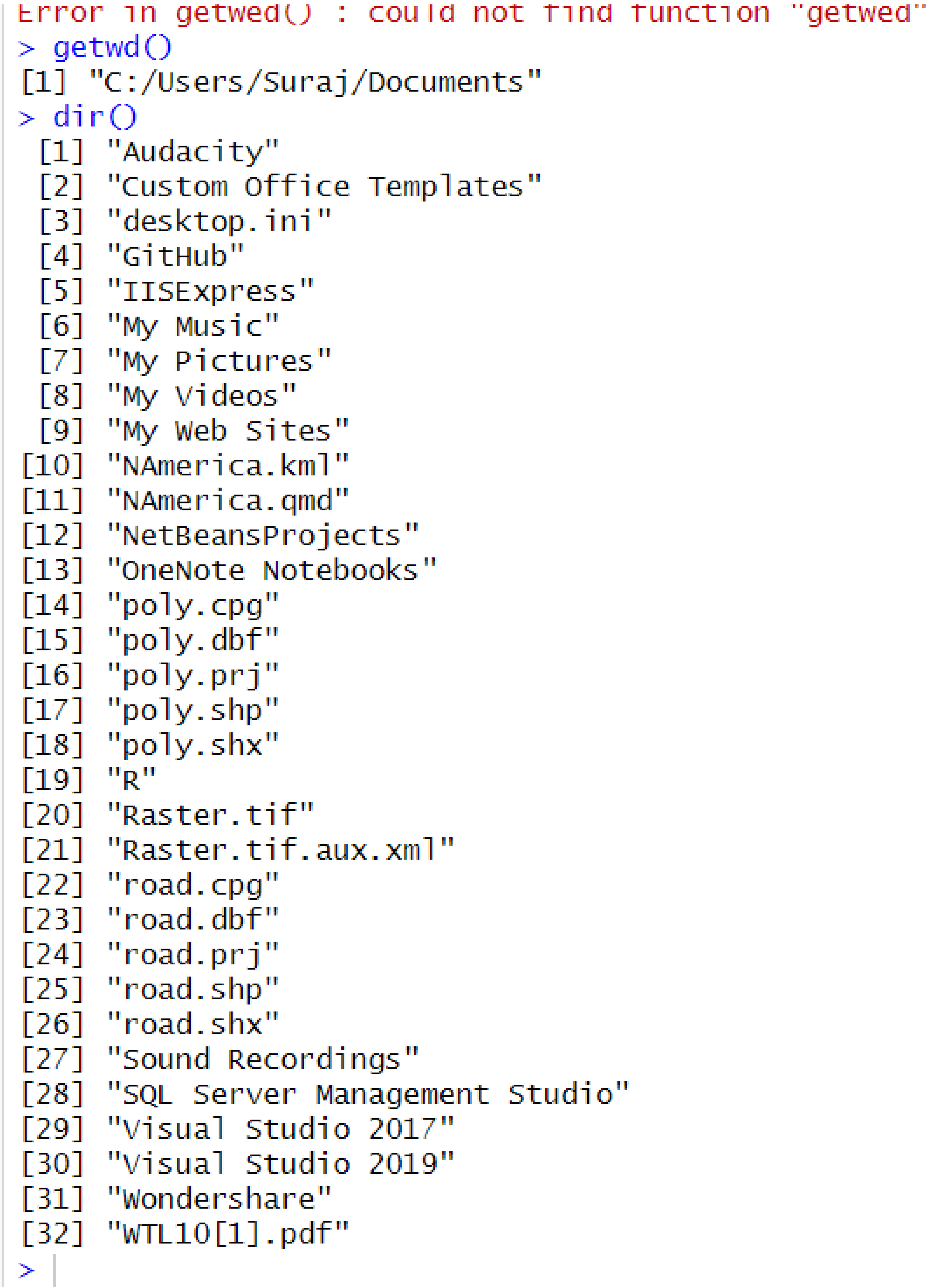


First program

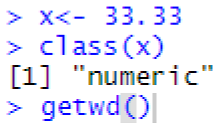


```
> myString <- "Hello, Suraj!"
> print(myString)
[1] "Hello, Suraj!"
> getwed()
Error in getwed() : could not find function "getwed"
```

