Stat243: Problem Set 1, Due Monday Sep. 16

September 6, 2013

This covers UNIX and the bash shell as well as practice with some of the tools we'll use in the course (Git, knitr/Sweave/R Markdown).

It's due at the start of class on 9/16.

Please note my comments in the syllabus about when to ask for help and about working together.

Formatting requirements

As discussed in the syllabus, please turn in (1) a copy on paper, as this makes it easier for us to handle AND (2) an electronic copy through bSpace so we can run your code if needed.

Your electronic solution should be in the form of a plain text file, with the shell code included. Ideally (but not required for this first problem set), your file would be a Latex or R Markdown file, with the shell code in chunks demarcated using one of the following types of syntax:

- Latex with knitr:

```
%% begin.rcode chunkName, engine='bash'
% # try a basic shell command
% ls
%% end.rcode
```

- Sweave:

```
<<chunkName, engine='bash'>>=
# try a basic shell command:
ls
@
```

- R Markdown: ```` ```{r, engine='bash'} your bash code here ``` ````

If you do choose to use Latex/R Markdown for the entire problem set, see the hints in Problem 5 for how to create a PDF or HTML file using the *knitr* package in R.

For problem 5, your solution should be in the form of Latex or Markdown with embedded R. Statistics students should use Latex.

For problems 3 and 4, your solution should start with a brief textual description of how you solved the problem, with the code following, including description of what your code does interspersed with the code. Do not just give us raw code.

Technical requirements for your solutions to Problems 3 and 4

All of your operations should be done using UNIX tools (i.e., you are not allowed to read the data into R or Python or other tools). Also, ALL of your work should be done using shell commands that you save in a file and turn in as part of your solution to the assignment. So you can't say "I downloaded the data from such-and-such website" or "I unzipped the file"; you need to give me the code that I could run to repeat what you did. This is partly for practice in writing shell code and partly to enforce the idea that your work should be replicable and documented. Any downloading and processing of FAO metadata for part (c) of Problem 4 should be done by script as well.

Problems

1. (Due Sep. 3) Please fill out the class entry survey:

   https://docs.google.com/a/berkeley.edu/forms/d/1RqoEgfR92PPlW1JltS3Hu0T4Skz27akaviI_1iaxMMI/viewform

2. Git practice:

   (a) Clone the class Git repository either onto your own machine or into a directory in your SCF home directory. The former may require that you install Git on your machine.

   (b) Create a local repository on your machine or in your SCF home directory. Use this repository as you prepare this problem set, making sure to make at least a few commits as you work your way through the problem set. Your answer to this problem should simply reporting your "git log" result (or the log information for the last 10 commits or so, if you have many commits).

3. This problem provides practice in downloading and manipulating data using shell scripting. We'll use United Nations Food and Agriculture Organization (FAO) data on agricultural production. If you go to http://data.un.org/Explorer.aspx?d=FAO and click on "Crops", you'll see a bunch of agricultural products with "View data" links. Click on "apricots" as an example and you'll see a "Download" button that allows you to download a CSV of the data. I've deconstructed the javascript on the site a little bit and found out that you can download a file directly via URL in the following format:
   `http://data.un.org/Handlers/DownloadHandler.ashx?DataFilter=itemCode:526&DataMartId=FAO&Format=csv&s=countryName:asc,elementCode:asc,year:desc&c=2,3,4,5,6,7&`
   That downloads the data for Item 526 (apricots). Note that you may need to put the http address inside double quotes when using a UNIX shell command to download it. Also make sure there are no carriage returns in the http address (I had to break the address above to fit on the page). You can see the item ID for other products by hovering over "View Data" link for the relevant product.

   (a) Download the data for apricots. Extract the data for individual countries into one file and for regions of the world into another file. Then subset the country-level data to the year 2005. Based on the "area harvested" determine the five countries using the most land to produce apricots. You'll need to look carefully at arguments to the *sort* utility, in particular the "key" will allow you to sort on a field that is not the first field. Now automate your analysis and examine the top five countries for 1965, 1975, 1985, 1995, and 2005. Have the rankings changed?
   Tip: you can use the following code, which uses a utility called *sed* (used for pattern matching and replacement) to deal with commas and quotes in the data file. I didn't go into *sed* in Unit 2 because we'll deal with regular expressions primarily through R. The first line removes the commas in any country names so that commas are used only for field delimiters. The second

line gets rid of the double quotes so that we can sort by fields numericall.

```
sed 's/, / /g' file1.csv > file2.csv
sed 's/\"//g' file2.csv > file3.csv
```

(b) Write a bash function that takes as input a single item code (e.g., 526 for apricots, 572 for avocados) and prints out to the screen the data stored in the CSV file, such that the information could be piped on to UNIX another operation.

(c) (Extra credit) Find the FAO metadata online that relate item codes to names of agricultural products. Modify your function in (b) so that the user can pass the name of the product instead of the code. Your function would then presumably query a file (i.e., a file that you have created in a separate step) that relates the codes to the names.

