Stat243: Problem Set 1, Due Friday Sept. 7

August 27, 2018

Comments:

- This covers UNIX and the bash shell as well as practice with some of the tools we'll use in the course (knitr/R Markdown).
- It's due at 2 pm on September 7, both submitted as a PDF to bCourses as well as committed to your Github repository.
- 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 problems may take you some time. Particularly if you are getting used to bash, it may be hard to get everything to work at the last minute.

Formatting requirements

1. As discussed in the (to-be-revised) syllabus, please turn in (1) a PDF through bCourses, as this makes it easier for us to handle grading AND (2) an electronic copy through Git following Omid's instructions.

   Your electronic solution should be in the form of an R markdown file named *ps1.Rmd* or a LaTeX+knitr file named *ps1.Rtex*, with bash code chunks included in the file.

2. 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.

3. 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 except as noted). 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.

   You should not need to use *awk* or *sed*, but you may if you want to.

4. Formatting comments:

   (a) If the downloading time interferes with generating the PDF, you can use eval=FALSE in one or more of your code chunks and then just cut and paste in example output, but ideally you can just cut down the amount of downloading and get all the chunks to run during the PDF generation.

(b) If you have a lot of lines running off the end of the page you can use a "\", e.g.,

```
grep a file.csv | grep b | \
grep c
```

Problems

1. This question is the class survey that you already filled out.

2. As preparation for the next couple units, please read the sections titled "Memory hierarchy" and "Cache in depth" in the following brief piece that talks about the difference between the CPU cache, main memory (RAM) and disk. You don't need to follow all the technical details in the "Cache in depth" section.

3. A friend of mine is planning to get married in Death Valley National Park in March. She wants to hold it as late in March as possible but without having a high chance of a very hot day. This problem will automate the task of generating information about what day to hold the wedding on using data from https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/by_year/.

   (a) Download yearly climate data for a set of years of interest into a new (temporary) subdirectory within your working directory. Do not download all the years and feel free to focus on a small number of years to reduce the amount of data you need to download. Note that data for Death Valley is only present in the last few decades. As you are processing the files, report the number of observations in each year by printing the information to the screen, including if there are no observations for that year.

   (b) Subset to the station corresponding to Death Valley, to TMAX (maximum daily temperature), and to March, and put all the data into a single file. In subsetting to Death Valley, get the information programmatically from the *ghcnd-stations.txt* file one level up in the website. Do NOT type in the station ID code when you retrieve the Death Valley data from the yearly files.

   (c) Create an R chunk that makes a single plot of side-by-side boxplots containing the maximum daily temperature on each day in March.

   (d) Now generalize your code from parts (a) and (b). Write a shell function that takes as arguments a string for identifying the location, the weather variable of interest, and the time period, and returns the results. Your function should detect if the user provides the wrong number of arguments or a string that doesn't allow one to identify a single weather station and return a useful error message. It should also give useful help information if the user invokes the function as: "get_weather -h". Finally the function should remove the raw downloaded data files.
   Hint: to check for equality in an if statement, you generally need syntax like: `if [ "var" == "7" ]`.

4. Your task here is to automatically download all the files ending in *.txt* from this National Climate Data Center website: https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/. Your shell script should provide a status message to the user, telling the name of the file as it downloads each file. You should be able to use UNIX utilities to extract the individual file names from the HTML index file linked to above. Alternatively you may use tools from Unit 3, but any use of R should be done directly from the command line and should only involve a line or two of R code. Do not hard code the names of the .txt files into your code.
   Note that when we work on webscraping in Unit 3, we'll see more elegant ways to process HTML

than we are using here, but doing it "manually" here is good for practicing with the shell commands and allows us to do quick-and-dirty processing.

5. This problem is actually just the formatting of this document.

   (a) Have the document be correct formatted according to the requirements above. You don't need to explicitly answer this sub-problem.

   (b) (extra credit) The *reticulate* package and R Markdown allow you now to have Python and R chunks in a document that interact with each other. Demonstrate the ability to use this functionality, in particular sending data from R to Python and back to R, with some processing done in Python (it doesn't have to be complicated processing). There's a blog post here to get you started: http://feedproxy.google.com/~r/RBloggers/~3/76l-uQEOoiU/?utm_source=feedburner&utm_medium=email .