How to Use R

February 14, 2007

1 What is R

R is open domain statistical software based on the language *S*. Open domain language *S* was originally written at Bell-Labs. S-Plus is a commercial statistical software also based on *S*. BOTH S-Plus and R are also programming languages and hence allow one to do innovative things with the data.

R is open domain and downloadable from the CRAN site at (size about 24 MB for Windows version)

```
http://cran.r-project.org/
```

The current version is 2.4.1. **Make sure to install the reference manual.** The site can be used to download many other packages that address particular statistical methodologies. However, the packages included in the basic download are extensive enough for all but the most specialized user. Even though it was written by statisticians for use in statistics, it can be used for many other functions as well. It has superior graphical facility.

S-Plus is more *user-friendly*, more *buggy*, more illogical and more expensive than R. The two are similar but not the same. In R, many more things need to be done with commands rather than with a drop down menu as in S-Plus. Both programs are unix based at heart though versions of both are available for Windows, Unix, and the Mac.

2 Getting Started

Double click on the R icon to load the program and use

File → Exit

to quit. **Make sure to choose the option "No" when it asks whether to save workspace image or not.** If any newly created objects are saved once, they would need to be deleted manually later. Command `objects()` will show all objects in the working directory and `rm(x,y)` will remove objects `x,y` from the working directory. **This command will NOT remove objects that got saved while exiting the previous time. In order to remove those objects,** `c` → `Program Files` → `R` → `R commands window`.

and manually delete it.

`R` does not automatically load all **available** commands. It loads only the commands of the basic stat and graphics packages that get loaded automatically. Various other commands are available in different **packages**. Most common users will need nothing more than what gets loaded automatically. Some other packages besides the basic ones, also get downloaded along with R, but stay in the background. They can be brought forward with

`Packages` → `Load package` → `MASS`

will load the package `MASS` from the list. Each time the `R` session is closed, the added packages disappear from the working directory. A loaded package can be detached without closing the session with

```
detach(package:MASS).
```

Many other packages are available from the site for downloads. These are contributed by various interested parties. They are usually targeted for a particular statistical methodology. For example the package *boot* has bootstrap functions. At last count, the list of available bundles and packages included 36 items in ”a”!

3 Help

- `Help` in `R` gives a sub-menu that contains `FAQ on R`, `FAQ on R for Windows` as well as `Introduction to R` and the `Manuals`.

  `Help` → `Manuals` → `R Reference Manual`.

  will load the `Acrobat Reader`. Explanation of the command may appear somewhat cryptic initially, but it gives examples and hence is quite useful.

- One can also use e.g. `help.search("time series")` to get information on the topic of time series. It will list number of commands used in time series analysis along with the name of the package in which the commands exist. (It is not an exhaustive list.) Command `?cor` can be used to produce information on how to compute the correlation coefficient because `cor(,)` is a valid `S` command. Command `?regression` will give an error message because `regression` is not a valid `S` command.

4 Objects and Data Types

The language `S` uses several forms of data such as `vectors, matrices, arrays, dataframes` and `lists`. Each type of data has different properties. `S` is `object oriented` and acts on `vectors` but built-in functions are particular about what type of data can be used as the input. It is getting cleverer and cleverer as time goes by so the command **hist**, which draws histograms of a vector of observations, will also work on numeric matrices, but will freak out if there are headers to the columns and hence it is not numeric.

Commands such as

```
is.matrix(m)
```

will say `TRUE` if the object `m` is a matrix and `FALSE` if it is not. The command `data.class(m)` will give result `"matrix"` if `m` is a matrix. Data can be forced into other forms than its own under certain circumstances. For example, if `m` is a numeric matrix, `as.vector(m)` will make a vector of the matrix entries reading column-wise. A particular command may not work because the data is in a wrong format.

5 Creating Data

- Easiest type of data to create is a vector. For example,

  ```
  x ← c(1,4,2,6).
  ```

- Command

  ```
  x ← scan(file="")
  ```

  will prompt for input. Enter each number. Clicking *Enter* twice will end the data input.

- Command

  ```
  m ← matrix(1:16,ncol=4)
  ```

  will make a 4x4 matrix.

