R Basics Continued…

MATRICES:

Matrices are the R objects where the elements are arranged in two dimensional formats. It contains rows and columns. The elements are of same type. Matrices are vectors with "*dimension*" attribute. The dimension attribute itself is a vector of length 2 (number of rows and number of columns).

Creating the matrix:

1.Matrix function:

A matrix can be created by "matrix" function as follows:

```
## It enters the data column wise i.e column is filled first and then row by default.
>m<-matrix(1:6,2,3)
> m
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

OR

```
>m<-matrix(1:6,nrow=2,ncol=3,byrow=FALSE) ## Enters column wise.
> m
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> m<-matrix(1:6,nrow=2,ncol=3,byrow=TRUE)  ## Enters row wise. Rows are filled first.
> m
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6

>d<-c(2,5,6)
> e<-c(9,5,5)
> m<-matrix(c(d,e),2,3)
> m
     [,1] [,2] [,3]
[1,]    2    6    5
[2,]    5    9    5
```

2.**Dimension Attribute**:

Matrix can be created by adding dimension attribute as follows:

```
>d<-1:8
> dim(d)<-c(2,4)  ## Considers rows as 2 and columns as 4.
```

```
> print(d)
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
> dim(d)  ## It retrieves the assigned values of dimension.
[1] 2 4
```

3.Cbind and rbind function:

Matrix can be created using cbind and rbind function as follows:

a.**cbind**: The elements are placed columnwise.

```
>v<-c(3,5,6,6,7)
> w<-c(4,7,8,9,0)
> m<-cbind(v,w)
> m
     v w
[1,] 3 4
[2,] 5 7
[3,] 6 8
[4,] 6 9
[5,] 7 0
```

b.**rbind** : The elements are placed row wise.

```
>m<-rbind(v,w)
> print(m)
  [,1] [,2] [,3] [,4] [,5]
v    3    5    6    6    7
w    4    7    8    9    0
```

2 Accessing the Matrix:

For a given matrices any one element or any one complete row or column can be extracted as follows:

```
>m<-matrix(1:16,4,4)
> m
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
> z<-m[4,3]   ## Gives 4th row 3rd column element
> z
```

```
[1] 12
> s<-m[,3]  ## Gives all rows and 3rd column element
> s
[1]  9 10 11 12
> q<-m[1,]   ## Gives 1st row and all column elements.
> q
[1]  1  5  9 13
```

3.Operations with Matrices:

```
>a<-matrix(1:4,2,2) ## declaring the matrix a
> a
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> b<-matrix(5:8,2,2) ## declaring the matrix b
> b
     [,1] [,2]
[1,]    5    7
[2,]    6    8
> c<-a+b ## adding two matrices
> c
     [,1] [,2]
[1,]    6   10
[2,]    8   12
>d<-a-b  ## subtracting two matrices
> d
     [,1] [,2]
[1,]   -4   -4
[2,]   -4   -4
>e<-a*b ## Element by element multiplication
> e
     [,1] [,2]
[1,]    5   21
[2,]   12   32
>f<-a%*%b ## Matrix multiplication
> f
     [,1] [,2]
[1,]   23   31
[2,]   34   46

>t(f) ## Gives transpose of matrix
```

```
   [,1] [,2]
[1,]   23   34
[2,]   31   46
> solve(a)  ## Gives inverse of matrix.
   [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
```

ARRAYS:

Arrays are the R data objects which can store data in more than two dimensions.

Creating Arrays:

If we create an array of dimension (3, 3, 2) then it creates 2 rectangular matrices each with 3 rows and 3 columns. An array is created using the array() function. It takes vectors as input and uses the values in the dim parameter to create an array.