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



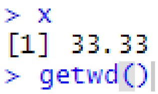
```
Error in getwed() : could not find function "getwed"
> getwd()
[1] "C:/Users/Suraj/Documents"
> dir()
 [1] "Audacity"
 [2] "Custom Office Templates"
 [3] "desktop.ini"
 [4] "GitHub"
 [5] "IISExpress"
 [6] "My Music"
 [7] "My Pictures"
 [8] "My Videos"
 [9] "My Web Sites"
[10] "NAmerica.kml"
[11] "NAmerica.qmd"
[12] "NetBeansProjects"
[13] "OneNote Notebooks"
[14] "poly.cpg"
[15] "poly.dbf"
[16] "poly.prj"
[17] "poly.shp"
[18] "poly.shx"
[19] "R"
[20] "Raster.tif"
[21] "Raster.tif.aux.xml"
[22] "road.cpg"
[23] "road.dbf"
[24] "road.prj"
[25] "road.shp"
[26] "road.shx"
[27] "Sound Recordings"
[28] "SQL Server Management Studio"
[29] "Visual Studio 2017"
[30] "Visual Studio 2019"
[31] "Wondershare"
[32] "WTL10[1].pdf"
> 
```



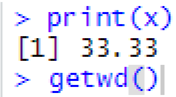
```
> x<- 33.33
> class(x)
[1] "numeric"
> getwd()
```

is. function tells whether variable is of mentioned type and returns Boolean value.

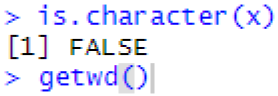
As. function convers variable into mentioned type.



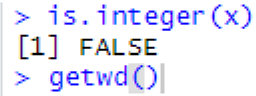
```
> x
[1] 33.33
> getwd()
```



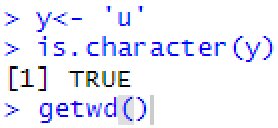
```
> print(x)
[1] 33.33
> getwd()
```



```
> is.character(x)
[1] FALSE
> getwd()
```



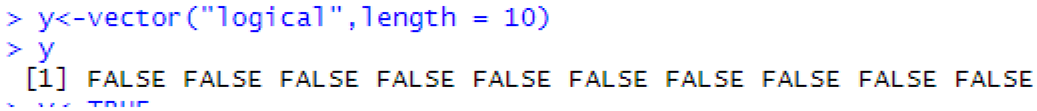
```
> is.integer(x)
[1] FALSE
> getwd()
```



```
> y<- 'u'
> is.character(y)
[1] TRUE
> getwd()
```

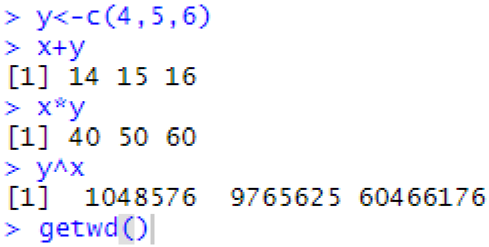
Creating vector: contains objects of same class.

It can be created in two ways using c() function and vector() function



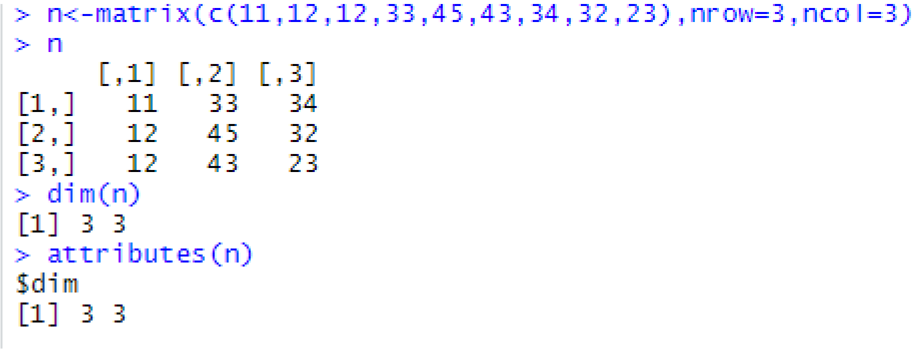
```
> y<-vector("logical",length = 10)
> y
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> y<- TRUE
```

Performing arithmetic operation on vectors.



