Stat 230, Spring 2016
Homework 11: Model Selection, Principal Components

Due Thursday 4/21/16 at 11:59pm on bspace.
Note: depending on how efficient your code is the simulations may take a little while. You can do three things (that I can think of) to deal with this problem:

1. Start early so you have time if it takes you a long time.

2. Rewrite your code so it works more efficiently.

3. Do 100 replications instead of 1000. I'd much rather have you get some results for each part rather than precise results for only part of the assignment.

PART 1. Model Selection
The file HW11.R has the code I used to generate the data for HW 10. Through simulation we will get an idea of how well each of the penalty methods discussed in lecture (Mallow's Cp, AIC, and BIC) performs relative to cross validation. For HW10, I used this function to generate the data with the argument `wh=20` so that the full model was actually all 20 variables, but some were very weakly correlated with the response.

You will use the arguments as given, so you have some chance of fitting noise and you may or may not end up including some of the variables with very weak relationships with the response.

This lab is written using Freedman's notation; HTF would use d instead of p and N instead of n. Perform 1000 replications using each of the following values of n: 100 and 1000. Use backwards selection (it's easiest) to find a sequence of models as you did in HW 10. Then using each of the four criteria for model selection (10-fold cross validation using RSS, Mallow's Cp, AIC, and BIC), pick the best model in the sequence. You probably want to reuse your code from HW 10 for CV/RSS and `stepAIC()` in the `MASS` package for AIC and BIC. Mallow's Cp is pretty straightforward just using residuals from `lm()`. The `stepAIC()` function will do the backwards selection for you (look at the arguments) and it's fine to use that even if it produces a slightly different sequence of models.

For each replication you'll want to store the value of the criterion used for each of the 21 different model sizes (intercept + 0-20 of the variables are used), so your end results should be saved in, for example, a 1000 by 21 matrix or data frame (for each value of n and for each selection rule, so 8 such objects).

For each value of n, you will make 2 plots.

1. Make a plot that has four lines with points (use `type="b"`), one for each method. The horizontal axis will be for p. The vertical axis will be for the criteria used for each method. Take the mean of the value across all replications, take the log for the CV RSS and Mallow's Cp, then divide all 4 by the value at $p = 16$ (intercept plus 15

of the 20 variables). At least they can be reasonably shown on the same plot then, and you can see how shallow/sharp the different lines are.

We are basically reproducing the picture in the upper left panel on page 62 of HTF with the following differences:

- They search through all subsets, which is much more computationally intensive (especially when the number of variables is large).
- They are choosing the simplest model within a SE of the smallest CV Error, we are choosing based on the 4 criteria (including smallest CV Error).
- Since we are comparing the 4 methods, we want to include them on the same scale. That's why we're dividing by the $p = 16$ value (that corresponds to the true model including intercept).

2. Make a plot that shows the empirical distribution of best model sizes for the 1000 replications, for each method. You can do this in a few different ways: a mosaic plot, side-by-side box plots, histograms stacked vertically, or any other visualization that you think shows any differences in the distributions as well as possble.

Each replication results in a "best" model (for each method). Calculate the proportion of times each coefficient was left out (for the first 15) and put in (for the last 5). Comment briefly on differences between the four methods. Do any methods tend to overfit/underfit consistently?

PART 2. Principal Components

Repeat PART 1 using principal components, but choose model complexity based on the rule in Hastie, Tibshirani, and Friedman on page 62. Duplicate the plot for PCR in figure 3.7 on that page. When you do part 1 over again, you only need to do the cross validation to select model size, don't bother with the different criteria.

In order to get the vertical SE to get the "best RSS + 1 SE", get the RSS for each of the 10 folds separately and take the SD of those RSS's (see page 244 of HTF).

For the picture, don't do quite what they did for one dataset, but do a similar picture for all 1000 replications. The point for each model size will be the average RSS over the replications, and the interval will be the SD over the 1000 replications.

Since each principal component is a linear combination of the original vectors of X, you can solve for coefficients of the original variables. You should also reverse the standardization you did before making the principal components, so that the coefficients correspond to the actual data used to generate Y (this should only make a slight difference). Now compare with the true coefficients. Based on the replications, does PCR seem biased? You can answer this based on either:

1. A rough hypothesis test for each coefficient based on the empirical distribution of the 1000 $\hat{\beta}_j$ for each coefficient. OR

2. A confidence interval again based on the empirical distribution of the $\hat{\beta}_j$ for each coefficient. See whether the true β_j is in the CI for each coefficient.

Note that since the original vectors in X are independent (not orthogonal) the cloud of points is basically a hypersphere and the direction of the first principal component is whatever direction that hypersphere is randomly stretched into being slightly elliptical. If you want to think about this in two dimensions, imagine two independent standard normals. Because of the rotational symmetry of this distribution, the angle of the first principal component is equally likely to be anything. For this reason this is not a very natural dataset to use principal components, but this HW is already long enough. Next HW we'll do it for correlated variables...