Abstract

This short course (roughly 2 hours) presents R, a statistical software package and programming language, which thanks to its open source licensing (read: FREE) and the support of academics across the world, is quickly replacing traditional statistical packages, such as STATA and SAS, in a wide range of academic departments.

This tutorial is intended for students with limited or no previous experience of R but some familiarity with other stats/math packages. The course presents some of the basic operations in R (importing data, running OLS regressions, etc.) as well as some of the uses of R that distinguish the package from its licensed alternatives (programming, graphics, and native matrix multiplication, among others.) After taking this course, students should have an understanding of what R is and what kind trade-offs are involved in using R vs. SAS, STATA, or other similar packages.

1 Topics Covered

- The origins of R
- Advantages and disadvantages of R
- Getting R
- Help
- Setting up your workspace
- Program files (.R)
- Importing data
- A very brief intro to object oriented programming
• Summarizing and analyzing data
• Linear regression
• Getting new R packages
• Why R is sometimes frustrating
• Programming in R
• Graphics in R
• Matrix multiplication in R
• R and \LaTeX

2 The origins of R

In early 2009 the New York Times ran a story about R that gives a good introduction to the history and guiding philosophy behind the project. You can find that story at http://tinyurl.com/6davm8e or by searching "Data Analysts Captivated by R’s Power NYTimes" on Google. Here is my cliff notes version of that story:

• Started in 1991 by Ross Ihaka and Robert Gentleman of the University of Auckland in New Zealand and first released in 1996.

• Based on S, what is now a ye olde statistics language.

• Ihaka and Gentleman wanted a flexible statistical program that could allow statisticians to make their own functions and produce nice graphics.

• Additionally, they did not like the exorbitant licensing fees associated with existing packages.

• Quickly taken up by a die-hard community of academics who churned out packages that make R a gateway to very specific, pre-written routines covering subjects from Astronomy to Zoology.

• In many departments it has become THE statistics package, and it has slowly begun creeping into the private sector through companies like Google and the quantitative finance departments of many big banks.

3 Advantages and disadvantages of R

By far the most important advantage of R is its academic community: the community provides great packages, great documentation, and blogs to walk you through most things you want to do. In the class, we’ll visit some of the sites like http://cran.r-project.org/ and http://www.r-bloggers.com/, where you can see what comprehensive (and free) resources this community put at the hands of everyone who makes the effort to get familiar with R.

On the disadvantages side, we’ll discuss how R’s seemingly infinite customizability can make operations that are relatively simple in SAS or STATA a serious pain for newbies.

See: http://xkcd.com/196/ for a comic on this frustrating phenomenon.
4 Getting R

R will already be installed on all the computers in the lab hosting this tutorial, but we’ll talk about where you can get it for your own computers and introduce a few of the many user interfaces (GUIs in nerd-speak) that you can choose from.

5 Help

Although your hard work learning R will be handsomely rewarded, I want to be clear that R is not particularly easy to start. Even as you get comfortable with the program, you are going to need to call on the help function a fair amount. We’ll go over how you can very quickly get comprehensive documentation, including specific examples, for any function.

6 Setting up your workspace

Using the command:

> setwd("insert your path here/R working directory")

we’ll make sure that your computer is pointed to the right place. This means that we can quickly load up data and begin with the real statistics. (Note: since we are using university computers, some file directories (like the desktop) may have weird limitations on access to files. I’m going to try and make sure we don’t hit this snag, but if you are getting strange errors, that may be your problem.) Also note that on Windows, for whatever reason, you need to use " " or "/" to separate folders when setting out the path to a file.

7 Program files (.R)

Here we walk through the steps needed to load a text file with all the commands that we will run today. This procedure varies a bit depending on what GUI you’re using.

8 Importing data

We’re going to being the tutorial in earnest by importing some data. Since I’m in the Agricultural Economics department, the data are corn prices for different US cities and the transportation costs between those cities …

> costs <- read.csv("lopdata.csv")

Note that because we’ve set the working directory as the folder containing the file with our data, all we have to do is specify the file name when we read in the data. If we wanted to read in data sitting outside our working directory, we’d have to input the full file name, as in:

> costs <- read.csv("your path here /lopdata.csv")
So what have we done with this command? Well we have used the function `read.csv()` to bring in data from `lopdata.csv` and stored that data using the assignment operator `<-` as the object `costs`. I could have named the object anything I wanted, `costs` is just a simple way of remembering what I’m dealing with. (Note: some people don’t like `<-` as the assignment operator. They instead want us to use `=`. Both work find in R, so I’m note sure what the fuss is about. If you care about that sort of thing, and what to know more about the debate, I suggest you check it out on the relevant blogs.)

