Week-7

(a) Reading different types of data sets (.txt, .csv) from Web or disk and writing in file in specific
disk location.
(b) Reading Excel data sheet in R.
(c) Reading XML dataset in R.

7(a) Reading different types of data sets (.txt, .csv) from Web or disk and writing in file in specific disk location.

Reading data from txt or csv files

The R base function read.table() is a general function that can be used to read a file in table format. The data will be imported as a data frame.
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().

- **read.csv**(): for reading **"comma separated value"** files (".csv").
- **read.csv2**(): variant used in countries that use a comma "," as decimal point and a semicolon ";" as field separators.
- **read.delim**(): for reading *"tab-separated value"* files (".txt"). By default, point (".") is used as decimal points.
- **read.delim2**(): for reading *"tab-separated value"* files (".txt"). By default, comma (",") is used as decimal points.

The simplified format of these functions are, as follows:

```
# Read tabular data into R
read.table(file, header = FALSE, sep = "", dec = ".")
# Read "comma separated value" files (".csv")
read.csv(file, header = TRUE, sep = ",", dec = ".", ...)
# Or use read.csv2: variant used in countries that # use a comma as decimal point and a semicolon as field separator.
read.csv2(file, header = TRUE, sep = ";", dec = ",", ...)
# Read TAB delimited files
read.delim(file, header = TRUE, sep = "\t", dec = ".", ...) read.delim2(file, header = TRUE, sep = "\t", dec = ",", ...)
```

- **file**: the path to the file containing the data to be imported into R.
- **sep**: the field separator character. "\t" is used for tab-delimited file.
- **header**: logical value. If TRUE, **read.table()** assumes that your file has a header row, so row 1 is the name of each column. If that's not the case, you can add the argument **header = FALSE**.
- **dec**: the character used in the file for decimal points.

Reading a local file

To import a local .txt or a .csv file, the syntax would be:

```
# Read a txt file, named "mtcars.txt"
my_data <- read.delim("mtcars.txt")
# Read a csv file, named "mtcars.csv"
my_data <- read.csv("mtcars.csv")
```

Note:
The above R code, assumes that the file "mtcars.txt" or "mtcars.csv" is in your current working directory. To know your current working directory, type the function **getwd**() in R console.

- ❖ It's also possible to choose a file interactively using the function **file.choose**(), which I recommend if you're a beginner in R programming:

```
# Read a txt file
my_data <- read.delim(file.choose())
# Read a csv file
my_data <- read.csv(file.choose())
```

If you use the R code above in RStudio, you will be asked to choose a file.

Reading a file from internet

It's possible to use the functions **read.delim**(), **read.csv**() and **read.table**() to import files from the web.

```
my_data <- read.delim("http://www.sthda.com/upload/boxplot_format.txt")
head(my_data)
```

Here I am using weather data.

Example-1: R program reading a .text file

```
# Read a text file using read.delim()
Data1 = read.delim("weather.txt", header = TRUE)
print(Data1)
```

Output:

```
my_data
   outlook temperature humidity windy play
1  overcast        hot     high FALSE  yes
2  overcast       cool   normal  TRUE  yes
3  overcast       mild     high  TRUE  yes
4  overcast        hot   normal FALSE  yes
5     rainy       mild     high FALSE  yes
6     rainy       cool   normal FALSE  yes
7     rainy       cool   normal  TRUE   no
8     rainy       mild   normal FALSE  yes
9     rainy       mild     high  TRUE   no
10    sunny        hot     high FALSE   no
11    sunny        hot     high  TRUE   no
12    sunny       mild     high FALSE   no
13    sunny       cool   normal FALSE  yes
```

```
14    sunny       mild   normal  TRUE  yes
```

```
Data2 <-read.table("weather.txt", header=TRUE, sep = "\t")
Data2
```