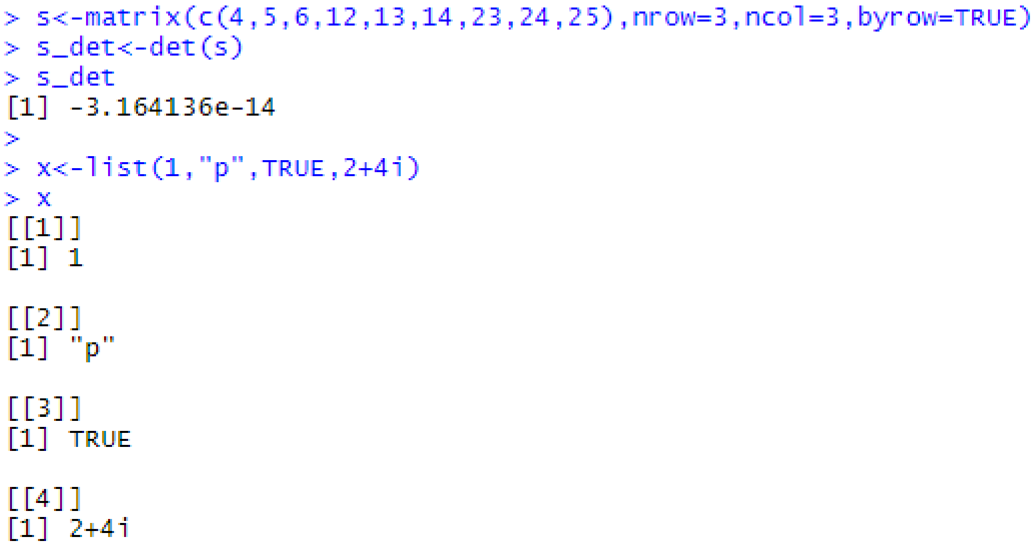
```
> y<-c(4,5,6)
> x+y
[1] 14 15 16
> x*y
[1] 40 50 60
> y^x
[1]  1048576  9765625 60466176
> getwd()
```

Creating Matrix: Two-Dimensional array having same class.



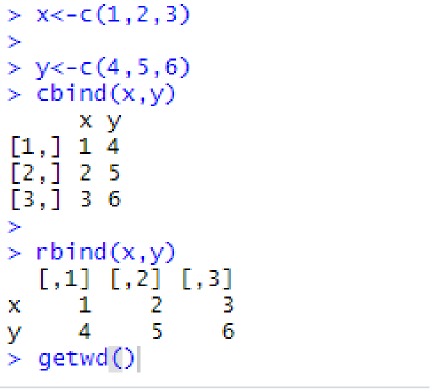
```
> n<-matrix(c(11,12,12,33,45,43,34,32,23),nrow=3,ncol=3)
> n
     [,1] [,2] [,3]
[1,]   11   33   34
[2,]   12   45   32
[3,]   12   43   23
> dim(n)
[1] 3 3
> attributes(n)
$dim
[1] 3 3
```

By default, in matrix elements are inserted column wise but we can insert row wise using byrow argument



```
> s<-matrix(c(4,5,6,12,13,14,23,24,25),nrow=3,ncol=3,byrow=TRUE)
> s_det<-det(s)
> s_det
[1] -3.164136e-14
>
> x<-list(1,"p",TRUE,2+4i)
> x
[[1]]
[1] 1

[[2]]
[1] "p"

[[3]]
[1] TRUE

[[4]]
[1] 2+4i
```

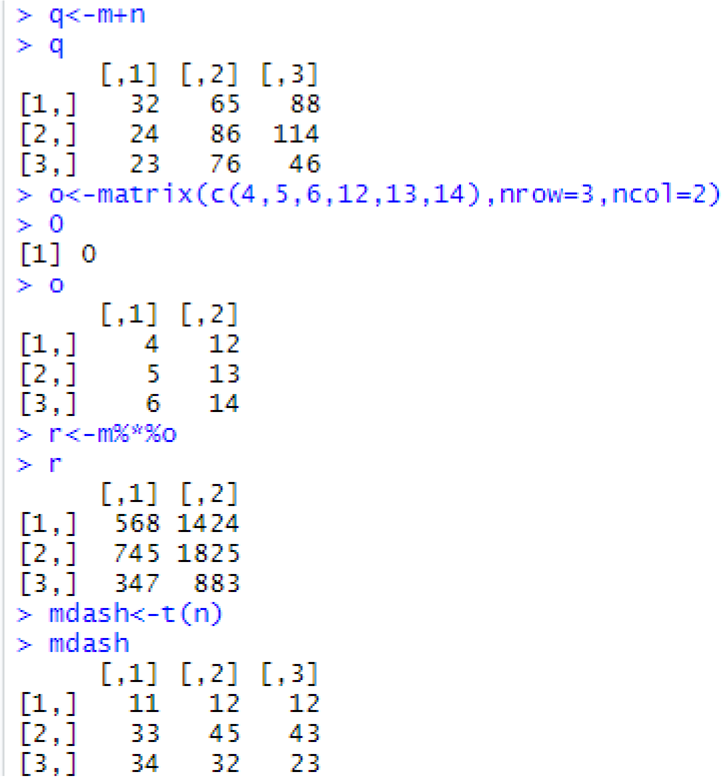
Matrix with functions



```
> x<-c(1,2,3)
>
> y<-c(4,5,6)
> cbind(x,y)
     x y
[1,] 1 4
[2,] 2 5
[3,] 3 6
>
> rbind(x,y)
  [,1] [,2] [,3]
x    1    2    3
y    4    5    6
> getwd()
```

