# Problem Set 5

Due Friday Oct. 28, 10 am

## Comments

- This covers Units 6 and 7.
- It's due at 10 am (Pacific) on October 28, both submitted as a PDF to Gradescope as well as committed to your GitHub repository.
- Please see PS1 and the grading rubric for formatting and attribution requirements.
- Running parallel code, working with large datasets, and using shared computers can result in delays and unexpected problems. Don't wait until the last minute to work on problems 1 and 2. We are happy to help troubleshoot problems you may have with accessing the SCF, running parallel R and Python code, submitting jobs to the SCF cluster, etc. We will not be happy to do so if it is clear that you started at the last minute.
- Given that you'll be running batch jobs and operating on remote servers, in many cases you'll need to provide your code in chunks with `eval=FALSE`. You can paste in any output you need to demonstrate your work. Remember that use can use "`"" to delineate blocks of text you want printed verbatim. You may also want to put your answer to the SQL question in an unevaluated chunk (or possibly use`cache=TRUE‘ as a chunk option), depending on how long the query takes.

## Problems

1. This problem asks you to use the `future` package to process some Wikipedia traffic data and makes use of tools discussed in Section on October 7. The files in `/scratch/users/paciorek/wikistats/dated_201` (on the SCF) contain data on the number of visits to different Wikipedia pages on November 4, 2008 (which was the date of the US election in 2008 in which Barack Obama was elected). The columns are: date, time, language, webpage, number of hits, and page size. (Note that the Unit 7 Dask bag example and Question 2 below use a larger set of the same data.)

a. In an interactive session on the SCF Linux cluster (ideally on the `low` partition, but if necessary on the `high` partition), started using `srun` as discussed in Section:

   i. Copy the *data* files to a subdirectory of the `/tmp` directory of the machine your interactive session is running on. (Keep your *code* files in your home directory.) Putting the files on the local hard drive of the machine you are computing on reduces the amount of copying data across the network (in the situation where you read the data into your program multiple times) and should speed things up in step ii.

ii. Write efficient R code to do the following: Using the `future` package, with either `future_lapply` or `foreach` with the `doFuture` backend, 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 in the page title (column 4). You can use the code from Unit 6 as a template. Collect all the results into a single data frame. In your `srun` invocation and in your code, please use 4 cores in your parallelization so that other cores are saved for use by other users/students. IMPORTANT: before running the code on the full set of data, please test your code on a small subset first (and test your function on a single input file serially).

iii. Tabulate the number of hits for each hour of the day. (I don't care how you do this - you could use dplyr or base R functions or something else.) Make a (time-series) plot showing how the number of visits varied over the day. Note that the time zone is UTC/GMT, so you won't actually see the evening times when Obama's victory was announced - we'll see that in Question 2.

iv. Remove the files from `/tmp`.

Hints: (a) `readr::read_delim()` should be quite fast if you give it information about the structure of the files, (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.

b. Now replicate steps i and ii but using `sbatch` to submit your job as a batch job to the SCF Linux cluster, where step ii involves running R from the command line using `R CMD BATCH`. You don't need to make the plot again. Note that you need to copy the files to `/tmp` in your submission script, so that the files are copied to `/tmp` on whichever node of the SCF cluster your job gets run on. Make sure that as part of your `sbatch` script you remove the files in `/tmp` at the end of the script. (Why? In general `/tmp` is cleaned out when a machine is rebooted, but this might take a while to happen and many of you will be copying files to the same hard drive.)

2. Consider the full Wikipedia traffic data for October-December 2008 (already available in `/var/local/s243/wikistats/dated_2017` on any of the SCF cluster nodes in the low or high partitions).

   a. Explore the variation over time in the number of visits to Barack Obama-related Wikipedia sites, based on searching for "Barack_Obama" on English language Wikipedia pages. You should use Dask to do the reading and filtering. Then group by day-hour (it's fine to do the grouping/counting in Python in a way that doesn't use Dask data structures). 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 Python itself either interactively or as a background job. Time how long it takes to read the data and do the filtering to get a sense for how much time is involved working with this much data. Once you have done the filtering and gotten the counts for each day-hour, you can simply use standard R or Python code on your laptop to do some plotting to show how the traffic varied over the days of the full October-December time period and particularly over the hours of November 3-5, 2008 (election day was November 4 and Obama's victory was declared at 11 pm Eastern time on November 4).

   b. Extra credit: Carry out some other in-depth analysis of the Wikipedia data (it doesn't have to involve Barack Obama), addressing a question of interest to you.

   Notes:

- I'm not expecting you to know any more Python than we covered in the Unit 6/7 material on Dask and in Section, so feel free to ask for help (and for those of you who know Python to help out) on Python syntax on the discussion forum or in office hours.
- There are various ways to do this using Dask bags or Dask data frames, but I think the easiest in terms of using code that you've seen in Unit 7 is to read the data in and do the filtering using a Dask bag and then convert the Dask bag to a Dask dataframe to do the grouping and summarization. Alternatively you should be able to use `foldby()` from `dask.bag`, but figuring out what arguments to pass to `foldby()` is a bit involved.
- Make sure to test your code on a portion of the data before doing computation on the full dataset. Reading and filtering the whole dataset will take something like 60 minutes with 16 cores. You MUST test on a small number of files on your laptop or on one of the stand-alone SCF machines (e.g., radagast, gandalf, arwen) before trying to run the code on the full 120 GB (zipped) of data. For testing, the files are also available in `/scratch/users/paciorek/wikistats/dated_2017`.
- When doing the full computation via your Slurm job submission:
  - Don't copy the data (unlike in Question 1) to avoid overloading our disks with each student having their own copy.
  - Please do not use more than 16 cores in your Slurm job submissions so that cores are available for your classmates. If your job is stuck in the queue you may want to run it with 8 rather than 16 cores.
  - As discussed in Section, when you use sbatch to submit a job to the SCF cluster or srun to run interactively, you should be using the `--cpus-per-task` flag to specify the number of cores that your computation will use. In your Python code, you can then either hard-code that same number of cores as the number of workers or (better) you can use the `SLURM_CPUS_PER_TASK` shell environment variable to tell Dask how many workers to start.

3. Using the Stack Overflow database, write SQL code that will determine which users have asked html-related questions but not css-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 are. There are various ways to do this, of which I only covered some approaches in Unit 7 and in the videos. You can run your query via either R or Python.

4. This question prepares for the discussion of a simulation study in section on Friday October 28. The goal of the problem is to think carefully about the design and interpretation of simulation studies, which we'll talk about in Unit 9, in particular in Section on Friday October 28. In particular, we'll work with Cao et al. (2015), an article in the Journal of the Royal Statistical Society, Series B, which is a leading statistics journal. The article is available as `cao_etal_2015.pdf` under the `ps` directory on GitHub. Read Sections 1, 2.1, and 4 of the article. Also read Sections 2 of Unit 9. Briefly (a few sentences for each of the three questions below) answer the following questions.

   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 might affect their assessment of their method?

c. Consider their Tables reporting the simulation results. For a method to be a good method, what would one want to see numerically in these columns?