Project: "Bias, Variance and Parsimony in Regression Analysis"

Important Note:

I plan to hold most of the interactive grading during March 11-14, including during class time. I can schedule some on March 17 if need be, but do not want to extend into the finals period.

Grading sessions will last 40 minutes per group. This includes a 5-minute presentation for each group member on some aspect of your report. (The presentations collectively need not cover your entire report.) The remainder of the time will be used in the manner of an ordinary interactive grading session, with my asking questions on both the report and the general course material in the last 3-4 weeks of the course.

Due date is March 10, with the same packaging rules etc. as for regular homework assignments.

Problem 1:

Problem 1 of the Project is now online. You will need Eqn. (23.32) and the comment at the end of (23.38). We'll probably reach that material in Thursday's lecture, but you should get started now, so here is a summary of what you need to know for this problem:

Say we want to model the mean Y given predictors $X^{(1)}, X^{(2)},...,X^{(p)}$, from sample data. We put our sample Y values in a vector V, and put our sample data for $X^{(i)}$ in column i+1 of a matrix Q, i = 1,...,p. We put 1s in the first column of the matrix.

For the baseball player data with n = 1033 and p = 2 (we are predicting weight from height and age), V and Q look like (23.30) and (23.32). The estimated coefficient vector (3 components) is then given by (23.33).

In our case here, p = 1, and as explained following (23.38), we do not have the column of 1s. Everything is much simpler.

Here you will explore the bias of an approximate regression model.

Suppose that, unknown to the modeler, X has a U(0,1) distribution and $m_X(t) = t^{0.75}$, t in (0,1). The modeler, eyeballing a scatter plot of the sample data, decides to model m_X as a straight line through the origin, i.e. $m_X(t) = \beta$ t for some scalar β (to be estimated from the data).

Denote the estimated regression function by mh_X ("h" for "hat"). Find the asymptotic bias at t = 0.5. An analytical answer is required, not simulation (though you may wish to simulate as a check on your answer).

Hint: Think about what the estimated β converges to as the sample size n goes to infinity. (Mathematically, it does not follow from $U_n \rightarrow c$ that $EU_n \rightarrow c$, but don't worry about this.)

If you don't use the hint (there are other ways to do the problem), you must justify your solution.

Problem 2:

Prologue:

As stated in class and in our book, one of the most important goals in regression analysis is *parsimony*, meaning the development of a model with a relatively small number of predictors. The term*dimension reduction* is often used more or less synonymously. A related term is *model selection*.

Note carefully that there is no real "solution" available to this problem, though there are many different approaches in common use today. Here you will explore a "new" approach, basically a mini-research project. (I say "new" in the sense that I haven't seen it described in books, research journals and the like, but I would guess that some people use it informally.)

By far the most common approach to model selection is to use significance testing, as discussed in Section 23.15.3.1 of our book. As is usually the case with testing, we especially run into problems in large data sets, where "everything is significant." With a large enough data set, significance testing doesn't reduce the number of predictors in our model at all.

It is thus desirable to have a model selection method that yields parsimony no matter how large the data is. It would choose a model that captures *almost* all the predictive ability of the full model, even if n is really large (in contrast to significance testing). That is what we will do here. Specifically, our approach will be this:

Choose a prediction accuracy criterion (PAC) and a proportion k, small enough so that 1-k fits your definition of "almost."

Then, starting with the full model (i.e. the one that uses all available predictor variables), delete predictors one at a time as long as the PAC doesn't deteriorate much.

Some PACs increase with accuracy (e.g. adjusted R^2, while others decrease (e.g. AIC). In the former case, delete the variable if the PAC after deletion is at least 1-k of the PAC before deletion; in the latter case, delete if the new PAC is less than 1+k of the old one.

So that the procedure can also work reasonably well in smaller samples (or large samples with many predictors), the PAC should be chosen so that it incorporates some kind of protection against overfitting.

Note that this is also a bias-vs.-variance issue. Generally, the beta-hat values in the submodel will have lower standard errors than in the full model, and thus so will our predicted values in new cases. However, omission of predictors results in some bias in those values. You won't be asked to address this in your report, but you should think about it.