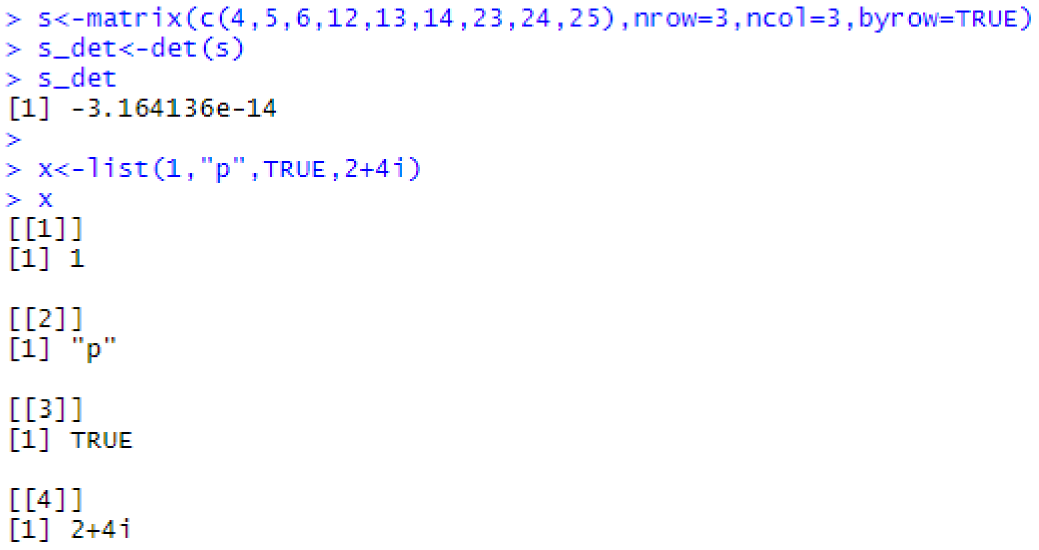
Performing matrix operation

#Addition, subtraction and multiplication of two matrices.
#Transpose, determinant of a matrix. etc.
#multiplication by a scalar



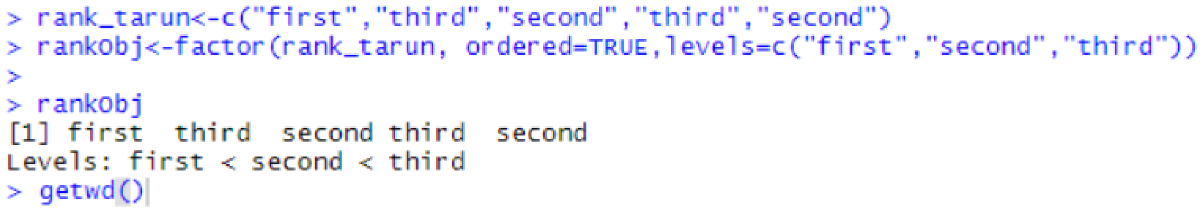
```
> q<-m+n
> q
     [,1] [,2] [,3]
[1,]   32   65   88
[2,]   24   86  114
[3,]   23   76   46
> o<-matrix(c(4,5,6,12,13,14),nrow=3,ncol=2)
> 0
[1] 0
> o
     [,1] [,2]
[1,]    4   12
[2,]    5   13
[3,]    6   14
> r<-m%*%o
> r
     [,1] [,2]
[1,]  568 1424
[2,]  745 1825
[3,]  347  883
> mdash<-t(n)
> mdash
     [,1] [,2] [,3]
[1,]   11   12   12
[2,]   33   45   43
[3,]   34   32   23
```