Output:

```
Data2
   outlook temperature humidity windy play
1   overcast        hot       FALSE  yes
2  overcast       cool   normal  TRUE  yes
3  overcast       mild     high  TRUE  yes
4  overcast        hot   normal FALSE  yes
5     rainy       mild     high FALSE  yes
6     rainy       cool   normal FALSE  yes
7     rainy       cool   normal  TRUE   no
8     rainy       mild   normal FALSE  yes
9     rainy       mild     high  TRUE   no
10    sunny        hot     high FALSE   no
11    sunny        hot     high  TRUE   no
12    sunny       mild     high FALSE   no
13    sunny       cool   normal FALSE  yes
14    sunny       mild   normal  TRUE  yes
```

Example-2: R program reading a .csv file

```
Data3 <- read.csv("weather.csv", header=TRUE)
Data3
outlook temperature humidity windy play
1  overcast        hot     high FALSE  yes
2  overcast       cool   normal  TRUE  yes
3  overcast       mild     high  TRUE  yes
4  overcast        hot   normal FALSE  yes
5     rainy       mild     high FALSE  yes
6     rainy       cool   normal FALSE  yes
7     rainy       cool   normal  TRUE   no
8     rainy       mild   normal FALSE  yes
9     rainy       mild     high  TRUE   no
10    sunny        hot     high FALSE   no
11    sunny        hot     high  TRUE   no
12    sunny       mild     high FALSE   no
13    sunny       cool   normal FALSE  yes
14    sunny       mild   normal  TRUE  yes
```

```
Data4 <-read.table("weather.csv", header=TRUE,sep=",")
```

```
Data4
outlook temperature humidity windy play
1  overcast        hot     high FALSE  yes
2  overcast       cool   normal  TRUE  yes
```

```
3  overcast       mild     high  TRUE  yes
4  overcast       hot   normal FALSE  yes
5     rainy       mild     high FALSE  yes
6     rainy       cool   normal FALSE  yes
7     rainy       cool   normal  TRUE   no
8     rainy       mild   normal FALSE  yes
9     rainy       mild     high  TRUE   no
10    sunny        hot     high FALSE   no
11    sunny        hot     high  TRUE   no
12    sunny       mild     high FALSE   no
13    sunny       cool   normal FALSE  yes
14    sunny       mild   normal  TRUE  yes
```

It's also possible to choose a file interactively using the function **file.choose**()

□ To read .txt file

```
data3 <-read.delim(file.choose(), header=TRUE)
data3

data4 <-read.table(file.choose(),header=TRUE, sep="\t")
data4
```

□ To read .csv file

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

data2 <-read.table(file.choose(), header=TRUE,sep=",")
data2
```

Reading a file from internet

It's possible to use the functions **read.delim**(), **read.csv**() and **read.table**() to import files from the web.

```
my_data <- read.delim("http://www.sthda.com/upload/boxplot_format.txt")
head(my_data)
```

Output:

```
        Nom   variable Group
1       IND1       10     A
2        IND2       7     A
3       IND3      20     A
4       IND4      14     A
5        IND5     14     A
6       IND6      12     A
7       IND7      10     A
8       IND8      23     A
9       IND9      17     A
10      IND10     20     A
```

11	IND11	14	A
12	IND12	13	A
13	IND13	11	B
14	IND14	17	B
15	IND15	21	B
16	IND16	11	B
17	IND17	16	B
18	IND18	14	B
19	IND19	17	B
20	IND20	17	B
21	IND21	19	B
22	IND22	21	B
23	IND23	7	B
24	IND24	13	B
25	IND25	0	C
26	IND26	1	C
27	IND27	7	C
28	IND28	2	C
29	IND29	3	C
30	IND30	1	C
31	IND31	2	C
32	IND32	1	C
33	IND33	3	C
34	IND34	0	C
35	IND35	1	C
36	IND36	4	C
37	IND37	3	D
38	IND38	5	D
39	IND39	12	D
40	IND40	6	D
41	IND41	4	D
42	IND42	3	D
43	IND43	5	D
44	IND44	5	D
45	IND45	5	D
46	IND46	5	D
47	IND47	2	D
48	IND48	4	D
49	IND49	3	E
50	IND50	5	E
51	IND51	3	E
52	IND52	5	E
53	IND53	3	E
54	IND54	6	E
55	IND55	1	E
56	IND56	1	E
57	IND57	3	E
58	IND58	2	E
59	IND59	6	E
60	IND60	4	E

61	IND61	11	F
62	IND62	9	F
63	IND63	15	F
64	IND64	22	F
65	IND65	15	F
66	IND66	16	F
67	IND67	13	F
68	IND68	10	F
69	IND69	26	F
70	IND70	26	F
71	IND71	24	F
72	IND72	13	F

Import dataset in R programming

R is a programming language designed for data analysis. Therefore, loading data is one of the core features of R.

R contains a set of functions that can be used to load data sets into memory. You can also load data into memory using R Studio - via the menu items and toolbars.

Data Formats

R can load data in two different formats:

- CSV files
- Text files

CSV means Comma Separated Values. You can export CSV files from many data carrying applications. For instance, you can export CSV files from data in an Excel spreadsheet. Here is an example of how a CSV file looks like inside:

