Experiment 9: Demonstrate importing and exporting data using R

Aim: To understand the process of importing and exporting of data in R.

Experiment 9: Demonstrate importing and exporting data using R

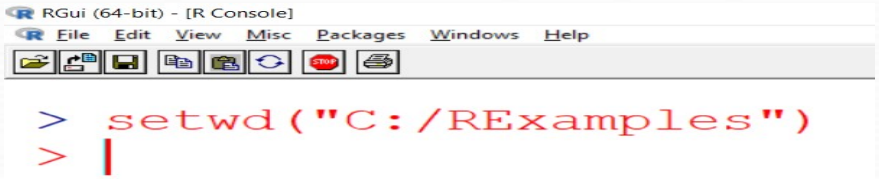
Setting up directories:

We can change the current working directory as follows:

>setwd(“<location of the dataset>”)

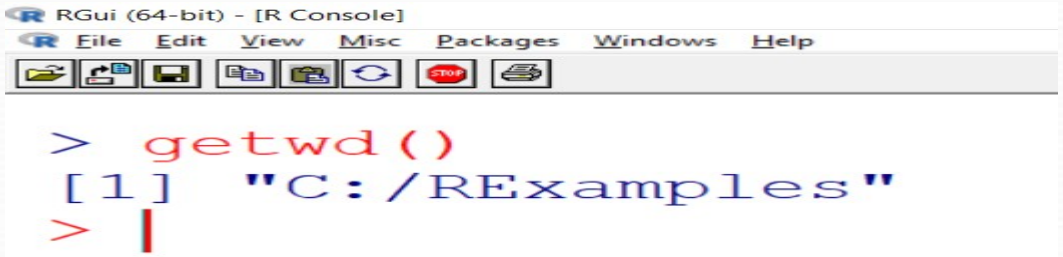
Example:

>setwd("C:/RExamples")



The following command returns the current working directory:

>getwd()



Experiment 9: Demonstrate importing and exporting data using R

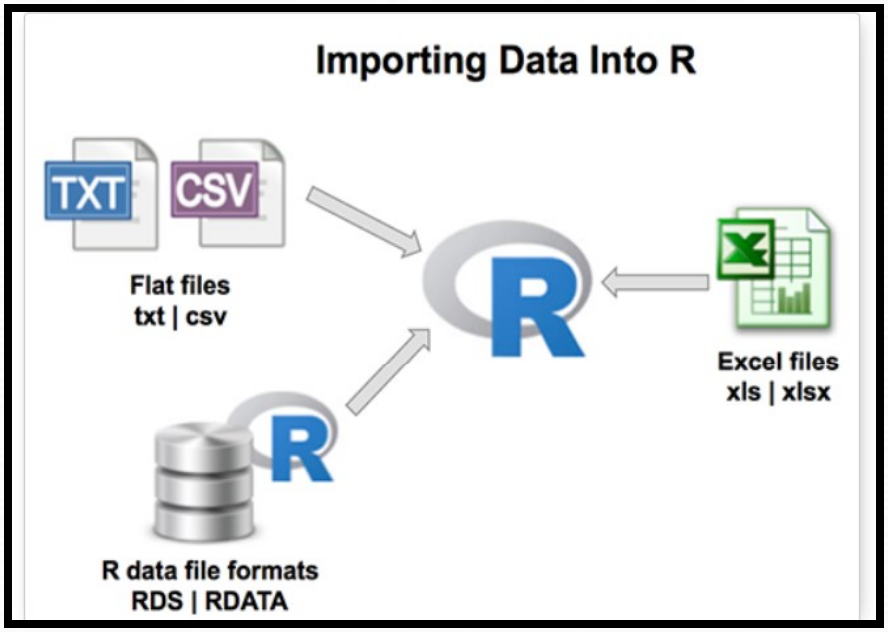
Importing Data Files:

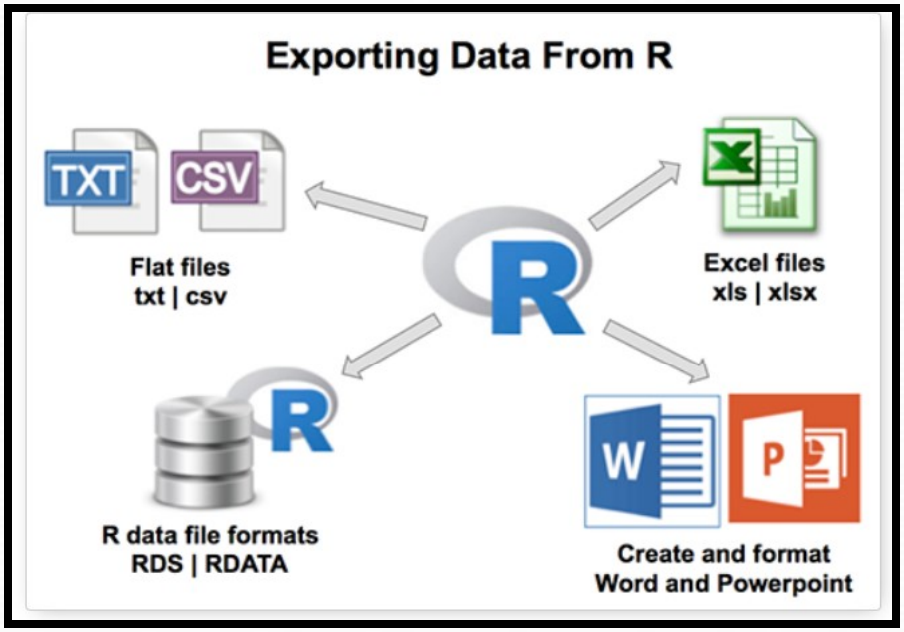
- Data comes in different formats from different sources.

- Suppose we have some data on our computer and we want to import it in R.

- **Different formats of files can be read in R**
  - Comma-separated values (CSV) data file,
  - Table file (TXT),
  - Spreadsheet (e.g. MS Excel) file (.xls or .xlsx),
  - Files from other software like SPSS, Minitab etc.

Experiment 9: Demonstrate importing and exporting data using R





Experiment 9: Importing and Exporting of data in R.

```
SepalLengthCm,SepalWidthCm,PetalLengthCm,PetalWidthCm,Species
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
```

CSV

Experiment 9: Importing and Exporting of data in R

```
SepalLengthCm   SepalWidthCm    PetalLengthCm   PetalWidthCm    Species
5.1 3.5 1.4 0.2 Iris-setosa
4.9 3   1.4 0.2 Iris-setosa
4.7 3.2 1.3 0.2 Iris-setosa
4.6 3.1 1.5 0.2 Iris-setosa
5   3.6 1.4 0.2 Iris-setosa
5.4 3.9 1.7 0.4 Iris-setosa
4.6 3.4 1.4 0.3 Iris-setosa
5   3.4 1.5 0.2 Iris-setosa
```

TAB Delimited

Experiment 9: Importing and Exporting of data in R

SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
5.4	3.7	1.5	0.2	Iris-setosa
4.8	3.4	1.6	0.2	Iris-setosa
4.8	3	1.4	0.1	Iris-setosa
4.3	3	1.1	0.1	Iris-setosa
5.8	4	1.2	0.2	Iris-setosa
5.7	4.4	1.5	0.4	Iris-setosa
5.4	3.9	1.3	0.4	Iris-setosa
5.1	3.5	1.4	0.3	Iris-setosa

XLSX

Experiment 9: Importing and Exporting of data in R

- **Things to be noted before importing data from a file:**
  - Use the first row as column headers (or column names). Generally, columns represent variables.
  - Avoid names with blank spaces. **Good column names: Long_jump or Long.jump**. Bad column name: Long jump.
  - Avoid names with special symbols: ?, $, *, +, #, (,), -, /, }, {, |, >, < etc. Only underscore can be used.
  - Avoid beginning variable names with a number. Use letter instead.

Experiment 9: Importing and Exporting of data in R

- The best way to read data from a CSV file1 is to use **read.table**.

