

Demography 213

Week 1: The "12" most important Unix commands, Emacs and an introduction to R

Carl Mason
carlm@demog.berkeley.edu

September 9, 2011

Abstract

This week's exercise consists of logging into the Unix system; becoming familiar with the *terminal window* application and with the *gnome desktop* in general; using the "12" most important Unix commands to set up a few directories; and experiencing R within Emacs. R within Emacs is a very efficient way to do science, although the setup costs are not trivial, it is well worth the effort.

- The class meets Mondays 2-3 in the seminar room and Wednesdays 1-3 in the basement lab. Occasionally there will be exceptions when Monday holidays screw things up.
- Demography 213 is a **required** pass/not pass course. To get a degree in demography you must pass this class; to pass this class you must complete **all** assignments.
- During the first 5 weeks, the class will consists of weekly assignments, which are handed out on Monday and due the following Monday before class.
- The final 9 weeks of class will be taken up with longer projects which will be done in groups of two or three and will include a class presentation.
- During the first 5 weeks of the course we will roughly follow *Introduction to R* by Venables and Smith <http://www.demog.berkeley.edu/Refs/R-intro.pdf>. For further reading, I like the book *R in Action* by Robert Kabacoff <http://www.manning.com/kabacoff>. It is not essential that you purchase this book, but if you find the course baffling, this book might really help.

Contents

1	Unix/GNOME basics	3
1.1	Logging in and arranging the desktop	3
1.2	Terminal window ?	3
1.3	Editors	4
1.4	emacs	5
2	Assignment	6
2.1	Creating a directory structure with terminal commands . . .	6
2.2	Emacs	7

1 Unix/GNOME basics

1.1 Logging in and arranging the desktop

If you have not already done so, now would be a good time to login to a Linux workstation.

Upon logging in, you should be presented with a graphic user interface – also called a “desktop”. The linux desktop “looks and feels” like any other modern computer and should take very little getting used to. (There are some very significant differences between the Unix windowing system and those of other operating systems – but you’ll catch on to those differences as they become important to you) One difference, that will become evident shortly is a greater reliance on typed commands as opposed to pointing and clicking. As you will soon discover, the Unix terminal is much more sophisticated than the command line under windows – If it weren’t, we would not use it.

Some typographical conventions will make things easier to understand. In what follows, “@:>” will be used to represent the *Unix prompt*. The Unix prompt appears in a terminal window to indicate that the command interpreter is ready. When you are instructed to type:

```
@:> ls -l
```

you do not type @:> –it is there to indicate that what follows is to be typed at the Unix prompt—generally, inside a *terminal window*. In real life the Unix prompt may not look exactly like @:> . If you were me, typing this handout, it would look like this: [carlm@twins Week1]\$.

1.2 Terminal window ?

Terminal Windows also known as *xterm windows* are windows in which the Unix *shell* is running. The shell, or command interpreter is the part of Unix that presents the command prompt, @:> , reads a command you type and then does what *it thinks you wanted*.

Opening terminal windows Since you will want to open terminal windows fairly frequently, it would probably make sense to create a *launcher* on your *panel*. Here’s how to do it:

Applications →System Tools →(drag to panel) Terminal
which is a typographical convention meaning:

1. click on the Applications menu

2. click on the System Tools
3. click **and hold** the Terminal item and drag it to the panel (the gray bar at the top of the screen that holds things like this)– then release the mouse button.

Now you can launch a terminal window process by simply clicking on the launcher in the panel. When the mouse pauses over the Terminal icon it will shout: Terminal: Use the Command Line.

Of course you will still be able launch terminal windows from the menu.

If you do not find Terminal under Accessories, try System Tools.

In your copious free time, you will probably want to modify some of the features of you *terminal window* application.

When it first comes up there may be a menu bar from which you can select Edit→Current Profile. if not, hold down the `RIGHT BUTTON`, somewhere over the main part of the terminal windows (not on the frame) and a dialog box will appear – at the bottom is a “show Menubar” check box. After you change things to suit your preferences, check that box to make the menubar go away – you can use that space to do science.

- Pick a better color scheme: A reasonable color choice is Black on Light Yellow but suit yourself.
- change the font size: The default font is kind of small for people over 30–or people who read alot as children. To increase the font size hit `ctrl` + `+` to increase the font size in the current terminal window. NOTE `+` implies `shift`. To shrink the font, hit `ctrl` + `-` (no `Shift` this time)

It is from within a terminal window that you generally perform the 12 most important Unix commands. These commands are described in the *The “twelve” most important Unix commands* lab.demog.berkeley.edu/12important (hereafter, 12UC) . You will need to use a few of them to complete this weeks assignment.

1.3 Editors

Emacs is the premier editor for scientific programming. Editors are at first blush similar to word processors – in that they allow you to put text on the screen – but there are huge differences. Word processors are very poor

substitutes for text editors. Text editors are for creating *plain text files*¹—generally containing programs for compilation or a series of commands for some application. This is very different from documents intended to be read by humans.

Although emacs is not hard to learn, it does actually require *learning*—you cannot simply launch it and expect to use emacs productively. It will take patience, experimentation and perhaps a little hair loss to make it work for you. But the pay off justifies the effort. Emacs is not only the best editor for R code, it is also the best editor for all sorts of programming languages and statistical applications as well as LaTeX (a document preparation system that is particularly well suited to science).

To be fair, the payoff for using emacs is significant, but since it is not immediate, those with high discount rates or short life expectancies may prefer other editors.