```
name,id,salary
"John Doe",1,99999.00
"Joe Blocks",2,120000.00
"Cindy Loo",3,150000.00
```

As you can see, the values on each line are separated by commas. The first line contains a list of column names. These column names tell what the data in the following lines mean. These names only make sense to you. R does not care about these names. R just uses these name to identify data from the different columns.

A text file is typically similar to a CSV file, but instead of using commas as separators between values, text files often use other characters, like e.g. a Tab character. Here is an example of how a text file could look inside:

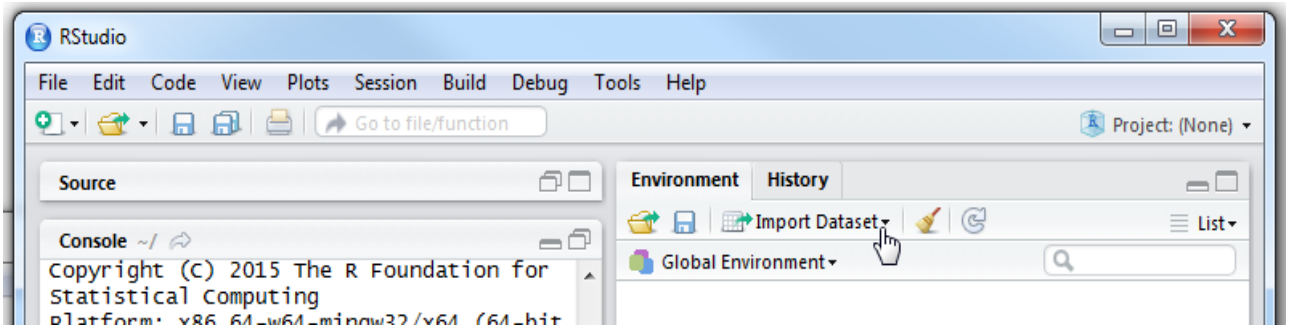
```
name            id      salary

