Stat243: Problem Set 6, Due Wed. October 31

October 20, 2018

This covers Units 7 and 8, as well as a part of Unit 10.

It's due **as PDF submitted to bCourses** and submitted via Github at 2 pm on Oct. 31, with the first question due before Section on October 30.

Some general guidelines on how to present your problem set solutions:

1. Please use Rmd/Rtex as in previous problem sets.
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 PDF submission should be the PDF produced from your Rmd/Rtex. Your Github submission should include the Rtex/Rmd file, any R code files containing chunks that you read into your Rtex/Rmd file, and the final PDF, all named according to the guidelines in *howtos/submitting-electronically.txt*.
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. Please note my comments in the syllabus about when to ask for help and about working together. In particular, **please give the names of any other students that you worked with on the problem set and indicate in comments any ideas or code you borrowed from another student.**

Problems

1. The goal of this problem is to think carefully about the design and interpretation of simulation studies, which we'll talk about in Unit 10. In particular, we'll work with Lo et al. (2001), an article in Biometrika, which is a leading statistics journal. The article is available as *lo_etal_2001.pdf* under the *ps* directory on Github. Read the first three pages and Section 3 of the article. You don't need to understand their algorithm for testing the null hypothesis [i.e., you can treat it as some black box algorithm] or the theoretical development, though it may help to skim through some of the material on the algorithm for context.
   Briefly (a few sentences for each of the four questions below) answer the following questions. **Please submit your answers before section on Tuesday, October 30** via this Google form.
   (a) What are the goals of their simulation study and what are the metrics that they consider in assessing their method?

(b) What choices did the authors have to make in designing their simulation study? What are the key aspects of the data generating mechanism that likely affect the statistical power of the test? Are there data-generating scenarios that the authors did not consider that would be useful to consider?

(c) Interpret their tables on power (Tables 2 and 4) - do the results make sense in terms of how the power varies as a function of the data generating mechanism?

(d) Do their tables do a good job of presenting the simulation results and do you have any alternative suggestions for how to do this?

2. Using the Stack Overflow database (http://www.stat.berkeley.edu/share/paciorek/stackoverflow-2016.db), write SQL code that will determine which users have asked Spark-related questions (tags that have "apache-spark" as part of the tag – you'll need to use the SQL wildcard character, %) but not Python-related questions. Those of you with more experience with SQL might do this in a single query, but it's perfectly fine to create one or more views and then use those views to get the result as a subsequent query. Report how many unique such users there. The Stack Overflow SQLite database is ~ 650 MB on disk, which should be manageable on most of your laptops, but if you run into problems, you can use an SCF machine or Savio.

3. With the full Wikipedia traffic data for October-December 2008 (available on Savio in */global/scratch/paciorek/wikistats_full/dated*), figure out a question to investigate with the data and use Spark to process the data to answer the question. You should use Spark to process the dataset, but you can then use R or Python as you wish to do the subsequent analysis (which might be simply a graphical presentation or might involve fitting a statistical model). For example, given the time period available, you could consider a U.S. politics-related question, a question related to holidays since many holidays occur in the fall, a question related to seasonality since the data cover the period where the northern hemisphere enters winter and the southern hemisphere enters summer, or a question related to daily/weekly patterns. Given the information on language, you may be able to get at some sort of pattern related to culture or religion. You can also extend the Obama analysis, but it should be more than a trivial extension.
   Notes:

   (a) We may give extra credit for particularly interesting or in-depth analyses. While the main purpose of the problem is to get you some basic experience in using Spark, in previous years, some students have discussed their work when interviewing for jobs, so doing a more extensive analysis may be beneficial outside the context of class or our grading of your solution.

   (b) Note that I'm not expecting you to already know Python, so feel free to ask for help (and for those of you who know Python to help out) on PySpark coding questions on Piazza or in office hours. Also there are materials available from a Python workshop I gave at https://github.com/berkeley-scf/python-bootcamp-fall-2018; in particular the *python.html* file.

   (c) In order to make sure we don't exceed our total allocation of computational hours on Savio (spread across the class), **please request no more than four nodes and do not set a time limit of greater than three hours on any job**, unless you discuss it with Chris first. Also, make sure to test your code on a portion of the data before doing computation on the full dataset (e.g., you can copy some of the data into your own directory and read into Spark from there).

4. This question asks you to complete the exercise begun in section on October 16. Consider the full Wikipedia traffic data as in problem 3, but use the data in */global/scratch/paciorek/wikistats_full/dated_for_R*.

It's the same data as in problem 3, but the partitions are half as big so that one can fit 24 partitions in memory on a single Savio node.

(a) Using either *foreach* or *parSapply* (or *parLapply*) on a single Savio node in the savio2 partition (i.e., use "-p savio2" when submitting your job), write code that, in parallel, reads in the space-delimited files and filters to only the rows that refer to pages where "Barack_Obama" appears. Collect all the results across the 960 files into a single data frame. (Note that the R packages you'll need should be available in R if you use "module load r r-packages", so you should not have to install them.) You can do this either in an interactive session using *srun* or a batch job using *sbatch*. And if you use *srun*, you can run R itself either interactively or as a background job. If it's taking a while and you want to run the code on, say, a quarter of the files and then assume the time scales accordingly, that's fine.
Hints: (a) *readr::read_delim()* should be quite fast if employed effectively, (b) there are lines with fewer than 6 fields, but *read_delim()* should still work and simply issue a warning, and (c) there are lines that have quotes that should be treated as part of the text of the fields and not as separators. Also, as usual, you should test your code on a small subset interactively to make sure it works before unleashing it on the full dataset.
Alternatively, if you want to explore parallelizing bash shell code, you should be able to do this problem without using R at all.

(b) When I run the Spark code provided with Unit 7, it takes ~15 minutes using 96 cores to create the filtered dataset that just has the Obama-related webpage traffic using Spark. Assuming that you can achieve perfect scalability (i.e., that if you ran your code in part (a) on 4 times as many cores, it would go 4 times as fast), compare the effectiveness of using parallelized R versus using PySpark in terms of how long it takes to do the filtering. (Note that Unit 8 shows how one could parallelize R across multiple nodes, but I won't ask you to actually do that here.)