Homework 7

Due: Tue, Nov 25 at Lecture

1. **Fox Problems:**
   - 11.1: This will have you thinking between matrix and scalar forms.
   - 11.2: This has been a proof on methods comps exams in the past. A matrix is *symmetric* if/f for the matrix $\mathbf{M}$, $\mathbf{M}' = \mathbf{M}$. A matrix is idempotent if multiplying the matrix time it self results in the same matrix – indeed, the concept of idempotents is a portmantu of idem (same) and potent (power). So $\mathbf{M^2} \equiv \mathbf{MM} = \mathbf{M}$.
   - 11.4: Don't worry too much, we will not to the "much more difficult problem" that Fox suggests could be tackled.
2. **Demonstration Problems**
   - Using Duncans' prestige data (either in the `car` package or on Fox's website, demonstrate that built in `influence.measures(·)` and other calls produce the same results as if you program by hand (aka, using the formulas from lecture and Fox). Demonstrate this for: (a) hat-values; (b) studentized residuals; (c) DFBETAS; and, (d) Cooks distance. That is, run a small regression (any of the canonical ones is fine), and then write your own function that you can apply to the fit model object to calculate the quantities of interest.
   - Create some fake data that, when run through a q-q plot, will produce each of the problems that we identified. Of course, your data can't both have thick tails and thin tails. So, you *can* make separate datasets for each type of problem, or you can show thick tails on the left- and right-sides in the same plot.
3. **Simulation Problem** Write a function that will let you assess the performance of an estimator that assumes normality when the data in fact come from a non-normal distribution. Compare p-values of test statistics when you (inappropriately) assume normalcy and when you (appropriately) test with the corresponding distribution. Make your function sufficiently general that you can pass arguments that: (a) change the strength of relationship between Xs and Y; (b) change the size of the sample; (c) change the number of simulations you run; (d) can toggle on and off diagnostics plots. So, your function should have some options in it like:
   `foo <- function(b0 = 1, b1 = 1, b2 = 2, n = 50, df = 50, nSims = 100, plots = FALSE)`
   Make sure that your function returns the p-values for tests. Then, use the function you've written to conclude when things are *bad* and how bad they are; and when we might have enough data for things to be *not so bad*.
4. **Data-based Problem** Go back to the data problem that you started in HW 5.[1] Use the diagnostics tools that we developed this week to assess whether the data you were using for that problem had any outliers or issues in the distribution of the errors.
5. I *might* include one more problem depending on where you get in lab today. I'll email the class if I do such a dastardly thing.

[1] How does your code look?