6 Spread Sheet in Data Window

One CAN create a spread sheet in `R` and then export it to where ever you want to save it. It is not at all RECOMMENDED. There are many easier ways to record data. The easiest is to use `Notepad`. However, if one wants to, here is how to do it.

Empty spread sheet named `test` can be obtained with commands

```
test ← edit(as.data.frame(NULL)).
```

After entering the data, the updated file `test` gets automatically saved by simply closing it. The default **column names** `Var1,Var2..` can be changed by clicking on the names. Column names can also be added to an existing file `test` with the command

```
colnames(test) ← c("Age","Wealth").
```

Once an object named test exists,

```
edit(test)
```

will bring it up and one can **edit** the cells. **However, if the edited version needs to be saved, reassign it using**

```
test ← edit(test).
```

and then close the edited spread sheet. It is possible to invoke various other editors for the spread sheet. Find help with ?edit.

The data set can be **exported** with the command

```
write.table(test,"c:/yash/newdat/test.txt").
```

7 Import Data

One needs to use different protocols for importing different types of files. Easiest to import is a .txt file that follows R protocol.

```
read.table("c:/yash/newdat/test.txt")
```

will import file test.txt (the extension .txt is necessary) from the address stated. The option header=T **MUST be used if and only if** the columns have names. Otherwise the data imports incorrectly. Ironically, and blissfully, R is **NOT case sensitive in the command** read.table(). THIS IS THE ONLY PLACE WHERE IT IS NOT.

8 Resident Data Sets

Several data sets automatically get loaded at the time of loading the package. These serve as examples in various Help items. These are **the only data sets that can be accessed without having to import them first in the working directory.** A list of the resident data sets can be obtained with

```
data().
```

Even though one does not see their names in the objects in commands window, one can work with them if the exact name of the data set is used. For example USArrests[,1] will print out the first column, named ”Murder”, of the data set USArrests. Command data(?USArrests) will give information on what the data is, which is usually quite scanty. The column ”Murder” will not be recognized by that name until one attaches the data set with the command attach(USArrests).

9 Graphics

Some demos of graphs from the base package can be viewed with command `demo(graphics)`. All graphs need to be drawn with commands and each graphics command has plenty of options to fine tune one's graph. A graphics screen can be **split** by `par(mfrow=c(2,3))`. Successive graphs will be filled by row.

Existing graphs can be augmented with commands such as `lines(x,y)` (which draws a line with intercept x and slope y) or `abline(h=c(y1,y2,y3))` (which draws three horizontal lines at y = y1, y2, and y3) etc. Text, symbols and legends can be inserted in an existing graph. Commands `locator()`, or `identify()` will make an active cursor in the existing graph and by successively clicking on desired points, we can find the co-ordinates of those points.

Commonly used graphs can be drawn with commands `plot(x,y)`, `hist(x)`,`boxplot(air)` etc.

10 Missing Values and Logical Operations

Missing values need to be handled in computational commands. The option `na.rm=T` removes missing values for most of the functions such as mean, median, var etc. For functions such as `cor`, use option `use="complete.obs"`. Missing values from the entire data set can be removed by `na.exclude(air)`. Many graphics commands use this option as the default.

If x and y be two vectors of **same length**. Then `y[y == x]`will give a subvector of y such that $y[i] = x[i]$. `y[y!=x[1]]` will give all values of `y` not equal to `x[1]`. We can extract **sub-matrices** such as `m[m[,2]>13 & m[,4]<38,]`.

11 Standard Distributions

For a list of available probability distributions and their `R` names, see table on page 37 of the `Introduction to R`.

```
Help → Manuals
  → An introduction to R
  → Probability Distributions
```

will get the list. For each one of these distributions the probability density, distribution function, quantiles, and simulation of the distribution is obtained by adding respectively `d, p, q` and `r` behind the `R` name of the distribution. The table also gives the parameters of these distributions. Any arbitrary discrete distribution can be simulated with `sample(x,50,replace=T,prob=p)`.

12 Programming in R

The greatest payoff of `R` is that it is also a programming language. Small programs can be written in a `Notepad` file and then run in the working directory by copying and pasting the program. For example, the function `meanunif` simulates $nU(0,1)$ random variables, computes its mean, and repeats the process N times.

```
meanunif <-function(n,N){
    x<-1:N
    for (i in 1:N){x[i]<-mean(runif(n))}
    x
    }
```