Stat243: Problem Set 1, Due Friday Sept. 12

September 2, 2014

Comments:

- This covers UNIX and the bash shell as well as practice with some of the tools we'll use in the course (VM, knitr/R Markdown).
- It's due at the start of class on 9/12.
- Please note my comments in the syllabus about when to ask for help and about working together.
- Finally, you should start early on this as the last three problems may take you some time. Later assignments for which you only have a week or a week and a half will not be as long.

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 Git following Jarrod's instructions.

Your electronic solution should be in the form of a plain text file named *ps1.txt*, with the shell code included. Ideally (but not required for this first problem set), your file would be a Latex (named *ps1.Rtex*) or R Markdown file (named *ps1.Rmd*), 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
```

- R Markdown:

````
```{r, engine='bash'}
ls
```
````

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 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 your solution file. So you can't say "I downloaded the data from such-and-such website" or "I unzipped the file"; you need to give us the code that we 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.

For Problem 3, to find the URL of a file you want to download, you can navigate on the FEC webpage and hover over a hyperlink with your mouse. This should allow you to see the URL as a text string. Your browser will probably allow you to right click on the link and 'copy the link location' as opposed to actually clicking the link. You should then be able to paste the URL into your code file.

You should not need to use the UNIX utilities *awk* or *sed* (apart from the one point I give you explicit syntax), though if you want to, you can.

Problems

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

   https://docs.google.com/a/berkeley.edu/forms/d/1Yr33EgFwcgtIgJ8FmhyugarGkGNmhnpS3QztknsK_Cs/viewform?

2. Virtual machine practice: Install VirtualBox on your computer and import the BCE virtual machine (VM), following these instructions: http://collaboratool.berkeley.edu/using-virtualbox.html. Then clone the class Git repository onto the virtual machine.

   (a) How many (virtual) processors are there on the virtual machine?

   (b) What is reported (from within the VM) in terms of how much RAM there is?

3. This problem provides practice in downloading and manipulating data using shell scripting. We'll use data from the US Federal Election Commission (FEC) on campaign expenditures and contributions about contributions to candidates for the US Senate. Not all of you will be familiar with the US election system or how candidates for office raise money, so if you're confused about the context, do ask me. Basically, individuals who are campaigning to be elected raise money as contributions from other people and organizations. The contributions must be reported to the FEC and the resulting data are available to the public.

   (a) This website, http://www.fec.gov/data/CandidateSummary.do?format=html, contains information on the total contributions and expenditures by each candidate for the 2014 elections. At the top of the page you'll see links to a CSV file and to metadata for all of the candidates running for federal office.

      i. Using the CSV file extract the data for Republican and Democratic candidates for Senate into two files (one for Republicans and one for Democrats). Be careful because S and REP and DEM can appear in fields other than the relevant fields.

      ii. At this stage, we want to be able to split into fields based on commas, but some of the fields themselves have commas in them. Here's how we can use a UNIX program called *sed* to do some useful replacement of certain patterns. You don't have to understand the syntax as working with regular expressions in *sed* is more than I want to cover in the course.

         - `sed 's/\([^",]\),/\1/g' file.csv` # this removes any comma that is NOT preceded by a " (double quote) or another comma

- `sed 's/[$"]//g' file.csv` # this removes the $ and " (double quote) from the file

Now extract the total contributions to each candidate and some basic information about each candidate and sort on the total contributions. 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. For each party find the five candidates with the largest total contributions and print the amount of contributions and some information about the candidate to the screen.

(b) For this part of the problem, we'll use FEC data on individual contributions to candidates from ftp://ftp.fec.gov/FEC/2014. The data are described at the website: http://www.fec.gov/finance/disclosure/ftpdet.shtml#a2013_2014. Basically the Candidate Master file contains a record for each candidate, which includes their "principal committee ID" for their campaign finance committee. Write shell code that takes a candidate's last name and returns their principal committee ID based on the *cn.txt* file (from the *cn14.zip* file), assigning the result to a shell variable. Now write shell code that uses that variable to extract the individual contributions to that candidate (from the individual contributions file, *itcont.txt* contained in the *indiv14.zip* file) and finds out the number of contributions above $200, both nationwide and restricted to California. Compare the two candidates for the Kentucky Senate seat, Mitch McConnell and Alison Lundergan Grimes.

(c) Extra credit: Take your code from part (b) and make it into a function that finds the number of contributions for a candidate, returning the result in a nicely formatted way. Now use a loop to loop through multiple predefined candidates and call the function, reporting the number and total contributions for each candidate.

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: 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. You should be able to use the UNIX tools in Table 2 of Unit 2 to extract the individual file names from the HTML index file linked to above. Do not hard code the names of the txt files into your code.

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 an HTML file (using R Markdown). If you are a Statistics student you should use Latex.

Requirements for your solution:

(a) Your solution should consist of the Latex+knitr or R Markdown syntax that produces the PDF, where the syntax embeds the necessary R code.

(b) Your resulting PDF or HTML output 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+knitr.

(c) Your resulting PDF or HTML 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 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* or the equivalent for R Markdown.

Hints:

(a) Using Latex with knitr: The following should work on the VM or an SCF Linux machine. However, on the VM, at the command line, you'll first need to run the following to install some necessary software:

```
sudo apt-get install texlive
```

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 run the following to produce a PDF:

```
library(knitr)
knit2pdf('knitrTest.Rtex') # this should produce knitrTest.pdf
```

On the VM, you can use the program *evince* to view the PDF.

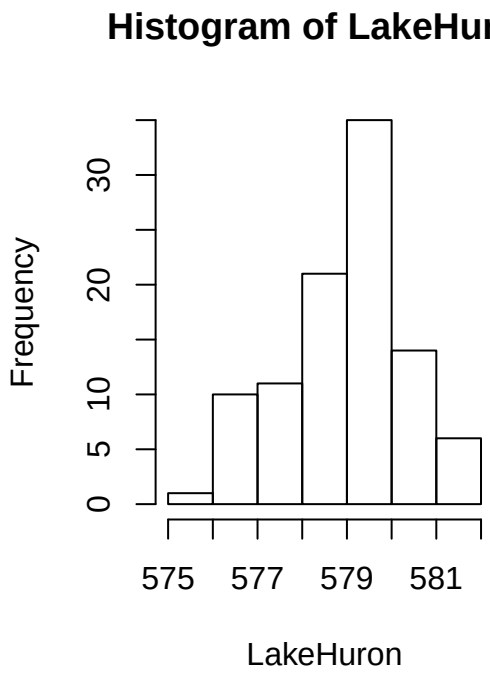
(b) To use R Markdown, see the demo file *module1_basics.Rmd* in the *berkeley-scf/r-bootcamp-2014* repository. You can create HTML from this file from within R as:

```
library(knitr)
knit2html('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
```