As we’ll cover later, R is an object oriented programming language. This fact is the foundation for many of it’s functional advantages over SAS or STATA, and why it is capable of doing much of the mathematical operations that we generally associate with programs like MatLab. In all object oriented programming languages the assignment operator, in this case `<-`, is really important because it allows us to do, for example, the following:

```r
> b <- 64
> 6 + b

[1] 70
```

The ability to assign names to objects, such as a matrix or a number, and have them retain their underlying properties, allows R to understand commands, like `6+b`, that might otherwise not make sense. This is a pretty important capability as we’ll see.

So, back to the data we just loaded ...we can check out a few line of the data we have in the object `costs` by entering:

```r
> costs[1:10,]

 pchicago ptoledo trans
 1 2,4597 2,3874 0,5222
 2 2,4902 2,4163 0,5222
 3 2,4902 2,4163 0,5222
 4 2,4902 2,4260 0,5222
 5 2,5191 2,4790 0,5222
 6 2,5256 2,4870 0,5218
 7 2,5095 2,4709 0,5218
 8 2,5802 2,5513 0,5218
 9 2,6348 2,6187 0,5218
10 2,6252 2,5898 0,5218
```

We can look at the objects within `costs`, which in this case are our variables, by entering:

```r
> names(costs)

[1] "pchicago" "ptoledo" "trans"
```

Now we will use the function `attach()` to make the variables in `costs` easily accessible for further analysis. This is analogous to setting the working directory (which we did earlier) insofar as it points R directly to our variables.

```r
> attach(costs)
```
9 A very brief intro to object oriented programming

Now that we have our variables loaded, let’s look at a weird little glitch in the way R understands this particular data set ...

> class(pchicago)
[1] "factor"

For some strange reason, R reads the data associated with the object pchicago as "factors". Normally factors are things like "red", "yellow","blue". Consequently it is impossible to talk about the mean of a series of factors and R is going to give us an error if we ask it for the mean of pchicago. This is an example of object oriented programming in action. We have an object, pchicago, and it has been endowed with certain properties in the eyes of R. When we try to do something that doesn’t square with those properties, R gets upset.

Generally however this linking of objects and properties is great because it means that if, for example, we ask R to summarize an object it as identified as a linear regression, it knows automatically what type of information we’re going to want.

So, before moving on, we have to make sure that R knows that the data in pchicago are not factors, but numerical values. To do this we enter:

> nchicago <- as.numeric(pchicago) ; ntoledo <- as.numeric(ptoledo) ; ntrans <- as.numeric(trans)

This creates three new objects containing the same data we’d already had stored in costs, but correctly identified as numerical values. Note here that ; serves as a line break in our code.

10 Summarizing and analyzing data

Let’s get some very basic summary statistics for our data.

> summary(nchicago)

Min. 1st Qu.  Median    Mean  3rd Qu.   Max.
1.0  524.8  1046.0  1072.0  1618.0  2238.0

> mean(nchicago)
[1] 1072.427

> sd(nchicago)
[1] 641.16

11 Linear regression

Now lets run a regression telling us how much of the variance in corn prices in Chicago is explained by variance in the corn prices in Toledo and the cost of shipping corn between those two cities ...
> base.reg <- lm(nchicago ~ ntoledo + ntrans)
> summary(base.reg)

Call:
  lm(formula = nchicago ~ ntoledo + ntrans)

Residuals:
      Min       1Q     Median       3Q      Max
-1621.35   -37.42    11.75    43.60   858.46

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept) 18.123324   3.875234   4.677  3.06e-06 ***
ntoledo     1.002515   0.003078 325.722  < 2e-16 ***
ntrans    -0.119601   0.016689  -7.166   9.96e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 90.6 on 2637 degrees of freedom
Multiple R-squared: 0.98,   Adjusted R-squared: 0.98
F-statistic: 6.477e+04 on 2 and 2637 DF,  p-value: < 2.2e-16

12 Getting new R packages

Perhaps we want to export the output from this regression to \LaTeX or some other program in a really sharp looking table. R has a canned package that allows us to do this easily called xtable. But, like many of the very specific functions we need, xTable doesn't come preinstalled in the R base package. Instead, it sits out on a server waiting on our beckon call. To get it on our computer we have to use the function install.packages(). Then, once it's installed we have to bring it to R's attention using library().