LIST: A special type of vector containing elements of different classes. Elements of a list can be accessed by giving element index or name in [[]



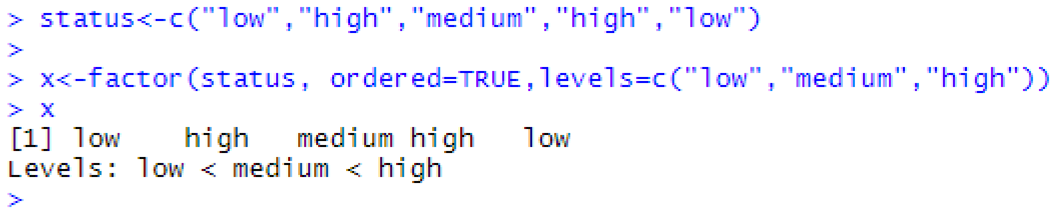
```
> s<-matrix(c(4,5,6,12,13,14,23,24,25),nrow=3,ncol=3,byrow=TRUE)
> s_det<-det(s)
> s_det
[1] -3.164136e-14
>
> x<-list(1,"p",TRUE,2+4i)
> x
[[1]]
[1] 1

[[2]]
[1] "p"

[[3]]
[1] TRUE

[[4]]
[1] 2+4i
```

Factor: Represents categorical data. These data can be ordered or unordered.

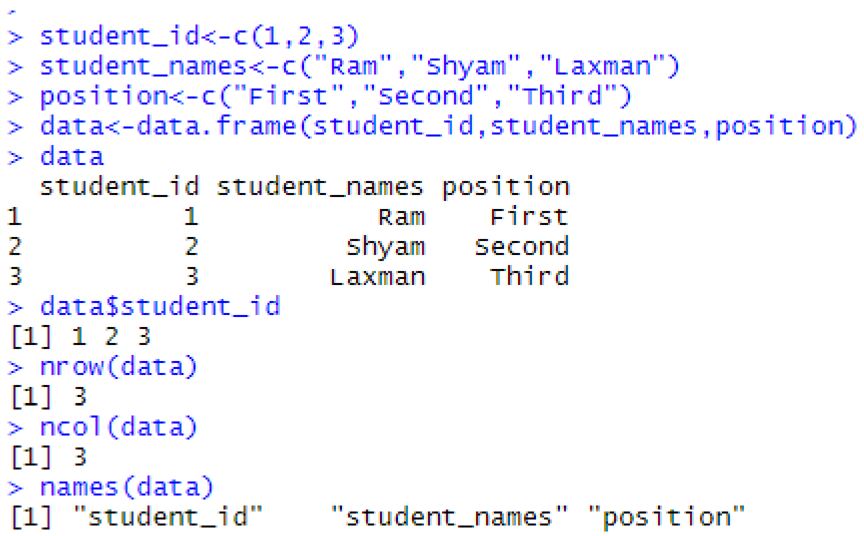


```
> rank_tarun<-c("first","third","second","third","second")
> rankObj<-factor(rank_tarun, ordered=TRUE,levels=c("first","second","third"))
>
> rankObj
[1] first  third  second third  second
Levels: first < second < third
> getwd()
```

Data Frame: used to store data into tabular form. It can store data of different classes.

