Getting Started with R

Jan Rovny

What is R?

R is a platform for the object-oriented statistical programming language S. It is widely used in statistics and has become increasingly popular in Political Science and Sociology over the last decade. R can be used as either a matrix-based programming language or as a standard statistical package that operates much like STATA, SAS or SPSS.

Getting R

The beauty of R is that it is free to download. To obtain R for Windows, Mac, or Linux, we can go to The Comprehensive R Archive Network (CRAN). There are no updates, so we must install a new version whenever we wish to upgrade.

While R is a program that executes our data management and statistical estimation, it is practical to run R via a so-called 'front-end' program, named RStudio. RStudio can be downloaded from here. It gives us a better visualization of what we are doing in R, and provides a number of useful tools. While you need to download and install both R and RStudio, once this is done, you only need to open RStudio (it will automatically start R).

Resources

As questions emerge when you use R, here are a few resources you may want to consider:

- Within R, there are the following help commands:

```
?
help
```

- RStudio provides support and tutorials accessibe *here*
- To do a web-based search for R related questions, *Rseek* (powered by Google) is a great search engine.
- Some useful introductory manuals are available from the CRAN web- site:
  - John Verzani *Simple R*
  - Grant Fairnsworth *Econometrics in R*
- UCLA Academic Technology Services have a very useful *website* covering many different statistical packages including R
- For quick reference *Quick-R* is very useful.

Getting Started

Once we install R and RStudio, we open RStudio to start working. There are a few pull-down menus, but commands in R are mostly entered on the command line:

- Some preliminaries on entering commands:
- Expressions and commands in R are case-sensitive.
- Command lines do not need to be separated by any special character, just push [Return]

- Anything following the pound character (#) R ignores as a comment. Comments are very useful when we program in R, because they allow us to remember what we have done (or meant to do) when we come back to our work later. Learning to neatly indicate what we are doing and why saves a lot of work!
- An object name must start with an alphabetical character, but may contain numeric characters afterwards. A period may also form part of the name of an object. For example, x.1 is a valid name for an object in R.
- You can use the arrow keys on the keyboard to scroll back to previous commands. As mentioned, R is an object-oriented language. A basic object can be a variable x with ten values: (0,1,2,3,4,5,6,7,8,9). To create such a variable, simply type:

```
x<-c(0,1,2,3,4,5,6,7,8,9)
```

Notice, that we are telling R to create x by specifying the operator function <-. This is a widely used symbol in R. The command c() simply connects the ten values into the object x. Objects. R saves all objects we create, whether they are variables, tables or model results. To list the objects we have created in a session we use either of the following commands:

```
objects()
ls()
```

To remove all the objects in R type:

```
rm(list=ls(all=TRUE))
```

To remove one object type:

```
rm(objectname)
```

To quit R type:

```
q()
```

Packages.

R has many useful add on components written by various statis- ticians that are called ???packages???. To load a package you simply type:

```
library(packagename)
```

It is, however, possible that our copy of R does not contain the particular package we want. In this case we must first download the package from the web. To download a package type: install.package("packagename") Programming. While we can type R commands one line at a time di- rectly into the R console, this is cumbersome and inefficient. Instead, most users type R commands into a text editor and then copy and paste them into R. This is most simply done with Notepad or with Document Editor which is directly in R. Type your R commands together with comments into the text editor and then cut and paste into R and press [Return] to run the program.

Importing Data into R

Getting data into R is quite easy. There are two primary ways to import data. The first is to read in a delimited text file with the read.table command.

```
Dataname<- read.table("/Users/path/datatable.txt", header=TRUE, na="NA")
```

When we work in Windows, the path will look something like this:

```
"D:/path/datatable.txt"
```

Remember we have to specify the exact location of the file ourselves depending on where the file is saved. The header option specifies whether the first line of the file contains the names of the variables.

If we now type:

```
Dataname
```

The dataset will appear on the screen. To check the content of the dataset, type:

```
names(Dataname)
```

More frequently, we import data from other statistical package formats, such as STATA (.dta) or SPSS (.sav). The 'rio' package in R makes it very easy to import data from other statistical packages. First we must install the foreign package. To install it (or any other R package) type:

```
install.packages("rio")
```

and follow the prompts. Once the package is installed, we must activate it:

```
library(rio)
```

Now we are ready to import a foreign data file.

```
Dataname<-import("/path/filename.dta")  #STATA datafile
Dataname<-import("/path/filename.sav") #SPSS datafile
```

Do not forget to specify the exact path to the imported file.

Missing Data.

R designates missing values with

```
NA
```

It translates missing values from other statistics packages into the NA missing format. To create a new data set with all the missing values deleted through listwise deletion, type:

```
hmnrghts <- na.omit(hmnrghts)
```