Explanations and demonstrations in this course will assume that you are using emacs. If you choose to use something else, the instructor will probably make fun of you – but don’t let this stop you from doing what works for you. Other text editors are:

pico is the simplest editor there is. It has only about 6 commands so you can learn it in no time. But it is not aware of R and will offer you no help in this regard. This is the appropriate choice if you have very little time, as would be the case for example if you were currently under hospice care.

gedit has macro capabilities, fancy search routines, and some awareness of the type of file you are editing. Smart people have been known to use this editor. I don’t use it so I can’t say much more.

vi, vim is, like emacs, as much a religion as an editor. It does everything you could possibly care about. Use it if you already know it. If you don’t, then learn emacs so you can be like the other kids.

1.4 emacs

To launch emacs , you can either find it on a menu Applications→Accessories (an opportunity to create another launcher) or from a terminal window, type the command:

¹“plain text” or “ASCII” files use a character encoding system that contains no formatting – just the roman alphabet. ASCII is the kind of file that software programs such as R like to read

| @:> emacs <&>

Note the **optional** “&” in the command above. The <brackets> indicate an optional *flag* or *argument* you do **NOT** type the <> – just what’s inside them – and only if you want to. In this case, typing the & will give you back the prompt and allow you to do other important things in that terminal window. Without the &, the terminal window will be tied up until you quit emacs. Terminal windows are cheap, so this does not matter much.

2 Assignment

Now that you are somewhat comfortable with *the idea* of using Emacs and the terminal window in a Unix environment, it’s time to actually do it. This week’s assignment is to create some directories for current and subsequent work in this class and then to launch Emacs and R and to work through the R commands in the `demonstration.r` file – which you will import rather than type.

2.1 Creating a directory structure with terminal commands

In order to do this and subsequent assignments, you must create a few directories and subdirectories. This is necessary both so that I can find your work and also so that you can figure out what you did when you come back to it in order to reproduce it or to (re)learn from it.

In subsequent weeks, directory structures will become more complicated in order to allow you to work with other people and to move your project back and forth between your own machine and the Demography Lab servers.

Setting up clever directory structures can make life much easier.

Look up their meanings and MEMORIZE these Unix commands

Obviously, 12UC (lab.demog.berkeley.edu/12important) is the place to go to find explanation of the 12 most important Unix commands.

1. `mkdir` (make directory)
2. `cd` (change current directory)
3. `ls` (list files in 732 different ways)
4. `chmod` (change permission “modes” of file/directory)

Open terminal window by whatever means you prefer.

Create a subdirectory of your home directory called 213.

Verify that 213 is publicly readable – i.e. readable by all users. If it is not, make it so. (Did you look up `chmod` ?) Ponder for a moment what it means that your work is publicly readable.

make 213 your current directory that is `cd` into 213.

create a subdirectory of 213 called “Week1” and make that your current working directory. Note that the current working directory concept applies to the terminal window and to processes (such as Emacs and R) that are launched from it. Other terminal windows and processes have their own current working directory.

The current working directory is important because when you read or write files from within programs such as R, you specify a *path* or filename, how that path is interpreted generally depends on the current working directory.

2.2 Emacs

You have now created a new directory for the class (213) and underneath that, a subdirectory for this week’s exercise. Our next task is to create and edit a file of R commands that do something... something that if not actually useful at least suggests how R might be useful.

Print out the official Free Software Foundation Emacs Reference Card : http://www.demog.berkeley.edu/213/emacs_refcard.pdf, and keep it with you until the paper deteriorates. There is information on printing in 12UC, but all you probably need to know for now is that the printer **age** is the one in the basement lab.

Memorize these 5 key combinations

1. `Ctrl` +g (back out of whatever emacs command you are beginning to type) – this is by far the most important Emacs command. Hit C-g whenever something crazy happens. It cannot hurt you.

2. `Ctrl` +x u (undo the previous action. Keep hitting it to undo more previous commands)
3. `Ctrl` +x i (insert (or “import”) the contents of a file into the current buffer at the present location.
4. `Esc` < (move to the beginning of the buffer)
5. `Esc` > (move to the END of the buffer)
6. `Ctrl` +a (move to beginning of line: mnemonic “a” is the first letter of the alphabet)
7. `Ctrl` +e (move to the end of the current line)

Read Chapter 2 of *Introduction to R* Reading it first is best. At least have it handy as you plow through the demonstration file below.

Launch emacs in such a way as to create a buffer which when saved will be a file in your `~/213/Week1` directory – called `213ex1.R`. That is the file that will be created when the buffer is saved will be `~/213/Week1/213ex1.R`. The most straight forward way of doing this is by:

1. making sure that your current directory, when you launch Emacs, is `~/213/Week1`.
2. specifying the buffer name “213ex1.r” as the argument when you launch Emacs:

```
| @:> emacs 213ex1.r &
```

import the demonstration.r file You *could* download the `demonstration.r` file from the web, but you will save keystrokes by importing in directly from where it lives on our network: `~carlm/213/Week1/demonstration.r`. There is a key combination for inserting/importing files into Emacs, but you can also use the mouse and menu.

Follow the directions in the demonstration.r file If you have successfully imported the demonstration.r then you should be able to do what it asks. Since Emacs is very confusing the first time you use it, saving your work often will minimize hair loss.

The file will instruct you to quit and restart a few times. It is very worthwhile to get familiar with the procedure of launching Emacs and R. Do it enough times so that you can comfortably stop and resume your project without looking at notes on how to do it.

Save and exit When you are finished Save this buffer – thus re(writing) a file called `~/213/Week1/213ex1.r`. It is best form to exit from R *before* exiting from Emacs, you will be warned if you don't.

send email to carlm—from your @demog.berkeley.edu account indicating how much you enjoyed this assignment. A quick, easy and useful way to do this is with `alpine`. Alpine is an easy and powerful (although seemingly primitive) email agent which by default it configured to use your @demog.berkeley.edu address. It is handy to know how to send mail from this account even if you have your @demog.berkeley.edu email forwarded elsewhere.