"John Doe"      1       99999.00
"Joe Blocks"    2       120000.00
"Cindy Loo"     3       150000.00
```

As you can see, the data might be easier to read in text format - if you look at the data directly in the data file that is. Once the data is loaded into R / R studio, there is no difference. You can look at the data in R Studio's tabular data set viewer, and then you cannot see the difference between CSV files and text files.

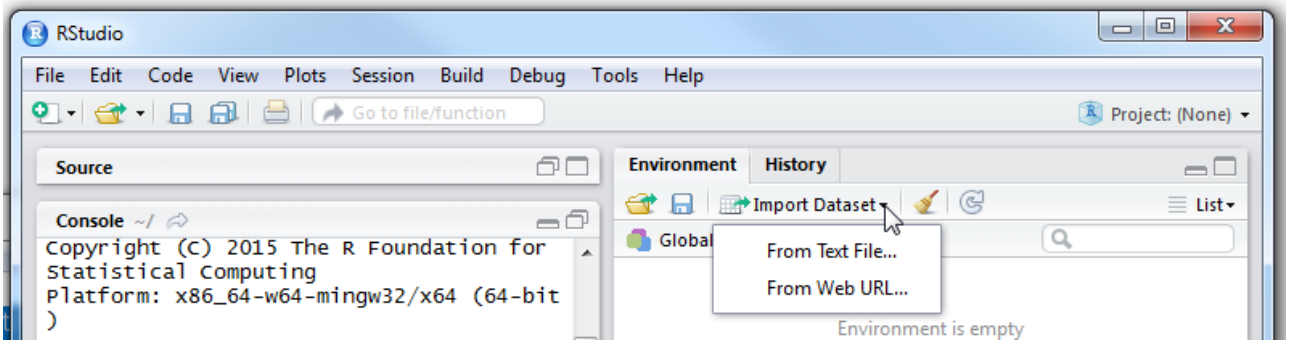
Actually, the name "text files" is a bit confusing. Both CSV files and text files contains data in textual form (as characters). One just uses commas as separator between the values, whereas the others use a tab character.

Load Data Via R Studio Menu Items

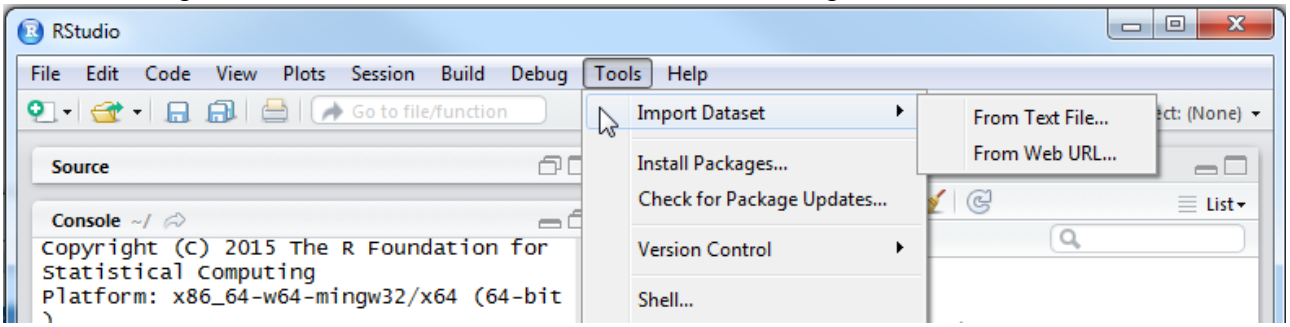
The easiest way to load data into memory in R is by using the R Studio menu items. R Studio has menu items for loading data in two different places. The first is in the toolbar of the upper right section of R Studio. This screenshot shows where the "Import Dataset" button is (look for the little mouse pointer "hand") :



When you click the button you get this little menu:



You can also import data from the top menu of R Studio. The next screenshot shows where the "Import Dataset" menu item is located in R Studio's top menu:



Text File or Web URL