The goal here is to practice writing shell code to download files, extract fields and subset datasets, and summarize information in those fields. If you'd like to use different fields or different data available online to answer a question of your choice, that's fine. Just run the data source and question(s) by me in advance.

4. More shell practice:

(a) Your task here is to automatically download all the files ending in *.txt* from this National Climate Data Center website: http://www1.ncdc.noaa.gov/pub/data/ghcn/daily/. Your shell script should provide a status message to the user, telling you the name of the file as it downloads each file.

(b) (Extra credit) Embed the code in a bash shell function that takes two arguments, the name of the website from which you wish to download any hyperlinks in the website and the file name extension (e.g., we used *.txt* above) you wish to use in restricting your downloads to particular types of files. As files are downloaded, your function should give messages such as "Downloading file 7 of 22: precip.txt". Test that it works with the pdf files on Philip Stark's publications page: http://www.stat.berkeley.edu/∼stark/bio.htm#publications.
Hint: if you're not sure how to do the necessary arithmetic, you should be able to find a useful command on the class Wiki.

5. As preparation for future problem sets, this problem explores embedding R code and output in a PDF or HTML document. Here is some R code that creates a plot and prints some output to the screen.

```
hist(LakeHuron)

lowHi <- c(which.min(LakeHuron), which.max(LakeHuron))

yearExtrema <- attributes(LakeHuron)$tsp[1] - 1 + lowHi
```

You should produce a PDF (using the knitr package in R with Latex or Sweave with Latex - I suggest knitr as it is a more recent technology with a lot of powerful features) or an HTML file (using R Markdown). If you are a Statistics student you should use Latex.

Requirements for your solution:

(a) The document should embed the R code above and the output of that code within a description of the analysis.

(b) Your result should look like the last page of the PDF of this assignment (it does not have to be exactly the same in terms of formatting and the actual prefacing text), which I created using Latex and knitr.

(c) Your document should be less than a page and your figure should be small enough that it only takes up half the width of the page (the *fig.width* argument for Sweave or similar arguments may be helpful).

(d) In your solution, you should NOT manually type '1875' or '1972' in your document, rather embed an R expression that returns '1875' and '1972' using *\rinline, \Sexpr* or the equivalent for R Markdown.

Hints:

i. Using Latex with knitr: The following should work on an SCF Linux machine. See the example Latex file with embedded R code at the knitr demo page: 005-latex.Rtex. Rename the file, e.g., knitrTest.Rtex (for some reason, when I tried to use the *knit()* function below with the original file name, it failed). Start R and create an interim Latex file

```
> library(knitr)
> knit('knitrTest.Rtex') # this should produce knitrTest.tex
```

And you can then use *pdflatex* at the UNIX command line to compile the pure Latex file: `pdflatex knitrTest.tex` and create a PDF.

ii. Using Sweave: Your files should have the .Rnw extension. See the example Sweave file at the knitr demo page: 002-minimal.Rnw. Rename the file, e.g., sweaveTest.Rnw.

```
> library(knitr)
> knit('sweaveTest.Rnw') # this should produce sweaveTest.tex
```

And you can then use *pdflatex* at the UNIX command line to compile the pure Latex file: `pdflatex knitrTest.tex` and create a PDF.

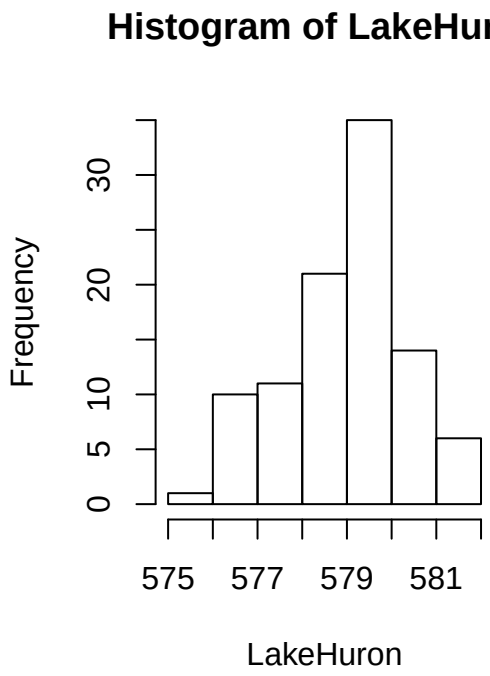
iii. To use R Markdown, see the demo file *module1_basics.Rmd* in the repository. You can create HTML from this file from within R as:

```
> library(knitr)
> knit('module1_basics.Rmd') # this should produce module1_basics.html.
```

Solution to Problem 5

The height of the water level in Lake Huron fluctuates over time. Here I 'analyze' the variation using R. I show a histogram of the lake levels for the period 1875 to 1972.

```
hist(LakeHuron)
```




```
lowHi <- c(which.min(LakeHuron), which.max(LakeHuron))
yearExtrema <- attributes(LakeHuron)$tsp[1] - 1 + lowHi
```