Stat243: Problem Set 5, Due Friday October 29

October 14, 2021

This covers Units 6-7 and part of Unit 9.
It's due **as a PDF submitted to Gradescope** and submitted via GitHub at 10 am on Oct. 29.
Comments:

1. The formatting requirements are the same as previous problem sets.
2. We are happy to help troubleshoot problems you may have with the SCF, parallelizing R, submitting jobs to the SCF cluster, etc. We will not be happy to do so if it is clear you didn't put in the time in section on Oct. 22 to get some initial practice. Also, because you'll be working on shared computers, waiting until the evening of Oct. 28 to do problem 3 is a bad idea.
3. You will probably want to wait until after section on Oct. 22 to work on problem 3 and until the week of October 25 to do the reading in problem 4.
4. 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. In class and in the Unit 6 notes, I mentioned that integers as large as 2^{53} can be stored exactly in the double precision floating point representation. Demonstrate how the integers 1, 2, 3, ...,$2^{53}-2$, $2^{53}-1$ can be stored exactly in the $(-1)^S \times 1.d \times 2^{e-1023}$ format where d is represented as 52 bits. I'm not expecting anything particularly formal - just write out for a few numbers and show the pattern. Then show that 2^{53} and $2^{53}+2$ can be represented exactly but $2^{53}+1$ cannot, so the spacing of numbers of this magnitude is 2. Finally show that for numbers starting with 2^{54} that the spacing between integers that can be represented exactly is 4. (Note you don't need to write out e in base 2; you can use base 10). Then confirm that what you've shown is consistent with the result of executing $2^{53}-1$, 2^{53}, and $2^{53}+1$ in R.
2. Here we'll consider the effects of adding together numbers of very different sizes. Let's consider adding the number 1 to 10000 copies of the number 1×10^{-16}. Mathematically the answer is obviously $1 + 1 \times 10^{-12} = 1.000000000001$ by multiplication, but we want to use this as an example of the accuracy of summation with numbers of very different magnitudes, so consider the sum $1 + 1 \times 10^{-16} + \cdots + 1 \times 10^{-16}$.

   (a) How many digits of accuracy are the most we can expect of our result (i.e., assuming we don't carry out the calculations in a dumb way). In other words, if we store 1.000000000001 on a computer, how many digits of accuracy do we have? This is not a trick question.

(b) In R, create the vector $x = c(1, 1 \times 10^{-16}, \ldots, 1 \times 10^{-16})$. Does the use of *sum()* give the right answer up to the accuracy expected from part (a)?

(c) Do the same as in (b) in Python. For Python the *Decimal()* function from the decimal package is useful for printing additional digits. Also, this code will help you get started in creating the needed vector:

```
import numpy as np
vec = np.array([1e-16]*(10001))
```

(d) Use a *for* loop to do the summation, $((x_1 + x_2) + x_3) + \ldots$. Does this give the right answer? Now use a *for* loop to do the summation with the 1 as the last value in the vector instead of the first value. Right answer? If either of these don't give the right answer, how many decimal places of accuracy does the answer have? Do the same in Python.

Your results from (b) and (d) should suggest that R's *sum()* function is not simply summing the numbers from left to right.

3. 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 22. 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 a subdirectory of the /tmp 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 *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 (and test your function on a single input file serially).

      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. 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 on PS6.

   (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.

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. The goal of this 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 November 6. 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 Section 1, Section 2.1, and Section 4 of the article. Also read Sections 2.1-2.2 of Unit 9

   You don't need to understand their method for fitting the regression [i.e., you can treat it as some black box algorithm] or the theoretical development. In particular, you don't need to know what an estimating equation is - you can think of it as an alternative to maximum likelihood or to least squares for estimating the parameters of the statistical model. Equation 3 on page 759 is analogous to taking the sum of squares for a regression model, differentiating with respect to β, and setting equal to zero to solve to get $\hat{\beta}$. Similarly in Equation 3, to find $\hat{\beta}$ one sets the equation equal to zero and solves for β. As far as the kernel, its role is to weight each pair of observation and covariate value. This downweights pairs where the covariate is measured at a very different time than the observation.

   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 1 and 3 reporting the simulation results. For a method to be a good method, what would one want to see numerically in these columns?

   In Section on October 22, we'll talk in more detail about this simulation study.

5. 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).