- Note that, depending on the format of your file, several variants of read.table() are available, including **read.csv(), read.csv2(), read.delim() and read.delim2().**

- The difference between **read.csv and read.csv2** is the default field separator, as "," and ";" respectively.

- The **read. delim** function is used when numbers in your file use periods as decimals. The **read. delim2** function is used when numbers in your file use commas as decimals.

Experiment 9: Demonstrate importing and exporting data using R

Importing Data Files / Reading a Comma-separated values(CSV) file:

The contents of a CSV file can be read as a data frame in R using the **read.csv(…) function**. The CSV file to be read should be either present in the current working directory or the directory should be set accordingly using the **setwd(…) command** in R. The CSV file can also be read from a **URL** using **read.csv()** function.

Experiment 9: Demonstrate importing and exporting data using R

Importing Data Files: Comma-separated values (CSV) data file:

Syntax to read the CSV file:

```
read.csv(file, header = TRUE, sep = ",", dec = ".", ...)
```

- **file:** file name
- **header:** first line as header or not, logical
- **sep:** field separator
- **dec:** The Character used in the file for decimal points.

Experiment 9: Demonstrate importing and exporting data using R

Importing Data Files: Comma-separated values (CSV) data file:

- First set the working directory where the CSV file is located.

setwd("<location of your dataset>")

>setwd("C:/RExamples")

	A	B	C	D	E
1	1	10	100	1000	
2	2	20	200	2000	
3	3	30	300	3000	
4	4	40	400	4000	
5					

Example 1:

```
> setwd("C:/RExamples")
> d=read.csv("example.csv")
> d
```

```
> setwd("C:/RExamples")
> d=read.csv("example.csv")
> d
  X1 X10 X100 X1000
1  2  20  200  2000
2  3  30  300  3000
3  4  40  400  4000
>
```

Experiment 9: Demonstrate importing and exporting data using R

Importing Data Files: Comma-separated values (CSV) data file:

- Data files have many formats and accordingly we have options for loading them.

- If the data file **does not have headers** in the first row, then use

d=read.csv("example.csv",header=FALSE)

Example 1:

```
> d=read.csv("example.csv",header=FALSE)
> d
```

```
> d=read.csv("example.csv",header=FALSE)
> d
  V1 V2  V3   V4
1  1 10 100 1000
2  2 20 200 2000
3  3 30 300 3000
4  4 40 400 4000
>
```

	A	B	C	D	E
1	1	10	100	1000	
2	2	20	200	2000	
3	3	30	300	3000	
4	4	40	400	4000	
5					

Experiment 9: Demonstrate importing and exporting data using R

Importing Data Files: Comma-separated values (CSV) data file:

Example 1:

```
> d=read.csv("example.csv",header=TRUE)
> d
```

```
> d=read.csv("example.csv",header=TRUE)
> d
  X1 X10 X100 X1000
1  2  20  200  2000
2  3  30  300  3000
3  4  40  400  4000
>
```

	A	B	C	D	E
1	1	10	100	1000	
2	2	20	200	2000	
3	3	30	300	3000	
4	4	40	400	4000	
5					

Experiment 9: Demonstrate importing and exporting data using R

Importing Data Files: Comma-separated values (CSV) data file:

In order to change the default header name we have to use the following syntax: (or) to rename the header names manually:

>names(d)=c(“column1”,”column2”,”column3”,”column4”)

```
> d
  V1 V2  V3   V4
1  1 10 100 1000
2  2 20 200 2000
3  3 30 300 3000
4  4 40 400 4000
> names(d)=c("column1","column2","column3","column4")
> d
  column1 column2 column3 column4
1       1      10     100    1000
2       2      20     200    2000
3       3      30     300    3000
4       4      40     400    4000
>
```

Experiment 9: Demonstrate importing and exporting data using R

Importing Data Files: Comma-separated values (CSV) data file:

Example 2: Input as CSV File

1. Reading CSV file:

```
>emp=read.csv("emp.csv")
>print(emp)
```

2. Analyzing the CSV File:

```
print(is.data.frame(emp))
print(ncol(emp))
print(nrow(emp))
```

```
[1] TRUE
[1] 5
[1] 7
```

