Stat243: Problem Set 5, Due Monday Oct. 26, 10 am

October 14, 2020

This covers Units 6-7.
It's due **as PDF submitted to Gradescope** and submitted via GitHub at 10 am on Oct. 26.
Comments:

1. The formatting requirements are the same as previous problem sets.
2. As discussed in the Piazza post, we are happy to help troubleshoot problems you may have with the SCF, parallelizing R, submitting jobs to the SCF cluster, etc. We are not so happy to do it the weekend of Oct. 24 if it is clear that you didn't put in time in section and during the week of Oct. 19 to get up to speed.
3. 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. Consider the following estimates of the variance of a set of numbers. Mathematically the variance of x and variance of z are identical. But the results depend on whether the magnitude of the numbers is large or small.

```
set.seed(2)
z <- rnorm(10, 0, 1)
formatC(z[1:5], 20, format = 'f')

## [1] "-0.89691454662498137917" "0.18484918464674249261"
## [3] "1.58784533120882320745"  "-1.13037567424628537793"
## [5] "-0.08025175655098928940"

x <- z + 1e12
formatC(x[1:5], 20, format = 'f')

## [1] "999999999999.10302734375000000000"
## [2] "1000000000000.18481445312500000000"
## [3] "1000000000001.58789062500000000000"
## [4] "999999999998.86962890625000000000"
## [5] "999999999999.91979980468750000000"
```

```
formatC(var(z), 20, format = 'f')

## [1] "0.97020065227876062242"

formatC(var(x), 20, format = 'f')

## [1] "0.97024419572618270102"
```

How many digits do these two estimates agree on? Explain why that is the case when mathematically the variance of z and the variance of x are exactly the same. Which of the two is the more accurate answer? You can assume that for a vector w, var(w) is calculated as $\sum_{i=1}^{n}(w_i - \bar{w})^2/(n-1)$

2. Numerical problems in logistic regression. Consider that in logistic regression the linear predictor, $X_i\beta$, (where X_i is the ith row of the predictor matrix, X) is transformed into a probability using the inverse logistic (expit) transformation.

$$p_i = \text{expit}(X_i\beta)$$

where

$$\text{expit}(z) = \frac{\exp(z)}{1+\exp(z)}.$$

This mathematical expression for the *expit* function doesn't work numerically on a computer for large values of z. Explain why not and re-express the function such that if implemented in computer code, it is numerically stable. Also explain what happens if you use your re-expression but for very small values of z.

3. This problem asks you to use the *future* package to process some Wikipedia traffic data. The files in ***/scratch/users/paciorek/wikistats/dated_2017_small/dated*** (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 in Unit 8 and in PS6, we'll work with a larger set of the same data using Python's Dask package.)

   (a) In an interactive shell on one of the SCF Linux servers named gandalf, radagast, or arwen:

      i. Copy the files to the '/tmp' directory. (Recall the use of wildcards in file globbing to select a set of files, from our work with bash.) 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 *unit7-parallel.R* as a template, in particular the chunks labeled 'rf-example', 'foreach' and 'future_lapply'. Collect all the results into a single data frame. Please use 4 cores in your parallelization (the machines have more cores, but other students may be using them at the same time).
      IMPORTANT: before running the code on the full set of data, please test your code on a small subset first.

iii. Tabulate the number of hits for each hour of the data. (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.

(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.

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.

4. Extra credit: This problem explores the smallest positive number that R can represent and how R represents numbers just larger than the smallest positive number that can be represented. (Note: if you did this in Python you'd get the same results.)

   (a) By experimentation in R, find the base 10 representation of the smallest positive number that can be represented in R. Hint: it's rather smaller than 1×10^{-308}.

   (b) Explain how it can be that we can store a number smaller than 1×2^{-1022}, which is the value of the smallest positive number that we discussed in class. Start by looking at the bit-wise representation of 1×2^{-1022}. What happens if you then figure out the natural representation of 1×2^{-1023}? You should see that what you get is actually a well-known number that is not equal to 1×2^{-1023}. Given the actual bit-wise representation of 1×2^{-1023}, show the progression of numbers smaller than that that can be represented exactly and show the smallest number that can be represented in R written in both base 2 and base 10.

   Hint: you'll be working with numbers that are not normalized (i.e., denormalized; numbers that do not have 1 as the fixed number before the decimal point in the representation at the bottom of page 7 of Unit 6).