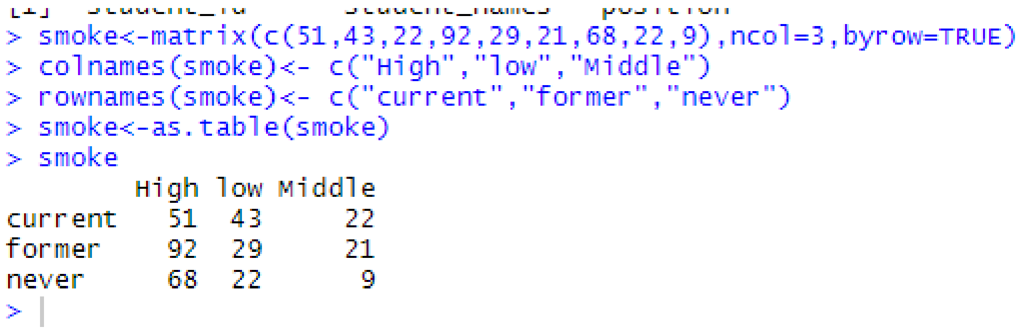


```
> status<-c("low","high","medium","high","low")
>
> x<-factor(status, ordered=TRUE,levels=c("low","medium","high"))
> x
[1] low    high   medium high   low
Levels: low < medium < high
>
```



```
> student_id<-c(1,2,3)
> student_names<-c("Ram","Shyam","Laxman")
> position<-c("First","Second","Third")
> data<-data.frame(student_id,student_names,position)
> data
  student_id student_names position
1          1           Ram    First
2          2         Shyam   Second
3          3        Laxman    Third
> data$student_id
[1] 1 2 3
> nrow(data)
[1] 3
> ncol(data)
[1] 3
> names(data)
[1] "student_id"    "student_names" "position"
```

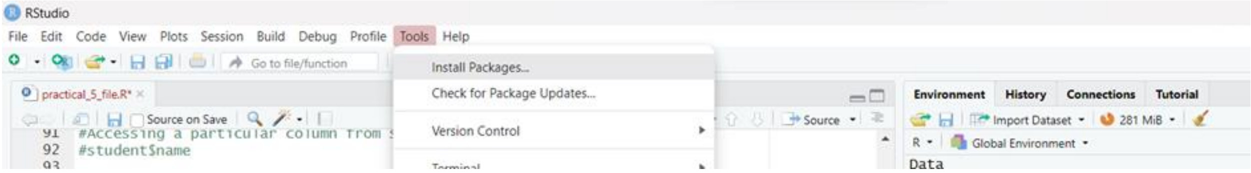
Creating two-dimensional table in R using Table command.



```
[1] student_id       student_names    position
> smoke<-matrix(c(51,43,22,92,29,21,68,22,9),ncol=3,byrow=TRUE)
> colnames(smoke)<- c("High","low","Middle")
> rownames(smoke)<- c("current","former","never")
> smoke<-as.table(smoke)
> smoke
        High low Middle
current   51  43     22
former    92  29     21
never     68  22      9
> 
```

Installing Packages in R.

In RStudio, click Tools -> Install Packages -> mention the package name.

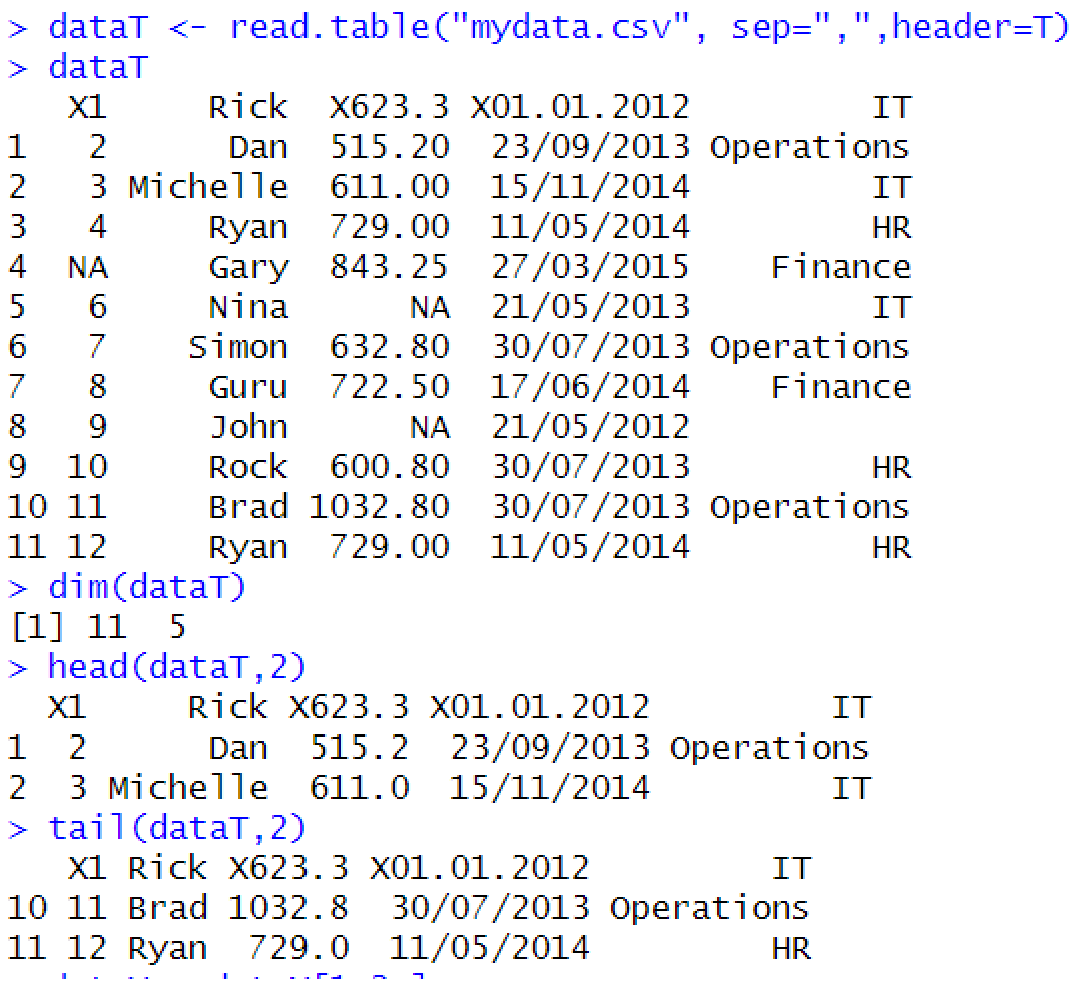


```
91 #Accessing a particular column from s
92 #student$name
93
```