> install.packages("xtable")

The downloaded packages are in
  /var/folders/TU/TU0cK+I3G38ofvFXtKzkBE+++TI/-Tmp-//Rtmp1XeLN1/downloaded_packages

Generally, once you enter this command, R gives you a list of servers to download from. Choose the one closest to you.
Once the package is installed, you have to bring the package's presence R's attention ...

> library(xtable)

Note: I think I know why you have to use quotes on the first command but not the second. Its a very nerdy explanation so ask me if you're curious.

Okay now that R is ready to make some beautiful tables, lets do it ...

> print(xtable(base.reg, caption = "Beautiful table"))
### Table 1: Beautiful table

|                | Estimate | Std. Error | t value | Pr(>|t|) |
|----------------|----------|------------|---------|----------|
| (Intercept)    | 18.1233  | 3.8752     | 4.68    | 0.0000   |
| ntleto         | 1.0025   | 0.0031     | 325.72  | 0.0000   |
| ntrans         | -0.1196  | 0.0167     | -7.17   | 0.0000   |

R has many many many packages with very cool functions. You can check them out at [http://cran.r-project.org/web/packages/nlme/index.html](http://cran.r-project.org/web/packages/nlme/index.html). In general though to use them you have to:

1. know which package you want  
2. install it  
3. tell R that you want to use it

### 13 Why R is sometimes frustrating

Now that we’ve seen some of R’s basic functionality, I’ll expound on the many ways that R has frustrated me in the last few years. I am totally self taught, and do not consider myself an expert with R so I have a very long history of trial and error with this program.

### 14 Programming in R

So far, we’ve seen R perform functions that are more or less common to SAS or STATA, here we’ll look at some of the things that really make R special when set next to other statistical packages. First, R is not just a stats package. Its a full programming language meaning that you can create your own functions.

There is a simple example of this in the book "Bayesian Computation with R" by Jim Albert in which he creates his own code for a t-statistic function. Keep in mind that the t-statistic is already available in other canned R functions. Instead of focusing on the function itself, imagine how, if you were doing your dissertation and it involved some new calculation or a series of predefined calculations that you’d rather link into one simple function, it would be a huge benefit to be able to make your own function.

Albert begins by explaining the function’s parts. First he gives us two vectors \( x \) and \( y \) whose lengths will be stored as \( m \) and \( n \).

```r
> m <- length(x)  
> n <- length(y)
```

Next we want the pooled standard deviation for the two vectors. As we saw earlier the function \( \text{sd()} \) gives us standard deviation.

```r
> sp <- sqrt(((m - 1) * sd(x)^2 + (n - 1) * sd(y)^2)/(m + n - 2))
```
Finally, we define the t statistic itself.

```r
> t <- (mean(x) - mean(y))/(sp * sqrt(1/m + 1/n))
```

Now let’s put that all together and enter it as a function called `tstatistic` requiring as inputs vectors `x` and `y`.

```r
> tstatistic = function(x, y) {
+   m = length(x)
+   n = length(y)
+   sp = sqrt(((m - 1) * sd(x)^2 + (n - 1) * sd(y)^2)/(m +
+               n - 2))
+   t = (mean(x) - mean(y))/(sp * sqrt(1/m + 1/n))
+   return(t)
+ }
```

Now we’ll use the new function on some made up data. Note we use the function `c()` to join up numbers in a vector.

```r
> data.x <- c(1, 4, 3, 6, 5)
> data.y <- c(5, 4, 7, 6, 10)
> tstatistic(data.x, data.y)
[1] -1.937926
```

### 15 Graphics in R

Beyond its great community and programmability, R is preferred by many stats folks because it’s data visualization is better than other stats packages. Here we’ll go through some very basic graphics. This is meant more as little display of what you can do with R rather than a genuine introduction. Data visualization in R is a huge topic in and of itself.

To start off, let’s look at a histogram of our corn price data:

```r
> hist(nchicago)
```

Like it’s functions R’s graphics are very easily manipulated to meet your need. Here we change the size and number of bars in our histogram by specifying breaks.

```r
> brk <- c(0, 25, 125, 400, 1000, 1050, 5000)
> hist(nchicago, breaks = brk)
```

You can easily put multiple graphs on a single panel. Here, using the function `par()` to specify that we want 1 row of charts and 3 columns. We also need the command `mfrow()`.

```r
> par(mfrow = c(1, 3))
> boxplot(nchicago)
> boxplot(ntoledo)
> boxplot(ntrans)
```
Now we’ll reset the graphics display window and put everything together on the same set of axes.

```r
> par(mfrow = c(1, 1))
> boxplot(ncicagao, ntoledo, ntrans)
```

Here we generate some random data, plot that data, and play with the labels on the axes.

```r
> cookies <- rnorm(500, mean = 50, sd = 60)
> monsters <- rnorm(500, mean = 50, sd = 60)
> plot(monsters, cookies, ylab = "COOKIES!", xlab = "monsters",
+       main = "Size of cookie predicted by size of monster")
> cmreg <- lm(cookies ~ monsters)
> abline(cmreg, col = "blue")
```

15.1 Spatial Econometrics in R

This introduction to R is meant to get you familiar with the software before a workshop led by Dr. Paul Voss of UNC, Chapel Hill on spatial regression analysis (May 17-19, 2011) here at the University of Kentucky. It looks to be a great talk and I encourage you all to go.