```
> v1<-c(1,3,5)
> v2<-c(2,4)
> v3<-c(9,10,11,19)
> a1<-array(c(v1,v2,v3),dim=c(3,3,2))
> a1
, , 1

     [,1] [,2] [,3]
[1,]    1    2   10
[2,]    3    4   11
[3,]    5    9   19

, , 2

     [,1] [,2] [,3]
[1,]    1    2   10
[2,]    3    4   11
[3,]    5    9   19
```

Accessing Array:

Like matrix you can access either single element or complete row or complete column of any of the matrices. For example:

>d<-a1[2,3,1] ## It gives 2nd row and third column element of 1st matrix

[1] 11

> f<-a1[2,2,] ## It gives 2nd row 2nd column element of both the matrices.

> h<-a1[2,,] ## It gives 2nd row all columns and all matrix data.

Array Operations:

Different arithmetic operations can be done on elements of arrays in the similar fashion as of matrices.

Exercise:

1. Create the matrix A and B with values as shown below:

A= $\begin{pmatrix} 5 & 2 & 6 & 1 \\ 0 & 6 & 2 & 0 \\ 3 & 8 & 1 & 4 \\ 1 & 8 & 5 & 6 \end{pmatrix}$ and B= $\begin{pmatrix} 7 & 5 & 8 & 0 \\ 1 & 8 & 2 & 6 \\ 9 & 4 & 3 & 8 \\ 5 & 3 & 7 & 9 \end{pmatrix}$

```
b<-c(7,5,8,0,1,8,2,6,9,4,3,8,5,3,7,9)
c<-matrix(b,byrow=TRUE,4,4)
```

Use a) function matrix b) function cbind c)function rbind. Perform the following task:

i. What is the largest number present in the matrix A and smallest number in matrix B.
ii. Extract the 2nd row and 3rd column element of matrix A and save it in variable c.
iii. Extract row number 4 of matrix B and save it in vector D.
iv. Which is the largest number present in the last column of matrix B?
v. Display the transpose of matrix A and inverse of matrix B.

Data Frame:

It is a tabular form structure. Each column of the table should have same number of elements and each column represents values of one variable. The elements of different columns can be of different objects (unlike matrices).

Creating Data Frame:

Consider the following data from website ESPN cricinfo live

Name	Matches	Innings	Highestscore	average
Tendulkar	200	329	248	53.78
Ponting	168	287	257	51.85
Kallis	166	280	224	55.37
Dravid	164	286	270	52.31
Cook	161	291	294	45.35

The data frame for batsmen with most runs can be created as follows:

```
> match_stat<-
data.frame(name=c("Tendulkar","Ponting","kallis","Dravid","cook"),matches=c(200,168,166,164
,161),innings=c(329,287,280,286,291),highestscore=c(248,257,224,270,294),avg=c(53.78,51.85,
55.37,52.31,45.35))
```

```
> match_stat
       name matches innings highestscore   avg
1 Tendulkar     200     329          248 53.78
2   Ponting     168     287          257 51.85
3    kallis     166     280          224 55.37
4    Dravid     164     286          270 52.31
5      cook     161     291          294 45.35
```

Getting structure of data frame:

The structure of data frame created can be obtained by function str() as follows:

```
>str(match_stat)
```

'data.frame':5 obs. of 5 variables:

$ name : Factor w/ 5 levels "cook","Dravid",..: 5 4 3 2 1

$ matches : num 200 168 166 164 161

$ innings : num 329 287 280 286 291

$ highestscore: num 248 257 224 270 294

$ avg : num 53.8 51.9 55.4 52.3 45.4

Getting summary of data in data frame:

The summary of data in data frame cab be obtained by function summary().

```
>summary(match_stat)
       name      matches         innings       highestscore        avg       
 cook     :1   Min.   :161.0   Min.   :280.0   Min.   :224.0   Min.   :45.35  
 Dravid   :1   1st Qu.:164.0   1st Qu.:286.0   1st Qu.:248.0   1st Qu.:51.85  
 kallis   :1   Median :166.0   Median :287.0   Median :257.0   Median :52.31  
 Ponting  :1   Mean   :171.8   Mean   :294.6   Mean   :258.6   Mean   :51.73  
 Tendulkar:1   3rd Qu.:168.0   3rd Qu.:291.0   3rd Qu.:270.0   3rd Qu.:53.78  
               Max.   :200.0   Max.   :329.0   Max.   :294.0   Max.   :55.37  
```

Accessing data from data frame:

The data from the data frame can be accessed as follows:

i.To get name of the batsman and his corresponding number of innings and average runs.

