Chapter 1

The Course

This short course is designed to:

- serve as an introduction to the R language and it's uses
- teach you the basics of R's syntax
- provide an overview of how to implement some rudimentary statistical techniques and compute basic statistics
- showcase some of R's graphical capabilities
- have some fun in the THE STAR LAB

We will not cover all the things you will eventually need to know about programming in R. This course is merely meant to provide you with a basic understanding of how R works and how to get started. There are no prerequisites and I assume no prior programming knowledge. You should be able to use a mouse and a keyboard. If you feel underwhelmed, please be courteous to your colleagues. If you are overwhelmed, immediately let me know. With any luck, however, you will be able to throw away that old TI-81 at the end of this tutorial.

1.1 Housekeeping & Logistics

We will meet in THE STAR LAB from 7:30pm to 9:00pm, starting January 17, 2013. During each meeting we will go over the contents of one chapter of this tutorial. After each meeting/class, I will hand out a small problem set. Since this is not a graded course, it is entirely up to you to complete them. I nevertheless encourage you to work through them. I swear things will make more sense if you do. Also, the problems/puzzles may actually be fun. The answers and all course materials can be found by directing your favorite web-browser to this address: R-Course Webpage.

If you have any questions, please ask. More likely than not, I'll make mistakes, will be unclear, or just plain wrong. So let's clear problems/misunderstandings/confusions out of the way as they come up. So again, if something doesn't make sense or if you don't understand something interrupt and ask. If you encounter errors in my code, in the problem sets, & cetera, let me know. I wouldn't want anybody get frustrated and waste hours on an unsolvable problem.

1.2 Preliminaries

1.2.1 Why R?

Aside from the fact that you will be required to write your own programs in R for PSC 505 and perhaps even PSC 405, R has a number of virtues and advantages compared to other statistical software packages (e.g. Stata, etc ...).

1. It is open-source, it is free!
2. It is cross-platform (Windows, Mac, Linux).
3. It is what "real" scientists use.
4. It has a large active, helpful, and friendly user base.
5. It is updated regularly.
6. It has unrivaled graphical capabilities.
7. It is extremely flexible and can do or be made to do just about anything.
8. It is better than Stata. Period.[1]

1.2.2 What is the catch?

1. It is not a spreadsheet (e.g. Excel). So you do not "see" what's going on.
2. There is no *real* GUI (i.e. no point-and-click interface).[2]
3. It is not the best tool for non-statistical programming (e.g. web scraping). Duh.
4. There is an initially somewhat steep learning curve (especially without any programming background).
5. It comes with ABSOLUTELY NO WARRANTY.

1.2.3 Installing R

R is installed on all computers in THE STAR LAB. If you don't own your own computer or if you do not want to install it on your personal machine, you can ignore this section. Make THE STAR LAB your home. If you decide, however, that you want to use R at your other home. Here is how to get it up and running.

1. Navigate to the Comprehensive R Archive Network (CRAN) website at http://cran.r-project.org/.

[1]Fact!
[2]You can install GUIs and IDEs separately, if you want to be like that. You don't want to be like that.

2. At the top of the page click the appropriate link for your operating system
    - **Windows**: Click on the link to the `base` subdirectory and then on Download R 2.15.2 for Windows. (This is the most recent version as of Dec. 2012.)
    - **Mac OS X**: Click the first link under the "Files" heading and download the file: R-2.15.2.pkg. (This is the most recent version as of Dec. 2012.)
    - **Linux**: Chose the directory of your Linux distribution and follow the instructions. Alternatively you can download the source code from the homepage and compile R yourself.
3. Unless you have chosen to compile R from source, run the executable you just downloaded.
4. You are done.[3]

R is updated regularly (roughly 3 times a year). These updates, contain improvements, bugfixes, and new features. Newly developed or updated packages also often are not backward compatible. As such you definitely want to make sure you keep R up-to-date. Updating can be tedious (e.g. on Windows R needs to be reinstalled). Make sure you keep track of all packages you download and backup your settings (e.g. your `Rprofile.site` file, et al.) to make the updating process as seamless and easy as possible.[4]

1.2.4 Text Editors

The R GUI is pretty sparse. When you start R you will only see the R-console which does include a few drop-down menus for some useful commands and actions. Beyond this the GUI is fairly limited when it comes to doing actual work, writing programs, and maintaining your code.

This is quite OK. After all, R is really just a command line interpreter and not a text editor or full-featured application. So R is simply designed to interpret your inputs.[5]. It does not care where those inputs come from, how you entered them, or if you saved them. To make a long story short, you will NEVER, EVER, EVER want to input commands/code directly into the R-console. Chances are that R will crash, the power goes out, or you close your R-session without saving, and all your precious code, and computations are gone - FOREVER.

To avoid endless frustration and to maintain good mental health, ALWAYS, ALWAYS, ALWAYS write all you code in a text or script editor. Good science requires repeatability and the communication of knowledge. Anything you manually type into the R-console or the command line will be lost forever after you close the terminal/console. You won't be able to replicate anything, or send your code and hard work to anybody, for help, debugging, or sharing of results.[6] You will not do science and just draw lines in the sand. So again R does not care which text editor you use. All R wants is interpretable input. There are a gazillion editor options that will do the trick and allow you to write your code and feed it to R. Below are some options and my two cents about them. I apologize for the emphasis on software for Windows.[7]

[3]On Windows, adding R to your `path` is advisable, especially if you do not want to use the R-console and instead spawn an R-session from the Windows Command Prompt, etc ...

[4]Your packages are located in the `/library` directory. For example `C:/Programs/R-2.1.5/library/`.