With that talk I want to quickly introduce some resources that might be really helpful, if you want more information on using R to display spatial data. I’ve stumbled across two nice tutorials on the subject: [http://www.fabioveronesi.net/rtutorial.html](http://www.fabioveronesi.net/rtutorial.html) [http://www.uclm.es/profesorado/vgomez/](http://www.uclm.es/profesorado/vgomez/) (click on the last link, “ASDAR tutorial files (useR!2010)”) and: [http://www.people.fas.harvard.edu/zukov/spatial](http://www.people.fas.harvard.edu/zukov/spatial)

Quickly, I’ll go over one of the examples. There’s no need to reproduce it now on your computer - if you want to do reproduce it later you can find a step-by-step tutorial at the first link above. I just want you to get a sense of R’s power. Don’t worry if this is a little over your head. In this example, a PhD student in soil science, Fabio Veronesi, has put together some simple spatial data and placed it on a raster image base.

We begin by loading up some packages which I’ve already installed.

```r
> library(rgdal)
> library(maptools)
> library(sp)
```

Now we read some spatial data:

```r
> data<-read.table("data.txt",sep="",header=T)
```

We have to specify that these data are coordinates

```r
> coordinates(data)=~Lat+Lon
```

And we have to specify the coordinate system we’re using.

```r
> proj4string(data)=CRS("+init=epsg:2078")
```

Then we similarly set up a boarder and a raster image as our basic map.

```r
> border<-readOGR("border.shp","border")
> proj4string(border)=CRS("+init=epsg:2078")
> org_mat<-read.asciigrid("org_matter.asc")
> proj4string(org_mat)=CRS("+init=epsg:2078")
```
Now we make the images and save them as files that can be read easily by \LaTeX

\begin{verbatim}
> library(grDevices)
> spplot(org_mat, scales=list(draw=T), sp.layout=list("sp.points", data, pch="+"))
> savePlot("basic",type="eps")
\end{verbatim}

\begin{verbatim}
> spplot(org_mat, scales=list(draw=T), sp.layout=list("sp.points", data, pch="+"), col.regions=terrain.colors(10))
> savePlot("bettercolors",type="eps")
\end{verbatim}
```r
> spplot(org_mat, scales=list(draw=T), sp.layout=list("sp.points", data, pch="+"), col.regions=terrain.colors(11),
> savePlot("colorsandtitle", type="eps")
```
16 Matrix multiplication in R

Another important point of distinction for R is that, unlike STATA or SAS, R can natively do all the same matrix multiplication as MatLab. That means that you can keep more of your work in one program. To show this I’m going to generate 2 vectors and multiply them using the function `t()` for transpose and the function `seq()` for a sequence of numbers. Note that you have to use `%*%` if you are multiplying matrices.

```r
> g <- seq(1:10)
> g
[1] 1 2 3 4 5 6 7 8 9 10
> h <- seq(1:4)
> h
[1] 1 2 3 4
```
You can also manipulate the matrices you’ve created here we sub in 9.73 for the value in the first row and second column.

> ghmatrix[1, 2] <- 9.73

17 R and \LaTeX

Since I started using \R a two years ago, I’ve found that roughly once a month I come in contact with a new function that blows my mind and changes how I work. The function \texttt{Sweave} is definitely one of those mind blowing functions. It has completely changed the way I write papers, by allowing me to write my \R code directly into my \LaTeX documents, run all the calculations, and automatically incorporate the output into my papers. This means that if, for example, I put together a series of really fancy graphics that I want in my paper, but in the course of editing my document, realize there is a mistake in my data, I can fix the mistake, run \texttt{Sweave} and have my entire paper update itself automatically. It is really like magic. And because it saves you so much time and allows you to easily display exactly the calculations you’ve performed it provides the right incentives for academics to make their work more scrutable and more easily reproducible.

In the live tutorial, I’ll walk you through setting up a basic \texttt{Sweave} document.