CHAPTER 19

Inputting Data from External Sources

In this chapter, we introduce various ways to input data from external files or data sets. Reading data from external sources are common for researchers and practitioners since manually inputting data is not feasible for a reasonably sized data set, let alone for huge data sets that are the norm in financial analysis. For the counterpart of saving our data or results to an external file or data set, see chapter 32: "Reading and Writing Binary Data in R." A simple way to input data from a text file is to use the **read.table()** function (see below). The structure of an input file with a text format is in the right panel.

```
> x<-read.table("c:/test2.txt")
```

```
20110102 0.001
20110104 0.002
```

The second example is related to the R data format, usually with an extension of **RData**. First, download and save a data set called **ibm.RData** to your PC then upload it by using the **load()** function.

```
# retrieve ibm.Rdata from CD
> load("c:/ibm.Rdata")
```

19.1. Reading Data from a Text File Using read.table()

Assume that we have a text file stored under the root directory of C drive (c:\). The file contains three observations with two variables (see the right panel below). The **header=T** will use the names of the first row of the input file as the column names.

```
> data<-read.table("c:/test.txt",header=T)
> data
var1 var2
1 29161 19761020
2 15763 19841229
3 10093 19830215
```

```
var1 var2
29161 19761020
15763 19841229
10093 19830215
```

In the above codes, the single forward slash (/) can be replaced by a double backward slash (\\).

```
> data<- read.table("c:\\test.txt",header=T)
```

If there are some extralines at the top of an input file, such as a note, we have to skip those lines since they are not part of the data set we plan to input. Usually we call this type of note as a header. Since the **header=T** is referred to as the variable name, we don't use the word *header* to avoid potential confusion.

```
> # skip the first two lines
> x<- read.table("c:\\test2.txt",skip=2)
```

```
File name is test.txt
Generated 11/10/2010
29161 19761020
15763 19841229
10093 19830215
```

19.2. Reading in the First Ten Rows to Explore

Sometimes we have no clue about the new data set. We like to explore the data set before processing it. We might like to input just ten observations. In those cases, we use **nrows=10**.

```
> d<- read.table("c:\\test.txt" ,skip=10,nrows=10)
```

19.3. Adding Column Names Using colnames() and col.names()

If an input data set has no names for any columns or we want to change the existing names, we use the **colnames()** function.

```
> x<-read.table("c:/test.txt ")
> colnames(x)<-c("ID","date")
```

To make our codes more condensed, we put the **col.names()** function in our **read.table()** function.

```
> x<-read.table("c:/test.txt",col.names=c("ID","date"))
```

Note that the **colnames()** and **col.names()** functions are used in different situations. We will explain their usages in other chapters.

19.4 Reading a CSV File

We use the **read.csv()** function to read an input file with a CSV format.

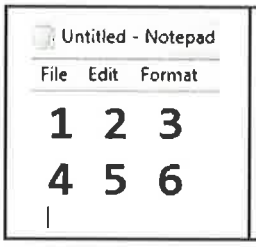
```
> x<-read.csv("c:/hsb.csv") # use default delimiter
```

For a comparison, if we use the **read.table()** function instead, we have to specify the delimiter (sep=",").

```
> x<-read.table("c:/hsb.csv",sep=',') # sep: separator
```

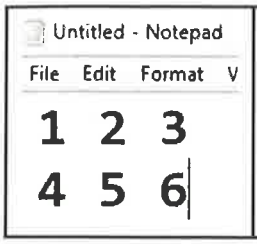
19.5. Reading from a Clipboard

First, we generate our simple input file. After launching Notepad, we type two lines of values then hit the **Enter** key (see below). In particular, the cursor is at the beginning of the third line, an empty line.



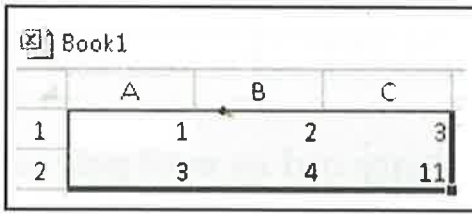

```
> x<-read.table('clipboard')
> x
V1 V2 V3
1 1 2 3
2 4 5 6
```

We highlight and copy the block (i.e., those two lines) then run the above R codes. If our cursor is at the end of the second line (i.e., after the value of 6—see the image below in the right panel), we would get a warning message after we issue the above R codes.




```
> read.table('clipboard')
V1 V2 V3
1 1 2 3
2 4 5 6
Warning message:
In read.table("clipboard"):
incomplete final line found by readTableHeader on 'clipboard'
```

Inputting from the clipboard this way is true for Excel. We can open an Excel file, generate a data set, then highlight and copy it. We can then issue the same R command **x<-read.table('clipboard')**.

