Stat243: Problem Set 7, Due Fri. November 16

November 2, 2018

This covers Unit 9 and a bit of Unit 10.
It's due **as PDF submitted to bCourses** and submitted via Github at 2 pm on Nov. 16.
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. Suppose I have a statistical method that estimates a regression coefficient and its standard error. I develop a simulation study and have m = 1000 simulated datasets that each give me an estimate of the coefficent and its standard error. How would I determine if the statistical method properly characterizes the uncertainty of the estimated regression coefficient? Note your answer could be as simple as a sentence or two describing what quantities to consider.
2. Suppose I have a very large dataset, with $n = 1 \times 10^9$, so I have n observations and an $n \times p$ matrix X of predictors, where $p = 8$.
   (a) Ordinarily, how much memory would the dataset take up?
   (b) Now suppose that there are only 10000 unique combinations of the p covariates. Given what you know about data structures in R, how could you store the data to use up much less memory? How much memory would be used by your solution?
   (c) Now suppose you need to run *lm()*, *glm()*, etc. on this data. Why would all of your work in part (b) go to waste?

(d) If you need to find the OLS, $(X^\top X)^{-1}X^\top Y$ estimator here, how could you code this up (please provide pseudo-code; you don't need to write any R code) so that you do not need to use up the full memory and can take advantage of your data structure(s).

3. Suppose I need to compute the generalized least squares estimator, $\hat{\beta} = (X^\top \Sigma^{-1} X)^{-1} X^\top \Sigma^{-1} Y$, for X $n \times p$, Σ $n \times n$ and assume that $n > p$. Assume n could be of order several thousand and p of order in the hundreds. First write out in pseudo-code how you would do this in an efficient way - i.e., the particular linear algebra steps and the order of operations. Then write efficient R code in the form of a function, *gls()*, to do this - you can rely on the various high-level functions for matrix decompositions and solving systems of equations, but you should not use any code that already exists for doing generalized least squares.

4. We've seen how to use Gaussian elimination (i.e., the LU decomposition) to solve $Ax = b$ and that we can do the solution in $n^3/3$ operations (plus lower-order terms). Now let's consider explicitly finding A^{-1} and then calculating $x = A^{-1}b$ via matrix-vector multiplication. If we look at R's *solve.default()*, we see it solves the system $AZ = I$ to find $Z = A^{-1}$ and *help(solve)* indicates it calls a Lapack routine DGESV (http://www.netlib.org/lapack/explore-html/d7/d3b/group__double_g_esolve_ga5ee879032a8365897c3ba91e that uses the LU decomposition.
Count the number of computations for

   (a) transforming $AZ = I$ to $UZ = I^*$ (where I^* is no longer a diagonal matrix),

   (b) for solving for Z given $UZ = I^*$, and

   (c) for calculating $x = Zb$.

   Then compare the total cost to the $n^3/3$ cost of what we saw in class.

   Notes: In counting the computations you should be able to make use of various results we derived in class concerning the Gaussian elimination computations and computations involved in a backsolve, so your answer should be able to simply combine together results we've already discussed without any detailed new derivation.
   Second note: Given that R's call to *dgesv* doesn't take account of the special structure of I on the right-hand side, you do not need to take account of the fact that because I has zeroes and ones, one can actually save some computation. You can simply count the calculations as if I were filled with arbitrary values. If we did actually try to be careful about making use of the structure of I, it turns out we could save $n^3/3$ calculations.

5. Compare the speed of $b = X^{-1}y$ using: (a) `solve(X)%*%y`, (b) `solve(X,y)`, and (c) Cholesky decomposition followed by solving triangular systems. Do this for a matrix of size 5000×5000 using a single thread, using a matrix X constructed from $W^\top W$ where the elements of the $n \times n$ matrix W are generated independently using *rnorm()*.

   (a) How do the timing and relative ordering amongst methods compare to the order of computations we discussed in class, the notes, and above in problem 4.

   (b) Are the results for b the same numerically for methods (b) and (c) (up to machine precision)? Comment on how many digits in the elements of b agree, and relate this to the condition number of the calculation.