```
> i<-data.frame(match_stat$name,match_stat$innings,match_stat$avg)
> i
  match_stat.name match_stat.innings match_stat.avg
1       Tendulkar                329          53.78
2         Ponting                287          51.85
3          kallis                280          55.37
4          Dravid                286          52.31
5            cook                291          45.35
```

ii.To find Tendulkar highest score and kallis average. i.e accessing 1st and 3rd row and 4th and 5th column.

```
>res1<-match_stat[c(1,3),c(4,5)]
> res1
  highestscore   avg
1          248 53.78
3          224 55.37
```

Adding New columns:

Any new column can be added in the data frame given as below. Let we want to add number of 50s and 100s for every player of the data frame. We use '$' operator to introduce

```
>match_stat$half_cent<-c(68,62,58,63,57)
> match_stat$cent<-c(51,41,45,36,33)
> match_stat
       name matches innings highestscore   avg half_cent cent
1 Tendulkar     200     329          248 53.78        68   51
2   Ponting     168     287          257 51.85        62   41
3    kallis     166     280          224 55.37        58   45
4    Dravid     164     286          270 52.31        63   36
5      cook     161     291          294 45.35        57   33
```

Adding New rows:

New rows can be added in the existing data frame. Let we want to add two more players Sangakkara and Lara. This can be done by rbind function.

```
>new_match_stat<-
data.frame(name=c("sangakkara","lara"),matches=c(134,131),innings=c(233,232),highestscore=c(319,400),avg=c(57.4,52.8),half_cent=c(52,48),cent=c(38,34))

> match_stat<-rbind(match_stat,new_match_stat)

> match_stat
       name matches innings highestscore   avg half_cent cent
1  Tendulkar     200     329          248 53.78        68   51
2    Ponting     168     287          257 51.85        62   41
3     kallis     166     280          224 55.37        58   45
4     Dravid     164     286          270 52.31        63   36
5       cook     161     291          294 45.35        57   33
6 sangakkara     134     233          319 57.40        52   38
7       lara     131     232          400 52.80        48   34
```

Data frame has too many rows and columns. You can display few starting or ending entries by function head and tail as follows:

```
>head(match_stat,n=2)
name matches innings highestscore   avg
1 Tendulakar     200     329          248 53.78
2    Ponting     168     287          257 51.85
>tail(match_stat,n=3)
name matches innings highestscore   avg
3 kallis     166     280          224 55.37
4 Dravid     164     286          270 52.31
5   cook     161     291          294 45.35
```

Exercise: Refer the table given above with five players.

1. What is the highest score of Tendulkar?
2. Display the name and the average of the player who is having maximum highestscore.
3. Display the name,matches,innings and average of the players having score above 250.
4. Find the row number of the data for which the highestscore is equal or greater than 270
5. Modify Tendulkar's number of matches as 201.

Operators :

1.Arithmetic

	Operator	Description
Arithmetic	+	Adds two vectors
	-	Subtracts two vectors
	*	Multiplies two vectors
	/	Divides first number by second
	%%	Gives remainder
	%/%	Gives quotient
	^	Gives raised to the power.

For example:

