Stat243: Problem Set 2, Due Friday Sep. 18

September 11, 2015

This covers material in Unit 3 (and back to Unit 2).
It's due **on paper** and submitted via Github at the start of class on Sep. 18.
Some general guidelines on how to present your problem set solutions:

1. Please use your Rtex/Rnw/Rmd solution from PS1, problem 3 as your template for how to format your solutions (only non-Statistics students are allowed to use R Markdown).
2. Your solution should not just be code - you should have text describing how you approached the problem and what the various steps were.
3. Your paper submission should be the printout of the PDF produced from your Rtex/Rnw/Rmd file. Your Github submission should include the Rtex/Rnw/Rmd file, any R or bash code files containing chunks that you read into your Rtex/Rnw/Rmd file, and the final PDF.
4. Your code should have comments indicating what each function or block of code does, and for any lines of code or code constructs that may be hard to understand, a comment indicating what that code does. You do not need to show exhaustive output but in general you should show short examples of what your code does to demonstrate its functionality.
5. Use functions as much as possible, in particular for any repeated tasks. We will grade in part based on the modularity of your code and your use of functions.
6. Please note my comments in the syllabus about when to ask for help and about working together.
7. Please give the names of any other students that you worked with on the problem set.

Problem

The purpose of this PS is to give you practice with manipulating large files and reading data into R, as well as operating programmatically via scripting.

1. The file *www.stat.berkeley.edu/share/paciorek/ss13hus.csv.bz2* is a zipped file (500 Mb zipped) containing household-specific data from the 2009-2013 US Census American Community Survey. This survey obtains a wealth of information on people and households every year, with about 1% of the population surveyed in each year. The data dictionary describing the fields is available at http://www2.census.gov/programs-surveys/acs/tech_docs/pums/data_dict/PUMS_Data_Dictionary_2009-2013.pdf. Note that the only non-numeric field is the RT field.

    (a) Your task is to take a random sample of 10,000 households and create a CSV file containing the following fields for those households: "ST", "NP", "BDSP", "BLD", "RMSP", "TEN", "FINCP",

"FPARC", "HHL", "NOC", "MV", "VEH", "YBL". The fields in your CSV should have the same names as the original data file.

(b) Explore whether *read.csv()* or either (your choice) of *readLines()* or *scan()* is faster. You might also (not required) explore whether skipping rows (not reading them into R) that are not part of your random sample speeds things up. You can use *system.time()* to time an operation, e.g., `system.time(x <- rnorm(10000))`.
I've found *read_csv()* from *readr* to be slower than *read.csv()* and to crash R in some cases. I may give extra credit if you figure out a way to get *read_csv()* to perform well or figure out why it is not performing well.

(c) Is there any pre-processing in bash that you can do to speed up the overall workflow?

(d) Finally, do a cross-tabulation or two to see if you see any interesting associations between any of the variables. Doing a quick analysis is a basic test that the data look reasonable and your processing code worked. In fact in preparing the solutions, my solution for this subquestion made me realize there was a bug in my code in part (a) that I was not extracting the correct columns in the correct order.

Some rules for your solution:

- All operations must be scripted. You cannot say "I downloaded the file from XYZ" or "I counted the number of fields, and found FINCP to be the 17th field and NOC to be the 13th and ..." (to put it another way, the input to your code should be of the form "ST", "NP", etc. as the column names of interest and not of the form 7, 13, ... as the column numbers of interest). I'm expecting a mix of bash and R code.
- You may not ever explicitly unzip the entire file. See the help info for bzip2 for how to be able to take a look at a piece of the file so you can see the structure of the file and extract out the information on the fields that you will need. This is to mimic a situation where the file is too big for your available disk space.
- On BCE, I can read in 1000 rows from the zipped file in approximately 0.2 seconds and 10000 rows in 1.8 seconds. Your solution should not use an approach that is substantially slower.
- You cannot read more than 100,000 rows into R at any given time and should discard what you have read after obtaining the row(s) you are keeping.
- Please make sure to write your code modularly. E.g., you might have an R function that figures out what rows you want and another function that does the actual reading from the file.
- We haven't covered this yet (it's in Section 7 of Unit 4), but you should create your full dataframe or matrix to store the subsample of 10,000 values in advance and not append rows as you go.
- Make sure to set the random number seed using *set.seed()* before determining your random sample, so your sample is reproducible. E.g., simply put `set.seed(0)` at the top of your R code.