

Demography 213

Simulating population: Arrays, for() loops, matrix multiplication and some plotting.

Carl Mason
carlm@demog.berkeley.edu

September 12, 2011

Abstract

This week’s adventure consists of a small population simulation, and a couple of simple yet profound questions about the workings of R and population dynamics.

Contents

1	Introduction	1
2	Exercise	2
2.1	Preliminaries	2
2.1.1	Memorize the following Emacs keystrokes	2
2.1.2	Memorize the following Unix commands	3
2.1.3	Remember the following special or “meta” character:	3
2.1.4	Create a subdirectory for this week’s project	3
2.2	Import the demonstration program into your Emacs buffer	3
2.2.1	Note the comments in the program	4
2.3	Work through the program in the buffer and ANSWER ALL THE QUESTIONS you find there	4
2.3.1	Running an entire file of R code	5

1 Introduction

The purpose of this exercise is gain some greater facility with the Emacs+R environment while at the same time pursuing a profound demographic truth – and picking up some graphics tricks. The `demonstration.r`

file this week consists of some code that will create a classic kind of demographic simulation and then draw some graphs. You will modify the code a little bit to improve the graphs and to learn something about how *Leslie Matrix* population projections behave.

While a certain amount of confusion is to be expected, everything will be much less confusing if **before Wednesday**, you

1. Read the rest of this document.
2. (re)read **Chapters 2 and 3 and the first half of Chapter 5** of *Introduction to R*. (You can find it on the course website <http://www.demog.berkeley.edu/Refs/R-intro.pdf>).

2 Exercise

2.1 Preliminaries

2.1.1 Memorize the following Emacs keystrokes

No doubt you know them already.

1. $\boxed{\text{C-k}}$ \rightarrow $\left\| \begin{array}{l} \text{erase (kill) everything from the} \\ \text{cursor to the end-of-line charac-} \\ \text{ter (another C-k kills the end-of-} \\ \text{line character)} \end{array} \right.$
2. $\boxed{\text{C-w}}$ \rightarrow $\left\| \begin{array}{l} \text{erase (kill) the currently selected} \\ \text{region} \end{array} \right.$
3. $\boxed{\text{M-w}}$ \rightarrow $\left\| \begin{array}{l} \text{put the current region onto the} \\ \text{clip board without killing it} \end{array} \right.$
4. $\boxed{\text{C-y}}$ \rightarrow $\left\| \begin{array}{l} \text{past whatever was last killed (or} \\ \text{placed on the clip board by way} \\ \text{of M-w) at the cursor} \end{array} \right.$
5. $\boxed{\text{C-x a}}$ \rightarrow $\left\| \begin{array}{l} \text{toggle on and off auto-fill-mode.} \\ \text{(affects how line breaks are in-} \\ \text{serted)} \end{array} \right.$

6. C-SPACE → anchor the selected region at the cursor. This is a mouse-free way of selecting a region. Put the cursor at one corner of the region you wish to select; C-SPACE; navigate to the opposite corner using arrow keys or C-e,M-e,C-a,M-a,C->, C-<, arrow keys or whatever. The highlighted region is the region between the current cursor location and the anchor point most previously selected

2.1.2 Memorize the following Unix commands

You know where to look them up right?

1. `rm filename` remove (erase) filename
2. `mv filename1 filename2` move (rename) filename1 to filename2 works with directory names too.

2.1.3 Remember the following special or “meta” character:

`~` substitutes for “home directory”. `cd ~` changes the current directory (or moves you) to your home directory. `cd ~carlm/213` moves you to the 213 subdirectory of carlm’s home directory.

2.1.4 Create a subdirectory for this week’s project

How about `~/213/Week3` ?

Cd into that directory and launch Emacs so as to create a new empty buffer called `ex3.r`. Verify that Emacs knows that you are thinking of writing some R code.

2.2 Import the demonstration program into your Emacs buffer

Turn your attention now to your empty Emacs buffer.

In order to dispense with a whole lot of typing, we are going to import the R program from my home directory. Use the now familiar `C-x` to import the contents of `~carlm/213/LeslieSimulation/demonstration.r` into your `ex3.r` buffer. If you want to minimize key strokes, `C-a` and `C-k` and TAB can help you.

You should now see a whole lot of neat R code in your buffer. Note that not all characters are the same color.

2.2.1 Note the comments in the program

Comments are important. Near the top of the file, there is a discussion of how to use comments for purposes other than to help you figure out later what you did. Also note the business about `auto-fill-mode`.

In order to further enhance your comments, here are two kind of useful tricks. Please sprinkle them about in your file to suit your tastes.

How to create a decorative row of say 70 #'s

`M-70 #` `enter` → `###...`

Note that `C-u 70` is the way this command would be written in the tutorial. The “meta” is not standard on all keyboards, but on **most keyboards** the `Alt` key works.

How to add the current date to the buffer

`## M-1 M-! date` `enter` → `## Mon Aug 31 08:52:18 PDT 2009`

2.3 Work through the program in the buffer and ANSWER ALL THE QUESTIONS you find there

The file that you read into your Emacs buffer contains several questions all of which are written as valid R comments. ANSWER each question **in your Emacs buffer** make sure that your answer is also written as a valid comment so that *everything in the buffer is either a comment or a valid and intentional R statement* this is very important because later you will have to execute your `buffer(file)` as a batch job.

Below are some things that will help you run the program and figure out what's going on:

2.3.1 Running an entire file of R code

There will come a time towards the end of this project when you will find it useful to execute all of the R code in a file – probably a small file that you will have created to run a few commands that you developed today.

There are several ways to do this, two that are likely to work well for today’s project are:

1. Emacs’s C-c C-b – this simply send the buffer to the R process running in the other window. there is an menu icon that screams “load file” when ESS is in effect. That does the same thing.
2. R’s `source('filename')` command. This causes R to pretend that you just typed everything that is in the file “filename” into the R process. All of the objects in your work space are accessible. Note that “filename” should be in quotes **unless filename** is a object that contains a character string that is the name of a the file that you want to read.

Note that `source()` reads the file from the disk whereas C-c C-b processes the contents of the **buffer**. If you make a change in the buffer you have to save it to the file before `source()` can read it. After saving the buffer, both the buffer and the file (with the same name) have the same contents.

When you are ready to go home, remember to:

- **save the work in your ex3.r buffer**
- quit R with the `q()` command
- **ALWAYS** decline to “save your workspace” in R.