Experiment 9: Demonstrate importing and exporting data using R

Importing Data Files: **Comma-separated values (CSV) data file:**

Example 2(Continues):

Get the maximum salary

```
>sal=max(emp$salary)
>print(sal)
```

OUTPUT:

```
[1] 69040
```

Get the details of the person with max salary

```
Msal=subset(emp,salary== max(salary))
print(Msal)
```

OUTPUT:

	id	name	department	salary	projects
3	3	C	Marketing	69040	8

Experiment 9: Demonstrate importing and exporting data using R

Importing Data Files: **Comma-separated values (CSV) data file:**

Example 2(Continues):

Get all the people working in IT department

```
emp=read.csv("emp.csv")
ID=subset(emp,department=="IT")
print(ID)
```

Get the persons in IT department whose salary is greater than 61000

```
emp=read.csv("emp.csv")
ID=subset(emp,department=="IT" & salary>60000)
print(ID)
```

Experiment 9: Demonstrate importing and exporting data using R

Importing Data Files: Comma-separated values (CSV) data file:

- We can set the delimiter with **sep.**
- If it is tab delimited, use **sep="\t".**

```
d=read.csv("example.csv",sep="\t")
```

- If it is space-delimited, use **sep=" ".**

```
d=read.csv("example.csv",sep=" ")
```

Experiment 9: Demonstrate importing and exporting data using R

Importing Data Files: Reading Tabular Data Files:

- Tabular data files are text files with a simple format:
  - Each line contains one record.
  - Within each record, fields(items) are separated by a one character delimiter, such as a space, tab, colon, or comma.
  - Each record contains the same number of fields.

Syntax to read a text file that contains a tale of data:

```
read.table(file, header = FALSE, sep = "", dec = ".")
```

Experiment 9: Demonstrate importing and exporting data using R

Importing Data Files: using website/url:

- One can also read or upload the file from Internet site.

- We can import data from a website using the read.table, read.csv, read.delim

  http://www.jaredlander.com/data/TomatoFirst.csv

as follows

```
>t=read.table(file="http://www.jaredlander.com/data/TomatoFirst.csv",header=TRUE,sep=",")
> t
```

Experiment 9: Demonstrate importing and exporting data using R

Importing Data Files: Spreadsheet(Excel) file data:

- The **xlsx** package has the function **read.xlsx()** for reading Excel files.
- This will read the first sheet of an Excel Spreadsheet.
- To read the Excel files, we first need to install the package.

```
install.packages("xlsx")    # install the package
library(xlsx)               # Import the package
d=read.xlsx("data.xlsx",sheetIndex or sheetName)
```

Experiment 9: Demonstrate importing and exporting data using R

Importing Data Files: Spreadsheet(Excel) file data:

- To load other sheets with **read.xlsx(),** we specify a number for **sheetIndex** or a name for **sheetName**

d=read.xlsx("data.xlsx",sheetIndex =2)

(or)

d=read.xlsx("data.xlsx",sheetName ="marks")

Experiment 9: Demonstrate importing and exporting data using R

Importing Data Files: Spreadsheet(Excel) file data:

- For reading older Excel files in **.xls** format, use **gdata** package and the function **read.xls()**.
- This will read the first sheet of an Excel Spreadsheet.
- To read the Excel files, we first need to install the package.

```
install.packages("gdata")
library(gdata)
```

Experiment 9: Demonstrate importing and exporting data using R

Importing Data Files: Spreadsheet(Excel) file data:

- To load other sheets with **read.xls(),** we specify a number for **sheetIndex** or a name for **sheetName**

d=read.xls(“data.xls”,sheetIndex =2)

(or)