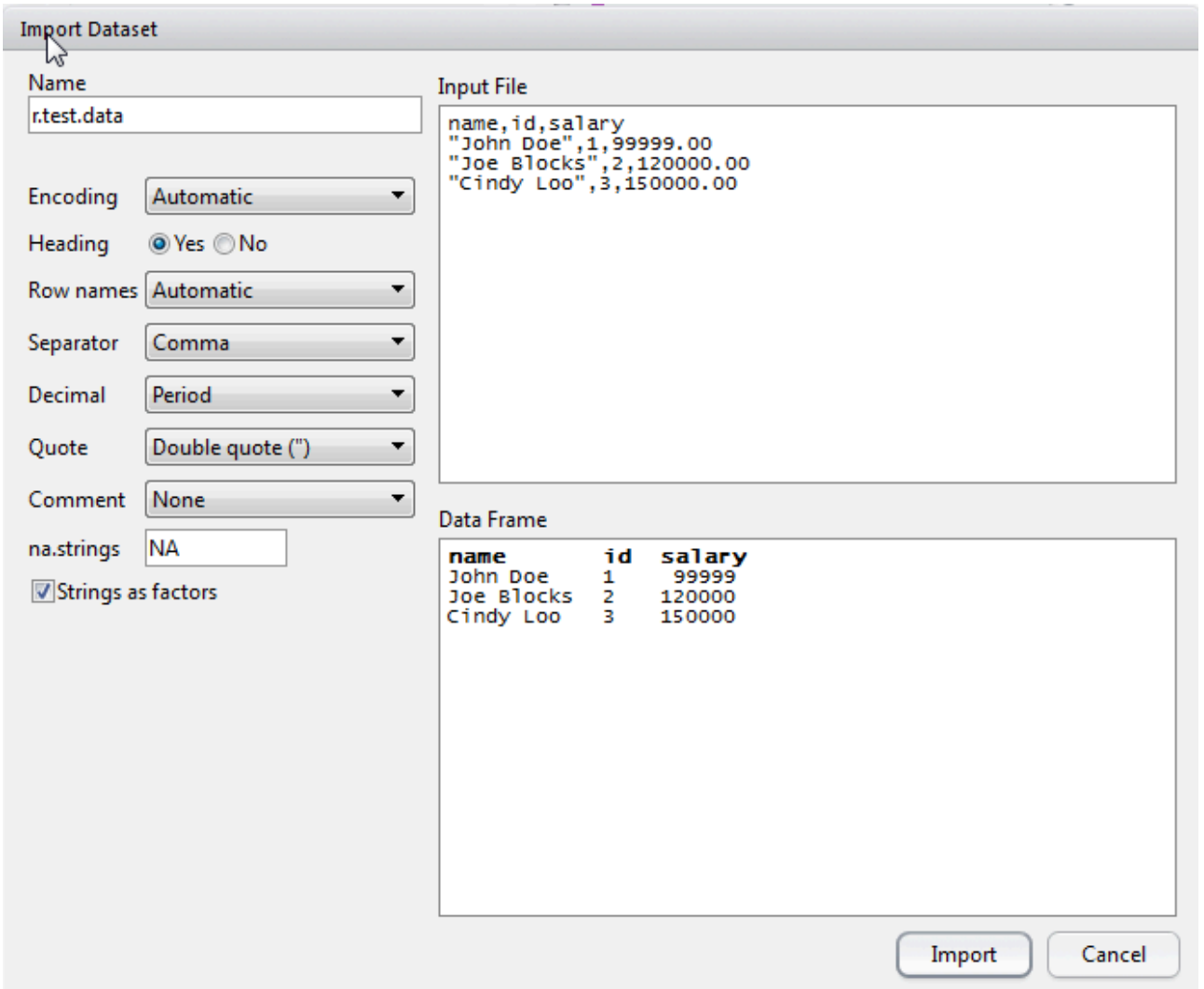
As you can see in both the "Import Dataset" menu items, you can import a data set "From Text File" or "From Web URL". These two options refer to where you load the data from. "From Text File" means from a text file on your local computer. "From Web URL" means that you load the data from a web server somewhere on the internet.

Regardless of whether you choose "From Text File" or "From Web URL", R can load the file as either a CSV or text file. The location of the file has nothing to do with the data format used inside the file. Don't get confused by that. The menu item "From Text file" does not mean "text file format" (tab characters as separators). It just means

"a file on your local computer". "From Local File" would probably have been a more informative text for this menu item.

Selecting Data Format

After you have chosen the location to load the file from, you will be shown a dialog like this:



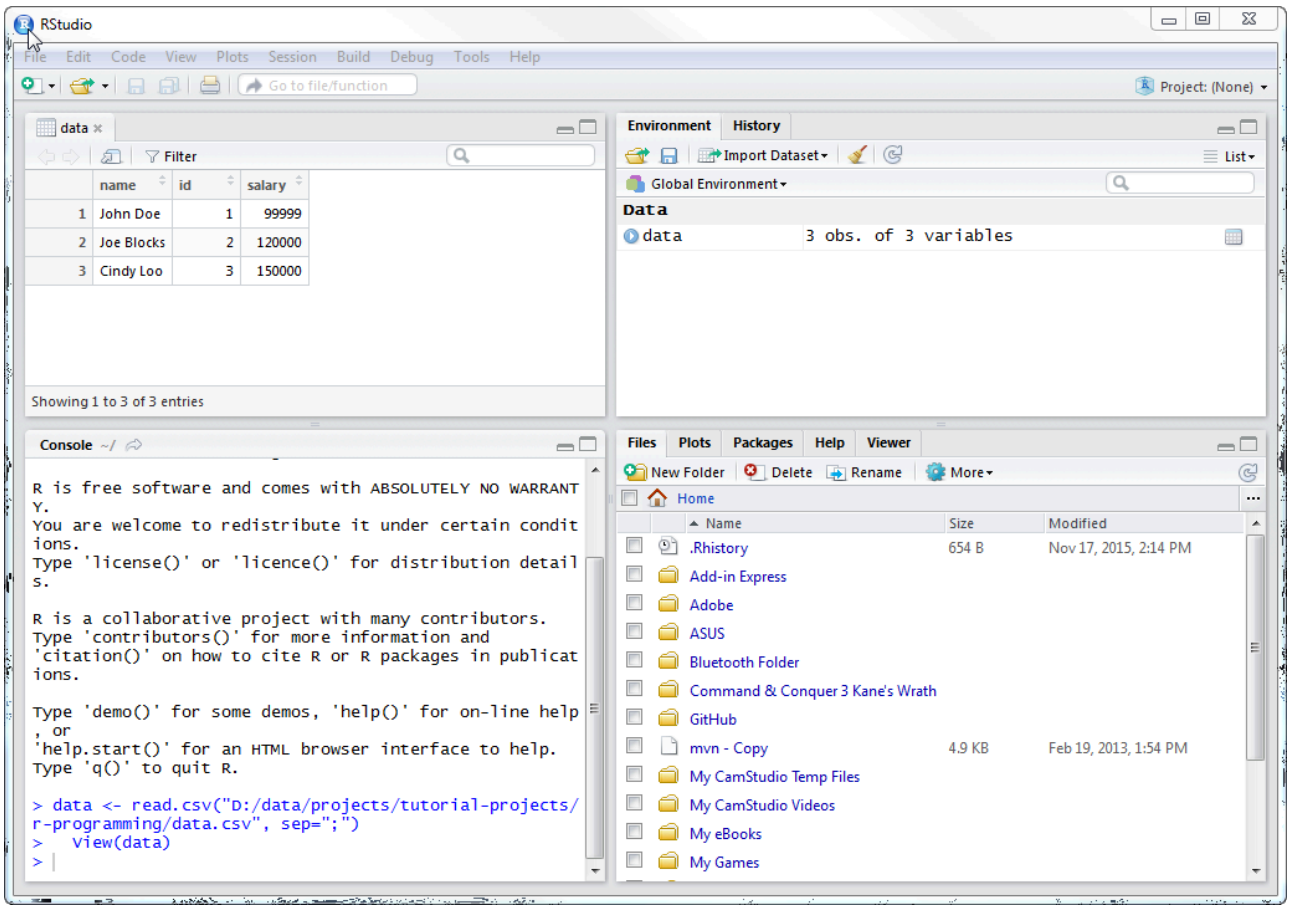
The select boxes (drop down boxes) allows you to specify different configurations about the data format of the file you are about to import. In the boxes on the right you can see two boxes. The top box shows you what the data file looks like. The bottom box shows you how R Studio interprets the data in the file based on the configurations chosen in the select boxes in the left side of the dialog. If you change the choices in the select boxes you will see that the bottom right box changes.

When you have selected all the configurations you need in the select boxes on the left, click the "Import" button. The data will now be loaded into R Studio.

Note that R Studio prints the R commands needed to load the data into the R console in the left side of R studio. You can copy these functions and use them to load data into R via R code.

After the Data is Loaded

After you have loaded the data into R Studio it will look similar to the screenshot below:



7(b) Reading Excel data sheet in R.
Steps to Import an Excel file into R

Step 1: Install the readxl package
In the R Console, type the following command to install the **readxl** package:
install.packages("readxl")

Step 2: Prepare your Excel File
Let's suppose that you have an Excel file with some data about products:

Product	Price
Refrigerator	1200
Oven	750
Dishwasher	900
Coffee Maker	300

And let's say that the Excel file name is **product_list**, and your goal is to import that file into R.
Step 3: Import the Excel file into R

In order to import your file, you'll need to apply the following template in the R Editor:

library("readxl")
read.excel("Path where your Excel file is stored\\FileName.xlsx")

Example:

```
my_data <- read_excel("product_list.xlsx")
my_data
```

(OR)

```
my_data <- read_excel(file.choose())
my_data
```

Note:

If you use the R code above in RStudio, you will be asked to choose a file.

Output:

```
# A tibble: 4 x 2
    Product         Price
     <chr>          <dbl>
1   Refrigerator    1200
2   Oven             750
3   Dishwasher       900
4   Coffee Maker     300
```

Importing Excel files using xlsx package

The **xlsx** package, a java-based solution, is one of the powerful R packages to **read**, **write** and **format Excel files**.

Installing and loading xlsx package

- Install

```
install.packages("xlsx")
```

- Load

```
library("xlsx)
```

Using xlsx package

There are two main functions in **xlsx** package for reading both xls and xlsx Excel files: **read.xlsx**() and **read.xlsx2**() [faster on big files compared to read.xlsx function].

The simplified formats are:

```
read.xlsx(file, sheetIndex, header=TRUE)
read.xlsx2(file, sheetIndex, header=TRUE)
```

- file: file path
- sheetIndex: the index of the sheet to be read
- header: a logical value. If TRUE, the first row is used as column names.

Example:

```
library("xlsx")
my_data1 <- read.xlsv(file.choose(), 1)  # read first sheet
```

7(c) Reading XML dataset in R.

In R, we can read the xml files by installing "XML" package into the R environment. This package will be installed with the help of the familiar command i.e., install. packages.

install.packages("XML")

Creating XML File

Save the following data with the .xml file extension to create an xml file. XML tags describe the meaning of data, so that data contained in such tags can easily tell or explain about the data.

Example: xml_data.xml

Example: xml_data.xml

```
<records>
<employee_info>
<id>1</id>
<name>Shubham</name>
<salary>623</salary>
<date>1/1/2012</date>
<dept>IT</dept>
</employee_info>

<employee_info>
<id>2</id>
<name>Nishka</name>
<salary>552</salary>
<date>1/1/2012</date>
<dept>IT</dept>
</employee_info>

<employee_info>
<id>1</id>
<name>Gunjan</name>
<salary>669</salary>
<date>1/1/2012</date>
<dept>IT</dept>
</employee_info>

<employee_info>
<id>1</id>
<name>Sumit</name>
<salary>825</salary>
<date>1/1/2012</date>
<dept>IT</dept>
</employee_info>

<employee_info>
<id>1</id>
```

```
<name>Arpita</name>
<salary>762</salary>
<date>1/1/2012</date>
<dept>IT</dept>
</employee_info>

<employee_info>
<id>1</id>
<name>Vaishali</name>
<salary>882</salary>
<date>1/1/2012</date>
<dept>IT</dept>
</employee_info>

<employee_info>
<id>1</id>
<name>Anisha</name>
<salary>783</salary>
<date>1/1/2012</date>
<dept>IT</dept>
</employee_info>

<employee_info>
<id>1</id>
<name>Ginni</name>
<salary>964</salary>
<date>1/1/2012</date>
<dept>IT</dept>
</employee_info>

</records>
```

Reading XML File

In R, we can easily read an xml file with the help of xmlParse() function. This function is stored as a list in R. To use this function, we first need to load the xml package with the help of the library() function. Apart from the xml package, we also need to load one additional package named methods.

Example: Reading xml data in the form of a list.

```
install.packages("XML")
# Loading the package required to read XML files.
    library("XML")
# Also loading the other required package.
    library("methods")
# Giving the input file name to the function.
    result <- xmlParse(file = "xml_data.xml")
    xml_data <- xmlToList(result)
    print(xml_data)
```

Output:

```
  xml_data <- xmlToList(result)
 print(xml_data)
$employee_info
```

```
$employee_info$id
[1] "1"
$employee_info$name
[1] "Shubham"
$employee_info$salary
[1] "623"
$employee_info$date
[1] "1/1/2012"
$employee_info$dept
[1] "IT"
$employee_info
$employee_info$id
[1] "2"
$employee_info$name
[1] "Nishka"
$employee_info$salary
[1] "552"
$employee_info$date
[1] "1/1/2012"
$employee_info$dept
[1] "IT"
$employee_info
$employee_info$id
[1] "1"
$employee_info$name
[1] "Gunjan"
$employee_info$salary
[1] "669"
$employee_info$date
[1] "1/1/2012"
$employee_info$dept
[1] "IT"
$employee_info
$employee_info$id
[1] "1"