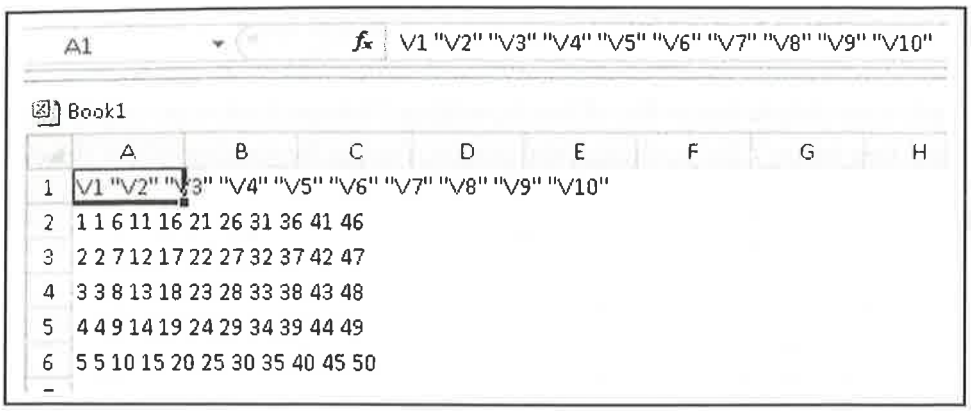



```
> x<-read.table('clipboard')
> x
V1 V2 V3
1 1 2 3
2 3 4 11
```

Copying files to Excel will be more complex. First, we generate a data set called y x, which is a matrix (see the codes below).

```
> x<-1:50
> y<-matrix(x,5,10)
> y
 [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]  1  6 11 16 21 26 31 36 41 46
[2,]  2  7 12 17 22 27 32 37 42 47
[3,]  3  8 13 18 23 28 33 38 43 48
[4,]  4  9 14 19 24 29 34 39 44 49
[5,]  5 10 15 20 25 30 35 40 45 50
> write.table(y,'clipboard')
```

Then we open an Excel file and right-click our mouse to paste it. We will end up with just one column of data (see below).



To separate the above one column into ten columns, we highlight the first column, click **Data** then **Text to columns**. Then we choose **delimited** and choose **space** as our delimiter (see below).

19.6. Input from a Delimited Input File

When retrieving files with a specific delimiter, we can use the **delim()** function.

```
> x<-read.delim("c:/temp/test2.txt",sep="*",header=T)
```

Alternatively we can use the **read.table()** function by specifying **sep='*'**.

```
> y<-read.table("c:/test2.txt",sep="*") # sep is separator
```

19.7. Input from a Fixed Width File read.fwf()

First, let's generate a text file called **test.txt**. The codes are in the left panel below while the final output is in the right panel.

```
>cat(file="c:/temp/test.txt","123456","987654",sep="\n")
```

```
123456
987654
```

The input format depends how we group those values. Assume that for each record, there are three different data items with lengths of 1, 2, and 3. Thus, for the first record, we have **data1=1**, **data2=23**, and **data3=456**. Then we can use **widths=c(1,2,3)**, as in the codes below:

```
> read.fwf(file="c:/temp/test.txt",widths=c(1,2,3))
  V1 V2 V3
1  1 23 456
2  9 87 654
```

To skip certain digits, a negative number is used. For the following codes, since the second one is negative, we end up with two data items only.

```
> read.fwf(file="c:/temp/test.txt",widths=c(1,-2,3))
  V1 V2
1 1 456
2 9 654
```

19.8. load() an R data set

First let's generate a binary R data set.

```
> x<-1:10
> save(x,file="c:/temp/test.Rdata")
```

To upload this data set, we use the **load()** function.

```
> rm(x)
> load("c:/test_R/test.Rdata") # x will be there
```

19.9. Extension .RData of an R Data Set Is Not Critical

In the following codes, we don't even give an extension to the R data set we generated.

```
> y<-1:100
> save(y,file="c:/test_R/abc")
> rm(y)
> load("c:/test_R/abc")
```

19.10. Reading from an Internet File

We can read a file from the Internet (see an example below).

```
> d<-read.csv("http://www.ats.ucla.edu/stat/R/notes/hsb2.csv",header=T)
> head(d)
  id female race ses schtyp prog read write math science socst
1 70 male white low public general 57 52 41 47 57
2 121 female white middle public vocation 68 59 53 63 61
3 86 male white high public general 44 33 54 58 31
4 141 male white high public vocation 63 44 47 53 56
5 172 male white middle public academic 47 52 57 53 61
6 113 male white middle public academic 44 52 51 63 61
```