```
>a<-c(2.4,3,5)
> b<-c(1.2,3,4.5)
> a+b
[1] 3.6 6.0 9.5
> a-b
[1] 1.2 0.0 0.5
> a/b
[1] 2.000000 1.000000 1.111111
> a*b
[1]  2.88  9.00 22.50
> a%%b
[1] 0.0 0.0 0.5
> a%/%b
[1] 2 1 1
> a^b
[1]    2.859259   27.000000 1397.542486
```

2.Relational Operator

	Operator	Description
Relational	<	Checks if element of first vector is less than corresponding element of second vector.
	>	Checks if element of first vector is greater than corresponding element of second vector.
	==	Checks if element of first vector is equal to corresponding element of second vector.
	<=	Checks if element of first vector is less than or equal to corresponding element of second vector.
	>=	Checks if element of first vector is greater than or equal to corresponding element of second vector.
	!=	Checks if element of first vector is not equal corresponding element of second vector.
	It gives result in terms of Boolean value TRUE or FALSE	

3.Logical operator:

	Operator	Description
Logical	&	Element wise logical AND operator. It ANDs each element of first vector by corresponding element of second vector and gives result as TRUE if both are TRUE
	|	Element wise logical OR operator. It ORs each element of first vector by corresponding element of second vector and gives result as TRUE if either of the element is TRUE
	!	It is called Logical NOT operator. Takes each element of the vector and gives the

		opposite logical value.
	&&	Called Logical AND operator. Takes first element of both the vectors and gives the TRUE only if both are TRUE.