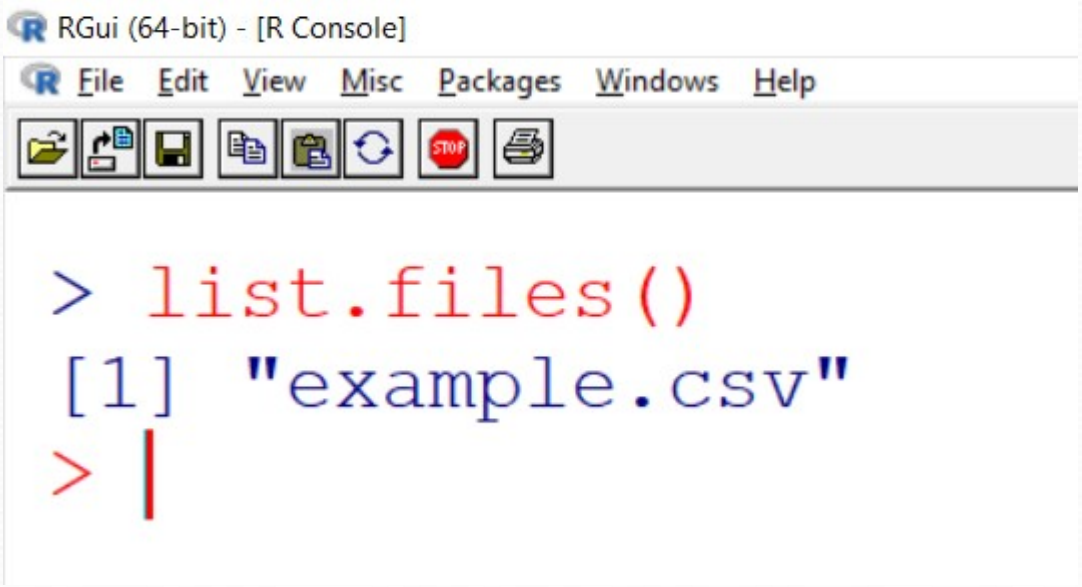
d=read.xls(“data.xls”,sheetName =“marks”)

Experiment 9: Demonstrate importing and exporting data using R

Contents of working directory:

- The list.files function shows the contents of your current working directory.

>list.files()



Experiment 9: Demonstrate importing and exporting data using R

Exporting Data / Writing into a CSV file:

The contents of the data frame can be written into a CSV file. The CSV file is stored in the current working directory with the name specified in the function **write.csv**(data frame, output CSV name) in R.

Experiment 9: Demonstrate importing and exporting data using R

Exporting Data:

- We can use any of the following methods to write data into a file:
  - write.table()
  - write.csv()
  - write_delim()
  - write.xlsx()

Experiment 9: Demonstrate importing and exporting data using R

Exporting Data:

write.table():

- It can be used to export a data frame or a matrix to a text file.

Syntax: write.table(x, file, append = FALSE, sep = " ",
dec = ".", row.names = TRUE, col.names = TRUE)

Parameters:

x: a matrix or a data frame to be written.
file: a character specifying the name of the result file.
sep: the field separator string, e.g., sep = "\t" (for tab-separated value).
dec: the string to be used as decimal separator. Default is "."
row.names: either a logical value indicating whether the row names of x are to be written along with x, or a character vector of row names to be written.
col.names: either a logical value indicating whether the column names of x are to be written along with x, or a character vector of column names to be written.

Experiment 9: Demonstrate importing and exporting data using R

Exporting Data: write.table():

 # R program to illustrate Exporting data from R

```
# Creating a dataframe
df=data.frame(
  "Name" = c("Amiya", "Raj", "Asish"),
  "Language" = c("R", "Python", "Java"),
  "Age" = c(22, 25, 45)
)

# Export a data frame to a text file using write.table()
write.table(df,
            file = "myDataFrame.txt",
            sep = "\t",
            row.names = TRUE,
            col.names = NA)
```

Experiment 9: Demonstrate importing and exporting data using R

Exporting Data: write.csv():

 # Writing into a CSV file

The contents of the data frame can be written into a CSV file. The CSV file is stored in the current working directory with the name specified in the function write.csv(data frame, output CSV name) in R.

Example:

```
csv_data=read.csv('emp.csv')
new_csv=subset(csv_data,department=="IT" & projects<6)
write.csv(new_csv,"new_sample.csv")
new_data=read.csv(file ='new_sample.csv')
print(new_data)
```