Importing Data

Importing data into R is fairly simple. For Stata and Systat, use the foreign package. For SPSS and SAS I would recommend the Hmisc package for ease and functionality. See the Quick-R section on packages, for information on obtaining and installing these packages. Example of importing data are provided below.

From A Comma Delimited Text File

```
mydata <- read.table("c:/mydata.csv", header=TRUE,
  sep=",", row.names="id")
```

From Excel

One of the best ways to read an Excel file is to export it to a comma delimited file and import it using the method above. Alternatively you can use the **xlsx** package to access Excel files.

```
library(xlsx)
mydata <- read.xlsx("c:/myexcel.xlsx", 1)

# read in the worksheet named mysheet
mydata <- read.xlsx("c:/myexcel.xlsx", sheetName = "mysheet")
```

Exporting Data

There are numerous methods for exporting R objects into other formats. For SPSS, SAS and Stata, you will need to load the foreign packages. For Excel, you will need the xlsReadWrite package.

To A Tab Delimited Text File

```
write.table(mydata, "c:/mydata.txt", sep="\t")
```

To an Excel Spreadsheet

```
library(xlsx)
write.xlsx(mydata, "c:/mydata.xlsx")
```

A binary file is a file that contains information stored only in form of bits and bytes.(0's and 1's). They are not human readable as the bytes in it translate to characters and symbols which contain many other non-printable characters. Attempting to read a binary file using any text editor will show characters like Ø and ð.

The binary file has to be read by specific programs to be useable. For example, the binary file of a Microsoft Word program can be read to a human readable form only by the Word program. Which indicates that, besides the human readable text, there is a lot more information like formatting of characters and page numbers etc., which are also stored along with alphanumeric characters. And finally a binary file is a continuous sequence of bytes. The line break we see in a text file is a character joining first line to the next.

Sometimes, the data generated by other programs are required to be processed by R as a binary file. Also R is required to create binary files which can be shared with other programs.

R has two functions **WriteBin()** and **readBin()** to create and read binary files.

Syntax

writeBin(object, con)

readBin(con, what, n)

Following is the description of the parameters used −

- **con** is the connection object to read or write the binary file.
- **object** is the binary file which to be written.
- **what** is the mode like character, integer etc. representing the bytes to be read.
- **n** is the number of bytes to read from the binary file.

XML is a file format which shares both the file format and the data on the World Wide Web, intranets, and elsewhere using standard ASCII text. It stands for Extensible Markup Language (XML). Similar to HTML it contains markup tags. But unlike HTML where the markup tag describes structure of the page, in xml the markup tags describe the meaning of the data contained into the file.

You can read a xml file in R using the "XML" package. This package can be installed using following command.

```
install.packages("XML")
```

Input Data

Create a XMl file by copying the below data into a text editor like notepad. Save the file with a **.xml** extension and choosing the file type as **all files(*.*)**.