Next, let us try Yahoo Finance. First, go to the Yahoo Finance at http://finance.yahoo.com/. Then enter "IBM" in the Get Quote box. Click **Historical Prices** on the left-hand side. Go to the bottom of the Web page and click **Download to Spreadsheet**. Alternatively we can use a simple line of to get this file.

```
> x<-read.table('http://canisius.edu/~yany/data/ibm.csv',sep=',',header=T)
```

We can assign this data set to a new variable and show its first couple of lines.

```
> t1<-'http://finance.google.comifinance/historical?q='
> t2<-paste(t1,'IBM&startdateJun+02,+1970',sep='')
>x<-read.table(paste(t2,'&enddate=Jun+16,+','2017&output=csv',sep=''),sep=',')
> head(x)
     V1    V2    V3     V4    V5     V6
1    Date  Open  High  Low  Close  Volume
2 15-Jun-17 153.29 154.69 153.29 154.22 4654297
3 14-Jun-17 153.97 154.94 152.94 153.81 3049726
4 13-Jun-17 155.44 155.48 154.15 154.25 3523529
5 12-Jun-17 154.19 157.20 154.02 155.18 6471479
6 9-Jun-17 152.00 154.26 151.88 154.10 4361460
```

Of course we can use the **read.csv()** function to simplify our codes a little bit instead of using the **read.table()** function.

```
> t1<-'http://finance.google.com/finance/historical?q='
> t2<-paste(t1,'IBM&startdate=Jun+02,+1970',sep'')
> x<-read.csv(paste(t2,'&enddate=Jun+16,+','2017&output=csv',sep=''))
> head(x)
      Date    Open    High    Low  Close  Volume
1 15-Jun-17 153.29 154.69 153.29 154.22 4654297
2 14-Jun-17 153.97 154.94 152.94 153.81 3049726
3 13-Jun-17 155.44 155.48 154.15 154.25 3523529
4 12-Jun-17 154.19 157.20 154.02 155.18 6471479
5 9-Jun-17  152.00 154.26 151.88 154.10 4361460
6 8-Jun-17  151.00 152.82 150.92 152.10 3708962
```

19.11. Reading from canisius.edu/~yany

We keep many CSV files or R Data sets at the above Web site.

```
>x<-read.csv("http://canisius.edu/~yany/ibm.csv",head=T)
```

Again, to view the contents of the variable, we can use the **head()** function.

```
> head(x)
  Date Open High Low Close Volume Adj.Close
1 2010-12-03 144.25 145.68 144.25 145.38 3710600 145.38
2 2010-12-02 144.33 145.85 144.30 145.18 5374000 145.18
3 2010-12-01 143.61 145.13 143.51 144.41 6822800 144.41
4 2010-11-30 142.24 142.76 141.28 141.46 7674800 141.46
5 2010-11-29 143.53 143.67 141.50 142.89 5040300 142.89
6 2010-11-26 145.30 145.30 143.57 143.90 2081300 143.90
```

Here is another example:

```
>x<-read.table("http://canisius.edu/~yany/F-F_Research_Data_Factors2.
txt",skip=3,header=T)
> head(x)
 Mkt.RF SMB HML RF
192607 2.95 -2.46 -2.87 0.22
192608 2.63 -1.19 4.60 0.25
192609 0.38 -1.33 -0.23 0.23
192610 -3.24 -0.08 0.20 0.32
192611 2.54 -0.29 -0.24 0.31
192612 2.62 -0.20 -0.05 0.28
```

If we intend to retrieve several data sets from the same Web page, it is not convenient to specify the whole path every time. Thus, we are better off defining a variable that changes with different keywords. Below, a variable called **data_set** is such a variable.

```
> http<-"http://canisius.edu/-yany/data/"
> data_set<-"ffMonthly.txt" # change this one only
> location<-paste(http,data_set,sep="")
> x<-read.table(location, header=T)
```

19.12. Finding Help for Inputting Data from an External File

We can use the **help()** function to find the related information about those functions.

```
> help(read.table)
> help(read.csv)
> help(read.delim)
> help(read.fwf)
```