$employee_info$name
[1] "Sumit"
$employee_info$salary
[1] "825"
$employee_info$date
[1] "1/1/2012"
$employee_info$dept
[1] "IT"
$employee_info
$employee_info$id
[1] "1"
$employee_info$name
[1] "Arpita"
$employee_info$salary
```

[1] "762"
$employee_info$date
[1] "1/1/2012"
$employee_info$dept
[1] "IT"
$employee_info
$employee_info$id
[1] "1"
$employee_info$name
[1] "Vaishali"
$employee_info$salary
[1] "882"
$employee_info$date
[1] "1/1/2012"
$employee_info$dept
[1] "IT"
$employee_info
$employee_info$id
[1] "1"
$employee_info$name
[1] "Anisha"
$employee_info$salary
[1] "783"
$employee_info$date
[1] "1/1/2012"
$employee_info$dept
[1] "IT"
$employee_info
$employee_info$id
[1] "1"
$employee_info$name
[1] "Ginni"
$employee_info$salary
[1] "964"
$employee_info$date
[1] "1/1/2012"
$employee_info$dept
[1] "IT"

 # To download file to the current working directory
download.file("https://www.w3schools.com/xml/simple.xml", "breakfast.xml")

 # Install XML package
install.packages("XML")

 # To load library
library(XML)

 # Giving the input file name to the function.

```
doc <- xmlParse("breakfast.xml")
print(doc)
#Converting the data into list
xml_data <-xmlToList(doc)
print(xml_data)
xmldataframe <- xmlToDataFrame("breakfast.xml")
xmldataframe
```

Output:

```
library(XML)
> doc <- xmlParse("breakfast.xml")
> print(doc)
<?xml version="1.0" encoding="UTF-8"?>
<breakfast_menu>
  <food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>Two of our famous Belgian Waffles with plenty of real maple
syrup</description>
    <calories>650</calories>
  </food>
  <food>
    <name>Strawberry Belgian Waffles</name>
    <price>$7.95</price>
    <description>Light Belgian waffles covered with strawberries and whipped
cream</description>
    <calories>900</calories>
  </food>
  <food>
    <name>Berry-Berry Belgian Waffles</name>
    <price>$8.95</price>
    <description>Light Belgian waffles covered with an assortment of fresh berries
and whipped cream</description>
    <calories>900</calories>
  </food>
  <food>
    <name>French Toast</name>
    <price>$4.50</price>
    <description>Thick slices made from our homemade sourdough
bread</description>
    <calories>600</calories>
  </food>
  <food>
    <name>Homestyle Breakfast</name>
    <price>$6.95</price>
    <description>Two eggs, bacon or sausage, toast, and our ever-popular hash
browns</description>
    <calories>950</calories>
  </food>
</breakfast_menu>
```

> xml_data <-xmlToList(doc)
> print(xml_data)
$food
$food$name
[1] "Belgian Waffles"
$food$price
[1] "$5.95"
$food$description
[1] "Two of our famous Belgian Waffles with plenty of real maple syrup"
$food$calories
[1] "650"
$food
$food$name
[1] "Strawberry Belgian Waffles"
$food$price
[1] "$7.95"
$food$description
[1] "Light Belgian waffles covered with strawberries and whipped cream"
$food$calories
[1] "900"
$food
$food$name
[1] "Berry-Berry Belgian Waffles"
$food$price
[1] "$8.95"
$food$description
[1] "Light Belgian waffles covered with an assortment of fresh berries and whipped cream"
$food$calories
[1] "900"
$food
$food$name
[1] "French Toast"
$food$price
[1] "$4.50"
$food$description
[1] "Thick slices made from our homemade sourdough bread"
$food$calories
[1] "600"
$food
$food$name
[1] "Homestyle Breakfast"
$food$price
[1] "$6.95"
$food$description
[1] "Two eggs, bacon or sausage, toast, and our ever-popular hash browns"
$food$calories
[1] "950"