<RECORDS>

```
<EMPLOYEE>
  <ID>1</ID>
  <NAME>Rick</NAME>
  <SALARY>623.3</SALARY>
  <STARTDATE>1/1/2012</STARTDATE>
  <DEPT>IT</DEPT>
</EMPLOYEE>

<EMPLOYEE>
  <ID>2</ID>
  <NAME>Dan</NAME>
  <SALARY>515.2</SALARY>
  <STARTDATE>9/23/2013</STARTDATE>
  <DEPT>Operations</DEPT>
</EMPLOYEE>

<EMPLOYEE>
  <ID>3</ID>
  <NAME>Michelle</NAME>
  <SALARY>611</SALARY>
  <STARTDATE>11/15/2014</STARTDATE>
  <DEPT>IT</DEPT>
</EMPLOYEE>

<EMPLOYEE>
  <ID>4</ID>
  <NAME>Ryan</NAME>
  <SALARY>729</SALARY>
  <STARTDATE>5/11/2014</STARTDATE>
  <DEPT>HR</DEPT>
</EMPLOYEE>

<EMPLOYEE>
  <ID>5</ID>
```

```
    <NAME>Gary</NAME>
    <SALARY>843.25</SALARY>
    <STARTDATE>3/27/2015</STARTDATE>
    <DEPT>Finance</DEPT>
  </EMPLOYEE>

  <EMPLOYEE>
    <ID>6</ID>
    <NAME>Nina</NAME>
    <SALARY>578</SALARY>
    <STARTDATE>5/21/2013</STARTDATE>
    <DEPT>IT</DEPT>
  </EMPLOYEE>

  <EMPLOYEE>
    <ID>7</ID>
    <NAME>Simon</NAME>
    <SALARY>632.8</SALARY>
    <STARTDATE>7/30/2013</STARTDATE>
    <DEPT>Operations</DEPT>
  </EMPLOYEE>

  <EMPLOYEE>
    <ID>8</ID>
    <NAME>Guru</NAME>
    <SALARY>722.5</SALARY>
    <STARTDATE>6/17/2014</STARTDATE>
    <DEPT>Finance</DEPT>
  </EMPLOYEE>
</RECORDS>
```

Reading XML File

The xml file is read by R using the function **xmlParse()**. It is stored as a list in R.

```
library("XML")
library("methods")
result <- xmlParse(file = "input.xml")
print(result)
```

When we execute the above code, it produces the following result −

```
1
Rick
623.3
1/1/2012
IT

2
Dan
515.2
9/23/2013
Operations

3
Michelle
611
11/15/2014
IT

4
Ryan
729
5/11/2014
HR

5
Gary
843.25
3/27/2015
```

Finance

6
Nina
578
5/21/2013
IT

7
Simon
632.8
7/30/2013
Operations

8
Guru
722.5
6/17/2014
Finance

Get Number of Nodes Present in XML File

```
library("XML")
library("methods")
result <- xmlParse(file = "input.xml")
rootnode <- xmlRoot(result)
rootsize <- xmlSize(rootnode)
print(rootsize)
```

When we execute the above code, it produces the following result −

output

```
[1] 8
```

Details of the First Node

Let's look at the first record of the parsed file. It will give us an idea of the various elements present in the top level node.

```
library("XML")
library("methods")
result <- xmlParse(file = "input.xml")
rootnode <- xmlRoot(result)
print(rootnode[1])
```

When we execute the above code, it produces the following result −

```
$EMPLOYEE
   1
   Rick
   623.3
   1/1/2012
   IT
 attr(,"class")
[1] "XMLInternalNodeList" "XMLNodeList"
```

Get Different Elements of a Node

```
library("XML")
library("methods")
result <- xmlParse(file = "input.xml")
rootnode <- xmlRoot(result)
print(rootnode[[1]][[1]])
print(rootnode[[1]][[5]])
print(rootnode[[3]][[2]])
```

When we execute the above code, it produces the following result −

```
1
IT
```

Michelle

XML to Data Frame

To handle the data effectively in large files we read the data in the xml file as a data frame. Then process the data frame for data analysis.

```
library("XML")
library("methods")
xmldataframe <- xmlToDataFrame("input.xml")
print(xmldataframe)
```

When we execute the above code, it produces the following result −

```
   ID  NAME    SALARY  STARTDATE      DEPT
1   1  Rick    623.30  2012-01-01     IT
2   2  Dan     515.20  2013-09-23     Operations
3   3  Michelle 611.00  2014-11-15     IT
4   4  Ryan    729.00  2014-05-11     HR
5  NA  Gary    843.25  2015-03-27     Finance
6   6  Nina    578.00  2013-05-21     IT
7   7  Simon   632.80  2013-07-30     Operations
8   8  Guru    722.50  2014-06-17     Finance
```

As the data is now available as a dataframe we can use data frame related function to read and manipulate the file.