	||	Called Logical OR operator. Takes first element of both the vectors and gives the TRUE if one of them is TRUE.
	It gives result in terms of Boolean value TRUE or FALSE	

4.Assignment operator:
Left Assignment:
<- , = , <<- a<-c(1,2,3)
Right Assignment:
->, ->> c(1,2,3)->a

5.**Miscellaneous operator**:
1. ':' e.g 1:4 [1] 1 2 3 4
2.%in%
It checks whether and element belongs to other vector
e.g:
>v1 <- 8
>v2 <- 12
>t <- 1:10
> print(v1 %in% t)
>print(v2 %in% t)
[1] TRUE
[2] FALSE

Working on the datasets

1.Internal Dataset:
R has internal data sets which can be used for study purpose.
>data() ## It gives the list of internal datasets available.
e.g "women" : Average height and weight of American women.
" Titanic" : Survival of passengers on Titanic.
"mtcars" :Motor trend car road tests.
>help*(data set name)* ## It gives the details about the dataset
e.g
> help("women")

>f1<-women ## it loads the "women" dataset in variable f1

2. External Dataset:

The external data can also be accessed by R as follows:

read.table: This is the generic command which can be used to read comma separated values files or tab delimited values files. The data set on which you want to work should be in the same directory where you are working for the R session. To check the directory where you are working in your R session you can use

>getwd() ## get working directory. It gives you the current working directory

If your data set is present in some different directory then change the directory by

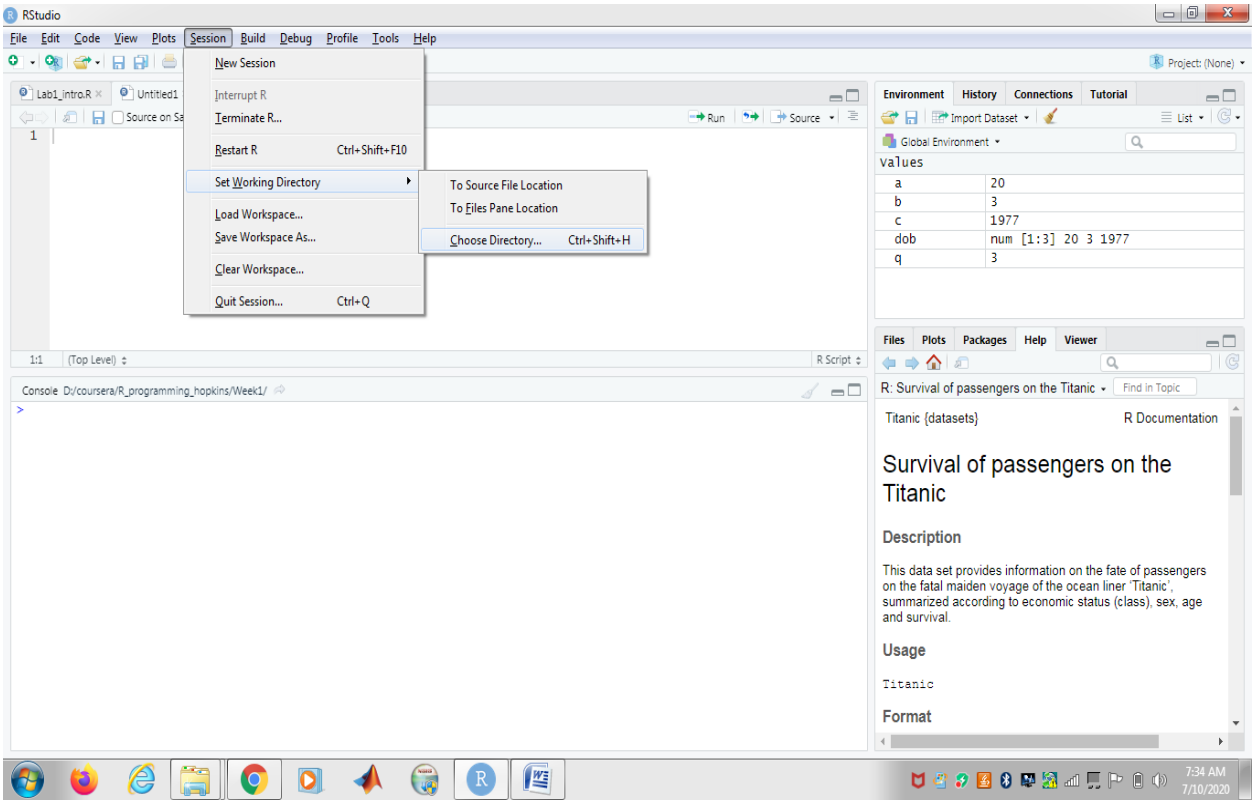
>setwd("path name") command.

e.g:

>setwd("D:/abha/R_programming/course") ## here the directory called as "course contains the dataset. It sets the working directory.

OR

Go to Tab ---Session→ set working directory→choose directory



Once the directory is set you can use the read.table and read.csv command as:

e.g:

>data2<-read.table("hw1_data_Q1_csv.csv",header=TRUE, sep=",") ## "hw1_data_Q1_csv" is the name of the file.

OR

>data2<-read.csv("hw1_data_Q1_csv.csv",header=TRUE) ## The read.csv does not require sep argument Thus read.csv is more specific used for .csv files and read.table is more generic and can be used for .csv as well as .txt files.

Note: If the working directory is not set, R could not find the required dataset. In such cases you can give complete path in read.table command instead of only name.

Following command can also be used to access the dataset:

1.For CSV files:

>data1<-read.csv(file.choose(), header=TRUE)

With this a window will pop up. You can browse you file. Here the file is saved in a object called data1.

OR

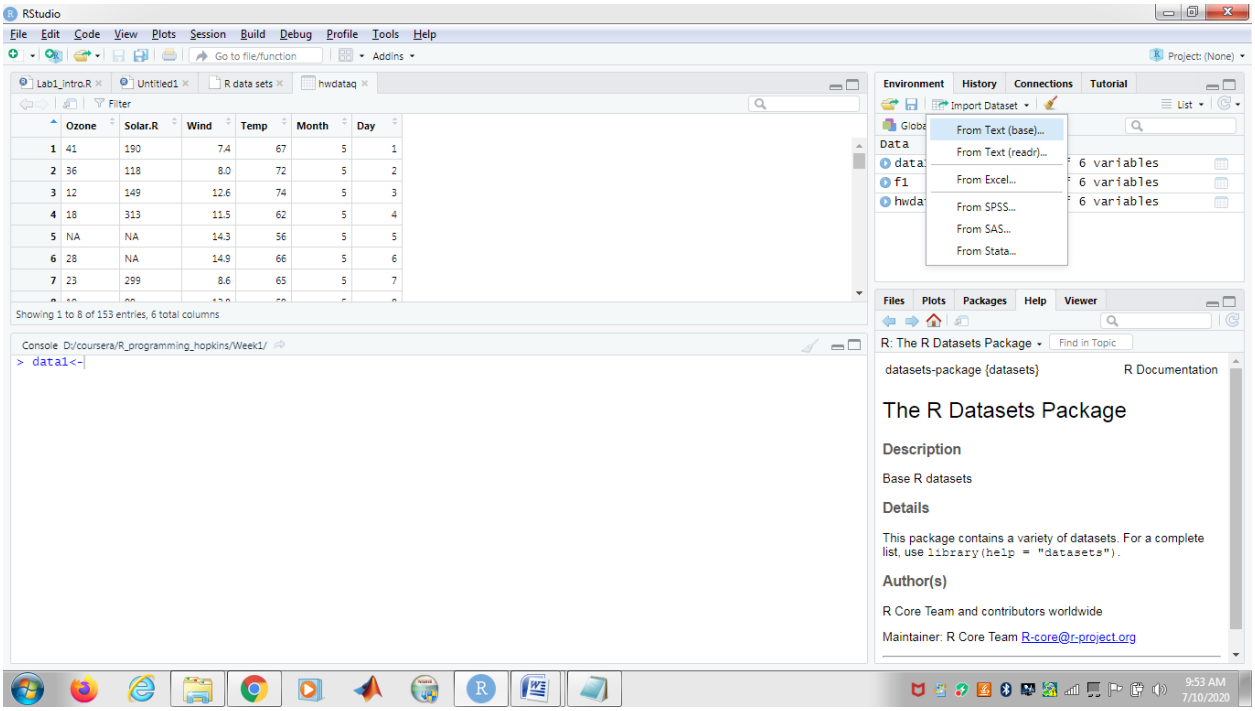
You can use read.table command

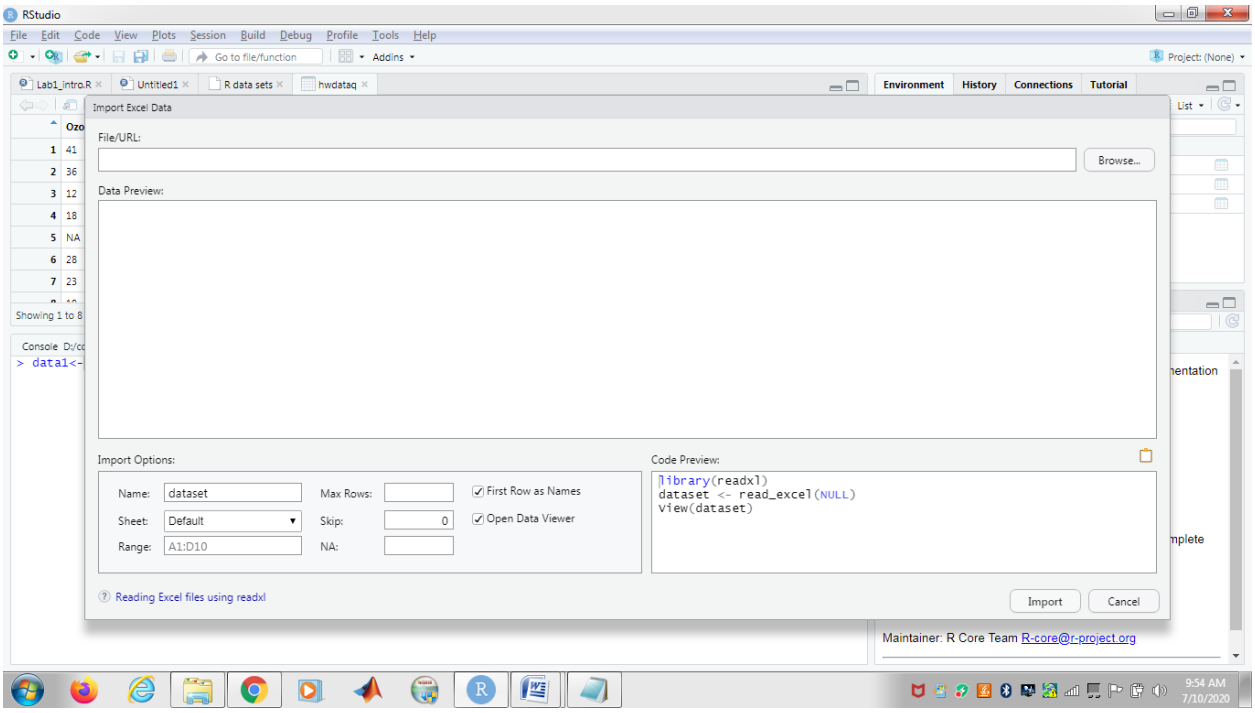
>data1<-read.table(file.choose(),header=TRUE,sep=",")

Note: The read.csv command is specific for csv files hence no need to mention "sep" argument. But read.table is more generic command and hence need to mention the "sep" argument. Here the data is separated by "," comma hence mentioned sep argument as comma.

2.For .xls and .xlsx files:

The excel files can be imported as shown below

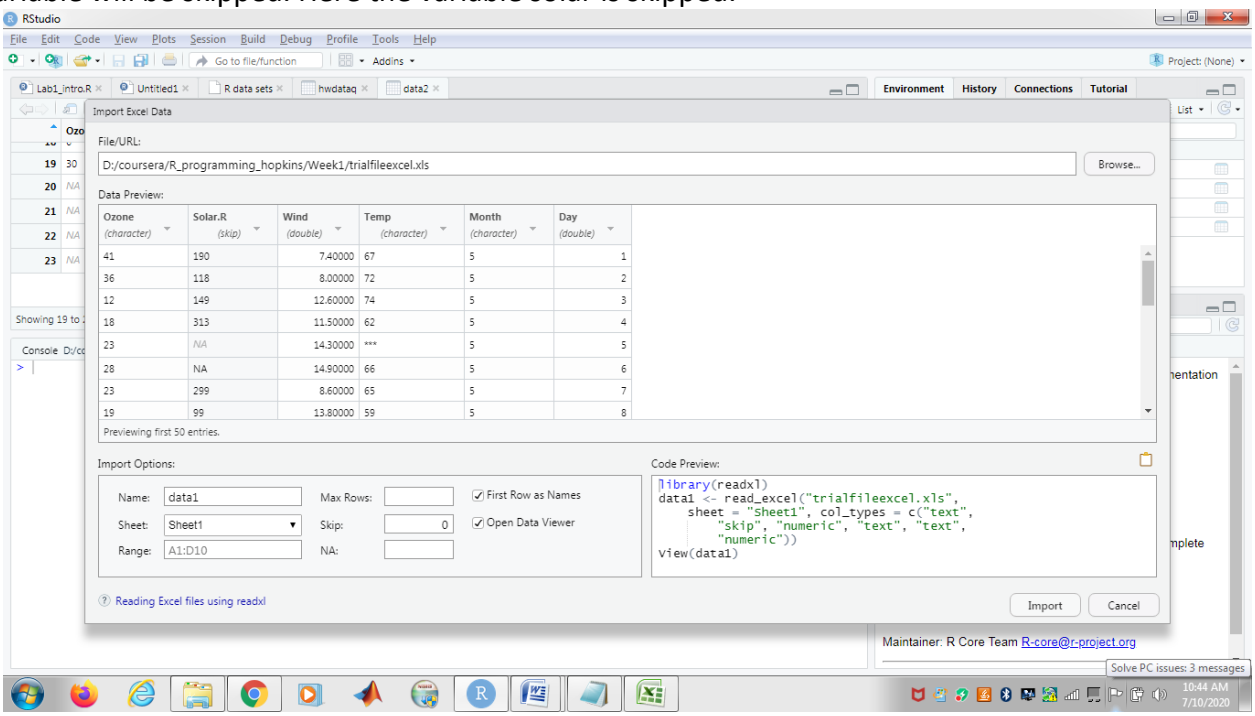