Problem specs:

a. Write a function **prsm()**, stored in a file **Parsimony.R**, with the following call form:

   ```
   prsm(y,x,k=0.01,predacc=ar2,crit=NULL,printdel=F)
   ```

   with arguments as follows:

   - **y:** The vector of response values in the data.
   - **x:** The matrix of predictor values.
   - **k:** The "almost" measure, discussed above.
   - **predacc:** The PAC function, with arguments **y** and **x**.

- **crit:** Either "max" or "min", depending on whether good values of the PAC are large or small.
- **printdel:** If TRUE, gives a "progress report" as the computation proceeds. Each time a predictor is deleted, the new value of the PAC is printed out, along with the name of the variable.

Your file **Parsimony.R** will also contain functions **ar2()** and **aiclogit()**, usable as the actual parameter for the formal parameter **predacc** in **prsm()**. The **ar2()** function calls **lm()** and then calls**summary()**, and returns the adjusted R^2 value. In the case of **aiclogit**, **glm()** and **summary()** are called for the logistic model, and the AIC value is returned.

Note: The computations can be quite lengthy. For Extra Credit, give the user the option of doing them in parallel, using the "Snow" portion of R's **parallel** package. Here you would add an argument **cls=NULL** for the cluster (so parameter computation is optional, and your code can still be run in Yingkang's tests). It's not hard, even if you've never done any parameter programming; see Chapter 1 in the draft first half of my forthcoming book. Make sure your parallel operation gives the same answers as the serial one!

Example:

Pima dataset from the UCI Machine Learning Repository.

```
> prsm(pm[,9],pm[,1:8],predacc=aiclogit,printdel=T) full outcome =
741.4454  deleted  Thick  new outcome =  739.4534  deleted  Insul
new outcome =  739.4617  deleted  Age  new outcome =  740.5596
deleted  BP  new outcome =  744.3059  [1] 1 2 6 7
```

In the end, the "recommended" predictor set consists of NPreg, Gluc, BMI and Genet. Note that it actually has a higher adjusted R^2 value than the full predictor set does.

b. One should always first try some new method on simulated data, to see how well the method does in known settings. You'll do that in this part.

Let $X_1,...,X_{10}$ be i.i.d. U(0,1), with

$m_X(t) = t_1 + t_2 + t_3 + 0.1\ t_4 + 0.01\ t_5$

and with the distribution of Y given X being U(0,2).

The user has a sample of size n from this population, so for example the matrix **x** has n rows and 10 (not 5) columns. Remember, the user doesn't know the true distributions; he/she just applies an ordinary linear regression model.

Run simulations for n = 100, 1000, 10000, 100000, with 3 runs for each n (i.e. a total of 12 runs). In each case, determine what predictor set **prsm()** chooses for k = 0.01 and 0.05. Determine what predictor set would be chosen if the analyst were to run the full model and select any predictor that is "significant" at the 5% level of less.

IN YOUR REPORT, PRESENT YOUR RESULTS IN TABULAR FORM.

c. Try your function on real data sets, of your choice, in all combinations of the following criteria:
    - small n (< 1000) vs. large n (> 10000)
    - small p (< 10) vs. large p (> 15)
    - continuous (or at least ordinary) Y vs. 0-1 Y (linear for the first, logistic for the second)

   That's a total of 8 distinct data sets. As in part (b) above, compare k = 0.01, k = 0.05 and the significance testing approach, **PRESENTING YOUR RESULTS IN TABULAR FORM.**

   You must either state where the data sets are on the Web, or provide them to me.

d. In the case of 0-1 Y, another possible PAC would be the "leaving one out" method, a form of cross-validation. For i = 1,...,n, we temporarily delete observation i from our sample, fit our model to the remaining n-1 cases, and then predict the Y_i. (We predict 1 if the estimated regression function is >0.5, and predict 0 otherwise.) Our PAC is the percentage of correct predictions.
e. Write the PAC for this, **leave1out()**, stored in **Leave.R**. (You're pretending to develop a user-written function not included with the **Parsimony.R** package.);a Test it on the same Pima example as above.