

Laboratory 1: Getting Started with R

Introduction

As we discovered this morning, this class is comprised of students with a variety of mathematics and computing skills. Some of you took calculus too many years ago to remember while a few of you have completed courses in advanced statistics. The same pattern holds for your comfort and skill with computers. Today's laboratory is designed to get you started with scientific programming using R. The specific objectives are to:

1. Introduce scientific programming along with some tips and tricks;
2. Develop programming skills with R.

While you are working on the laboratory, I would like for you to keep track of your activities in the format of a brief research journal entry. This entry should be an informal record of the questions you ask, the actions you take, and the discoveries you make. Please include (and clearly label) your answers to the following questions asked in the laboratory:

- Exercises 1.1 – 1.4
- Exercises 3.1 – 3.4
- Exercises 4.1 – 4.5.

These may be hand written or typed and are due by class on August 26th. Typed reports may be emailed to me at borretts@uncw.edu.

If you email me your laboratory work (which I highly encourage), please save the file with the following format "bio534-yourlastname-lab#.doc", where you replaces "yourlastname" with your last name and "lab#" with the appropriate laboratory number. As this is laboratory 1, I would name my file "bio534-borrett-lab1.doc". Also, **please include bio534 in the subject line** of all electronic correspondence with me.

Scientific Computing with R

Ellner and Guckenheimer (2006) describe R as follows:

- "R is an object-oriented scripting language that combines
- the programming language S developed by John Chambers (Chambers and Hastie 1988, Chambers 1998).
 - a user interface with a few basic menus and extensive help facilities.
 - an enormous set of functions for classical and modern statistical data analysis and modeling.
 - graphics functions for visualizing data and model output."

We will use R in this class because it is powerful, rapidly developing, and relatively easy to use. In addition, there are multiple resources online to help you get started. I list a few of the more useful online references below.

Helpful Resources

R Homepage

<http://www.r-project.org/>

Reference Card

<http://cran.r-project.org/doc/contrib/Short-refcard.pdf>

Kickstarting R

<http://cran.r-project.org/doc/contrib/Lemon-kickstart/index.html>

Dynamic Models with R <http://www.cam.cornell.edu/~dmb/DynamicModelsLabsInR.pdf>
Getting Started with R <http://cran.r-project.org/doc/manuals/R-intro.pdf>

You may also find <http://greenteapress.com/thinkpython/html/book002.html> useful for general tips on learning to program.

As R is a FREE environment, you can download and install it on your personal computers. It is available from <http://lib.stat.cmu.edu/R/CRAN/> and will run on Windows, Mac, and Linux operating systems.

Laboratory Exercises

Download and work through Sections 1-4 of Ellner and Guckenheimer's (2006) "An introduction to R for dynamic models in biology" (#4 in the list above) and complete the embedded exercises. Record the answers in your journal entry. You can copy and paste code directly from the R editor into a Word document, and all figures can be saved as a graphic file that can then be imported into Word.

All files mentioned in the laboratory can be found online at <http://www.cam.cornell.edu/~dmb/Rfiles/>. Several scripts will need modifications to work, which we can discuss in class.

While you are only required to turn in your work for sections 1-4, I highly encourage you to also work through sections 5, 6 and 7 on your own. You will need the techniques in these sections as the course progresses. Also, pay close attention to the exercises and check your work as R may not always do what you expect it to. For example, Exercise 1.4 is tricky.

Text Editors

A good text editor is an essential tool for programming, scientific computing, and general informatics work (see http://en.wikipedia.org/wiki/Text_editor). A text editor is used for editing plain text files, which is the format of all uncompiled programs including R scripts and functions. Notepad is an example of a basic text editor that comes with Microsoft Windows. Microsoft Word is not a text editor. Instead it is a word processing program that embeds hidden code into your documents. This code is important for word processing and page layout, but it renders Word useless as a plain text editor. Programs written in Word will generally not work.

As with all software, text editors come in multiple flavors. Some are barebones tools like Microsoft Notepad, others such as Vi and Emacs are more like Swiss army knives. The R program now comes with a general text editor to support writing R scripts. It is fairly simple to use, but (1) it is not very powerful, and (2) if you need to program for SAS or Matlab or any other language you will need a different text editor. One alternative option is a program called Tinn-R (<http://www.sciviews.org/Tinn-R/>), which is a more powerful text editor, but still fairly specific to R and only runs on Windows. I prefer using Emacs because it runs on multiple platforms, it can be modified to accommodate many programming languages, it is open source and free. The downside of Emacs is that it is challenging to learn because it is one of the oldest text editors around; however, once you learn it, it is a very, very powerful tool. Here is a link to a Windows version of Emacs that has already been partially customized to work with R (<http://vgoulet.act.ulaval.ca/en/ressources/emacs/>). Rather than enter into the text editor wars, you may choose which text editor you want to learn and use for this class.