19.13. Some R Data Sets from the Internet

The data set called **ff.Rdata** has two data sets: Fama-French monthly factors and Fama-French daily factors (http://canisius.edu/~yany/RData/ff.RData). We can use the following codes to load the data set called **ff.RData**.

```
> con<-url('http://canisius.edu/~yany/RData/ff.RData')
> load(con)
> close(con)
> ls()
[1] "con " "ff_daily_factors " "ff_monthly_factors"
```

19.14. Inputting Files with Irregular Formats

Below, we try to write an R program to retrieve correct files by using the readLines() function. In the following input file, we need to input data lines that contain seven data items.

generated 12/12/2011
Date Open High Low Close Volume Adj.Close
2011-12-20 10.21 10.38 10.18 10.33 45218900 10.33
2011-12-19 10.25 10.39 9.99 10.02 45055000 10.02
2011-12-16 10.32 10.4 10.16 10.25 45882000 10.25
end of the file

After we use the **readLines()** function, we generate a variable with six data items, which represent the number of lines.

```
> x<-readLines("c:/temp/test.txt")
> x
[1] "generated 12/12/2011"
[2] "Date Open High Low Close Volume Adj.Close"
[3] "2011-12-20 10.21 10.38 10.18 10.33 45218900 10.33"
[4] "2011-12-19 10.25 10.39 9.99 10.02 45055000 10.02"
[5] "2011-12-16 10.32 10.4 10.16 10.25 45882000 10.25"
[6] "end of the file "
```

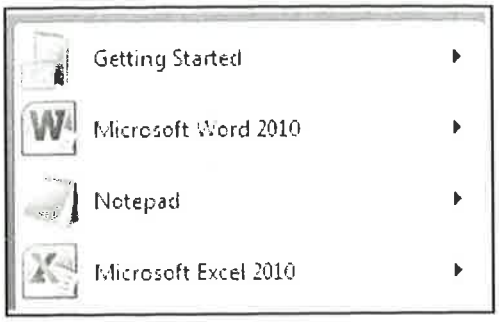
We know that the first line has two data items and is not a regular line, which have seven data items.

```
> k<-t(sapply(strsplit(x[1]," "),unlist))
> length(k)
[1] 2
> k<-t(sapply(strsplit(x[2]," "),unlist))
> length(k)
[1] 7
```

Now we have a way to get regular lines that contain only seven data items. Below is our complete program.

```
x<-readLines("c:/temp/test.txt ")
n<-length(x)
out<-NA
for(i in 1:n){
 k<-t(sapply(strsplit(x[i]," "),unlist))
 if(length(k)==7) out<-rbind(out,k)
 #print(length(k))
}
out2<-subset(out,!is.na(out[,1]))
> out2
 [,1] [,2] [,3] [,4] [,5] [,6] [,7]
 "Date " "Open" "High" "Low" "Close" "Volume" "Adj.Close"
 "2011-12-20" "10.21" "10.38" "10.18" "10.33" "45218900" "10.33"
 "2011-12-19" "10.25" "10.39" "9.99" "10.02" "45055000" "10.02"
 "2011-12-16" "10.32" "10.4" "10.16" "10.25" "45882000" "10.25"
```

There are many ways to generate a text file. For example, we can use Notepad. To launch Notepad, click. After that, we see the following setup:

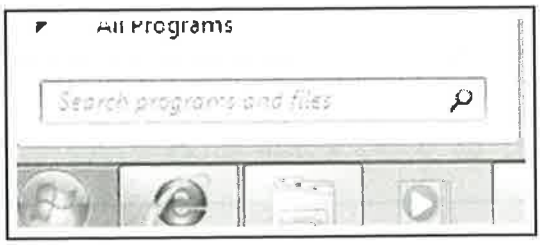


Click Notepad, type the following records, then save the output, such as **test.txt.**

29161 19761020
15763 19841229
10093 19830215

The second way to launch Notepad is to click **All Programs** then **Accessories** then **Notepad.** The third way to launch Notepad has two steps:

Type "notepad" in the following box (Search program and files).



Exercises

19.1. Use Notepad to generate a text file and input it into R.
-8.086741 10.011198 4.560525 -14.342503 -2.653048
6.417692 -4.150210 -4.595757 -7.924940 -11.585391

19.2. Launch Excel to generate a file and save it with a CSV format. Input the file into R.

19.3. Go to Yahoo Finance and download DELL's historical price data and input it into R.

19.4. Based on the result from 19.3, estimate the returns for DELL.

19.5. We generated an R data set called **ff.Rdata,** which has two data sets (called **ff_monthly_factors** and **ff_daily_factors**). Please download it at http://canisius.edu/~yany/RData/ff.Rdata then upload and play with it.