Name: It takes the default name of the object as the file name. You can change this to say data1.

Sheet: It selects the first sheet as default sheet. You can select the other sheet by down arrow.
Range: You can give the range of the data which you want to work on.
Max Rows: You can select how many rows you want to work.
Skip: You can skip the rows
NA: You can specify the values which you want to treat like NA values.

Also you can skip the complete variable name by using down arrow in the Data preview so that the complete variable will be skipped. Here the variable solar is skipped.



Import the data, the code for the corresponding import data will be displayed on the console. You can add more arguments in it.

Exercise: Consider the pollutant data.

1. What is the mean of "Temp" when "Month" is equal to 6?

2How many observations are there in the given data?

3Print last two rows of the data.

4.What is the value of Ozone in 47^{th} row?

5. How many values are missing in Ozone column?
6. What is the mean of Ozone column excluding missing values?

7. Extract the subset of rows of the data frame where Ozone values are above 31 and Temp values are above 90. What is the mean of Solar.R in this subset?

8. What was the maximum ozone value in the month of May (i.e. Month is equal to 5)?

Exercise:

Consider the hair color data. Answer the following questions.

1. How many people have brown eye color?

2. How many people have Blonde hair?

3. How many Brown haired people have Black eyes?

4. What is the percentage of people with Green eyes?

5. What percentage of people have red hair and Blue eyes?