Chap I: Introduction to R: Input/Output and graphics

Data Input/Output

- It is important to be capable of moving data in and out of R.
- We make a distinction between data files and R variables and objects.

Data Input/Output

- To save R internal objects, use the function save. It creates a platform independent binary file, typically with extension ".rda", or ".RData".
- Use load to load such files.

```
save(llfun,x,file='text.RData')
#same as
save(list=c('llfun','x'),file='text.RData')
#notice the presence/absence of quotes
load('test.RData')
```

- Script files are typically saved with extension *.R*. These are plain text documents.

Data Input/Output

- *R* can write matrix and data frames to file using the function 'write.table'. And read data from file using 'read.table' and its specialized versions.
- To read a text file you <u>need</u> to understand the format of the file.
- The most common text file formats are: *comma-separated file* (CSV), and *tab-delimited file*.

Data Input/Output

- Here is an example of CSV file. In fact the separator is not very important, and could be different.

  ```
  Year,Student,Major
  2009, John Doe,Statistics
  2009, Bart Simpson, Mathematics
  ```

- Here is a tab-separated file.

  ```
  Year       Student          Major
  2009       John Doe         Statistics
  2009       Bart Simpson     Mathematics
  ```

Data Input/Output

- To read a csv file, use the function read.csv. If in the file the decimal character is the comma (as in the french language for instance), then use the function read.csv2.
- To read a tab-delimited file use the function read.delim, or read.delim2.
- The above functions internally call 'read.table'.

Data Input/Output

Other things to keep in mind when reading or writing to file:

- header: whether the file has a first row giving the names of the variables.

  `header=TRUE/FALSE`

- Missing data character string: What character strings serve as missing data.

  `na.strings=c(...)`

- Do you want to allow R to convert characters variables to factors?

  `stringsAsFactors=TRUE/FALSE and as.is=c(...)`

- Are there some initial rows to skip?

  `skip=`

Data Input/Output

- The file 'UNData.tab' is a dataset that gives all the United Nation resolutions over the years. The first few rows:

```
session rcid rcid2 abstain yes no year day month date u
1 3 1 4 29 18 1946 1 1 1946-01-01 GAOR-1-66 1 0 AMENDME
1 4 2 8 9 34 1946 2 1 1946-01-02 GAOR-1-79 0 0 SECURITY
```

- We read as follows.

```
undata=read.delim(file='UNData.tab')
##There are some issues with the quote.
       Better to disable the quote
undata = read.delim('UNData.tab', header=T, quote="")
```

Data Input/Output

- The file 'spectrum.csv' is a remote sensing dataset.
- To properly load it in R we need to open it in a text editor and understand its structure.
- We see that although the naming suggests a csv format, it is actually a tab delimited file
- There are also some initial explaining in the file that we skip.

  ```
  specdt=read.delim(file='spectrum.csv',header=F,skip=26
  names(specdt)=c('wavelen','reflec')
  ```

- Try using read.csv to open this file.

Data Input/Output

- The file 'trafficflow.txt' contains some data on highway traffic in California.
- After opening the file (using a text editor such as TextEdit) we see that is it a csv format. It is a very clean file. We read it as follows.

```
trafficdt=read.csv(file='traffic_flow.txt');
```

Data Input/Output

- Another function to read text data is read.fwf that works with fixed-width text data. See the user manual for more detail.
- Yet, another function to read data from file is 'scan'. It is more efficient when reading data of a single mode. See the user manual.

Data Input/Output

- We can also save data in text format using the function write.table.
- The data set 'airquality' is available is R and gives weather measurement in New York city over some period of time.
- Load that data set in a data frame and save it to a file.

Data Input/Output

```
dt=airquality
##to create plain space separated file
write.table(dt,file='Airquality.dt',col.names=T,
         row.names=F,sep="  ");
##to create tab delimited file
write.table(dt, file='Airquality.dt',col.names=T,
         row.names=F,sep="\t");
##to create cvs file
write.table(dt, file='Airquality.dt',col.names=T,
         row.names=F,sep=",");
```

You could also use 'write.csv'. See the help documentation for details.

Data Input/Output: MSExcel data

- There is no direct function for loading MS Excel data in R.
- The simplest way is to open the file in MS Excel (or some other similar program) and save it as a tab-delimimited text file as discussed above.
- Then use read.delim to load it into R.

Data Input/Output: MSExcel data

The file 'earmarks08.xls' is an Excel file giving US Congress Earmarks in 2008. Load this file in R.

Graphics

Example: the airquality data set.

```
dt=airquality
names(dt)  # names of the variables
boxplot(dt$Temp)
plot(dt$Temp,type='l')
plot(dt$Temp,dt$Wind,type='p')
plot(dt$Temp,dt$Wind,type='p',xlab='Temperature',
ylab='Wind', main='Wind vs Temp.
                          in NY city May-Sept. 73')
```

Graphics

- What if we want to have multiple graphics on the same graphical device? There are many ways to do this.
- A quick way is to use the option mfrow or mfcol of the function par. The following code

  ```
  par(mfrow=c(nr,nc))
  ```

  will split the plotting screen in *nr* by *nc* squares. Subsequent plotting instructions will fill the squares row by row.
- Another (more flexible) possibility is the function layout.

Graphics

Example: the airquality data set. Boxplot and time plot of temp.

- Using par

  ```
  par(mfrow=c(1,2))
  boxplot(dt$Temp,main='Boxplot')
  plot(dt$Temp,type='l',main='Time series plot')
  ```

- Using layout

  ```
  m=matrix(c(1,2),ncol=2)
  layout(m)
  boxplot(dt$Temp,main='Boxplot')
  plot(dt$Temp,type='l',main='Time series plot')
  ```

Graphics

But layout is more flexible. Example: the airquality data set.

```
m=matrix(c(1,3,2,3),2,2)
layout(m)
layout.show(3)
boxplot(dt$Temp,main='Boxplot Temp. in NY city')
plot(dt$Temp,type='l',main='Temp. in NY city')
plot(dt$Temp,dt$Wind,type='p',xlab='Temp',
       ylab='Wind',main='xyplot')
```

Graphics

- What if we want to put multiple graphs on the same plot.
- issue

  `par(new=T)`

  first.

Graphics

Example:

```
n=10000;
X=rnorm(n);
hist(X,breaks=200,prob=T,col='blue',
          xlim=c(-4,4),ylim=c(0,0.4))
par(new=T)
curve(dnorm,xlim=c(-4,4),ylim=c(0,0.4),lwd=2,col='red'
        ,xlab='',ylab='')
```

Note: to superimpose two graphics, you need to control the xlim and ylim of the plotting screen as done above.

Graphics

Here is an example using a legend.

```
n=1000;
X=rnorm(n);
dens=density(X)
plot(dens, col='blue',lty=2
          xlim=c(-4,4),ylim=c(0,0.4),main='')
par(new=T)
curve(dnorm,xlim=c(-4,4),ylim=c(0,0.4),lty=1,col='red'
        ,xlab='',ylab='')
legend(x='topleft',
       legend=c('Estimated density','True density'),
        col=c('blue','red'),lty=c(2,1))
```