[5]You can do this at the command line, e.g. in the Windows Command Prompt, Apple's Terminal App, etc ...

[6]Imagine typing a paper straight into the command line to have LaTeX compile it to a `.pdf`. Once you close the terminal, you won't ever be able to edit or change anything.

[7]Did I mention that I hate everything Apple?

1. **Notepad**: It works. You can save your code. Nothing else.

   → Verdict: Do not use.

2. **R Editor**: The R Editor will automatically get installed with R. It is a marginal improvement over Notpad. It runs on all platforms. You can save your work. It allows you to send code directly to the R-console for interpretation (via keyboard shortcuts).

   → Verdict: Do not use.

3. **WinEdt**: This is a full-featured text editor. It comes with R integration via an R-package called RWinEdt. You can send code directly to R. It has syntax highlighting and a few other useful features. It is ugly. Think Windows 95. It costs money. It only runs on Windows. Duh. I will show you later on in this tutorial how to install and use R-packages.

   → Verdict: Do try. If it works for you, great.

4. **Notepadd++**: This is a really nice lightweight editor for Windows. It is free and open-source. It is highly customizable, theme-able, and good looking. Has auto-completion, function-completion, and function-folding capabilities. Multi-language support. It can be downloaded here: http://notepad-plus-plus.org/. Via a the NppToR plug-in you can also send code straight to R. It also allows PuTTy integration for passing to an R instance on a remote computer. NppToR can be downloaded here: http://sourceforge.net/projects/npptor/.

   → Verdict: I think this is a great choice if you are looking for a free, easy to use, and versatile editor for Windows.

5. **Emacs**: This is probably the editor of choice if you're some type of hardcore programmer, hacker, Linux-geek, or nerd. It is cross-platform. And does anything any text editor could ever be asked to do and more. There exist plug-ins making your R programming much easier (e.g. ESS: Emacs Speaks Statistics) but unless you have a ton of time on your hand or are crazy, you should stay away from Emacs. The learning curve is just as high as it is for R itself and we are talking about a text editor, here. I won't provide you with links.

   → If you are not using it already, stay away. You have better things to do.

6. **RStudio**: RStudio is not a text editor but a free and cross-platform IDE or software application that provides comprehensive facilities especially designed for R programmers.[8] It is extremely powerful and many of your colleagues here in the department swear by it. It can be downloaded here: http://www.rstudio.com/. You should definitely check out the "screencast" which lays out RStudio's capabilities: http://www.rstudio.com/ide/.

   → Verdict: If you don't have to do things your way or you like the look and feel of Stata, this is probably the best choice! We will not use an IDE for the purposes of this tutorial.

7. **Sublime Text 2**: This is by far the best looking text editor in this list. It is cross-platform but will cost you $50 for a license to disable rare pop-ups of the evaluation version. It is more versatile and has more features than say Notepad++. It has all the usual features of any good text editor, plus some nice stuff others do not. It is not specific to R. So you can write and compile your LaTeX documents right here. Then spawn an R-session right in the editor (via

[8]An alternative, not specific to R is Eclipse IDE found here: http://www.eclipse.org/

the SublimeREPL plug-in) while working on your Python or C++ code, etc. Think of it as the Emacs for dummies. You can get the editor here: http://www.sublimetext.com/. The SublimeREPL plug-in can be found here: https://github.com/wuub/SublimeREPL.

→ Verdict: Setting it up and customizing it to your needs will take some time (e.g.installing the package manager and the R and LaTeX plug-ins, etc). Once you have it running, it is the last editor you will ever use.

1.2.5 References & Help

You are not the first folks to learn R. Any problem you will encounter and any question you may have related to R over the next year or so will have been encountered or asked by somebody else.That's a good thing. It means that you will be able to get help. There exist many tutorials, papers, and books on how to use R and how to apply R to all sorts of problems. We will cover how to use and access R's internal help and reference features later in this tutorial. Below is a list of some reference materials I have found useful.

On the Web

1. Google it. In 99.9% of the cases you will find a solution to an issue or answers on how to do something with a quick Google search.
2. The official R manuals: Go to http://www.r-project.org/ and click on Manuals under the Documentation heading. You will find introductory materials and advanced tutorials on how to create your own packages.
3. *An Introduction to R* by Venables and Smith (2012) can be downloaded here: http://cran.r-project.org/doc/manuals/r-release/R-intro.pdf.
4. *simpleR* by Verzani (2002) can be downloaded here: http://www.math.csi.cuny.edu/Statistics/R/simpleR/printable/simpleR.pdf.
5. The RGraph Gallery is a great source for help and code for making graphs: http://gallery.r-enthusiasts.com/.

Books

1. *The R Book* by Crawley (2007) should be available in THE STAR LAB.
2. *ggplot2: Elegant Graphics for Data Analysis* by Wickham (2009) is an excellent source for getting started with ggplot.

1.3 Almost there

Actually, one last thing before we get started. It is important to note that R is a compiled or interpreted language. This means that all commands and code you give R (either by sending code via an editor or by typing it directly into the console) will be executed immediately. Unlike in uncompiled languages (C, Fortran, etc, . . .) you are not required to build a complete program and instead R will intepret things for you. Throughout the tutorial you will see examples of R-input and R-output like so:

```
1 > print("Hello world!")
  [1] "Hello world!"
```

When we reach these examples, you should input the R-code from the numbered lines into the R-console at your workstation. For the above example you should have typed: `print(''Hello world!'')` and then pressed enter. Whenever the output R spits back at you does not match what's in this tutorial, let me know and we shall hopefully figure things out.