install.packages("XLConnect")
library(XLConnect)
install.packages("readxl")
library(readxl)
install.packages("writexl")
library(writexl)

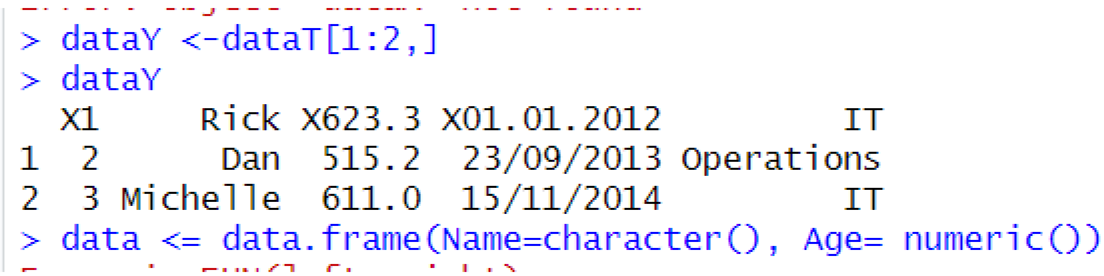
Creating Csv file in the system.

Reading and Writing data from Excel using XLConnect



```
> dataT <- read.table("mydata.csv", sep=",",header=T)
> dataT
   X1     Rick  X623.3 X01.01.2012          IT
1   2      Dan  515.20  23/09/2013 Operations
2   3 Michelle  611.00  15/11/2014          IT
3   4     Ryan  729.00  11/05/2014          HR
4  NA     Gary  843.25  27/03/2015     Finance
5   6     Nina      NA  21/05/2013          IT
6   7    Simon  632.80  30/07/2013 Operations
7   8     Guru  722.50  17/06/2014     Finance
8   9     John      NA  21/05/2012
9  10     Rock  600.80  30/07/2013          HR
10 11     Brad 1032.80  30/07/2013 Operations
11 12     Ryan  729.00  11/05/2014          HR
> dim(dataT)
[1] 11  5
> head(dataT,2)
  X1     Rick X623.3 X01.01.2012          IT
1  2      Dan  515.2  23/09/2013 Operations
2  3 Michelle  611.0  15/11/2014          IT
> tail(dataT,2)
   X1 Rick X623.3 X01.01.2012          IT
10 11 Brad 1032.8  30/07/2013 Operations
11 12 Ryan  729.0  11/05/2014          HR
```

Creating an empty data.frame



```
> dataY <-dataT[1:2,]
> dataY
  X1     Rick X623.3 X01.01.2012         IT
1  2      Dan  515.2  23/09/2013 Operations
2  3 Michelle  611.0  15/11/2014         IT
> data <= data.frame(Name=character(), Age= numeric())
```

Conclusion: I have successfully installed R, and applied Data Acquisition.