March 25, 2015

Tools for Multiple Regression:

1. R^2
2. ANOVA
3. Permutation test
   - F-test and t-test, where $\varepsilon_i \sim N(0, \sigma^2)$

*In our last class, we talked about how the above three methods are useful but rely only on informal reasoning and observation about the magnitude of a predictor variable's effect on the model.
* There are 2^x possible models when *x* variables are present.
* Considering all the possible pairwise comparisons of predictors is a long and tedious process.

4. **Variable Selection**
   - Choosing which variables to include in a model in a way that optimally balances fit and simplicity is difficult
   - With so many variables to consider, how do we choose which to include in our model?
   - We need an automatic procedure to navigate these possibilities

Walkthrough: hourly wages in the Current Population Survey

There is no single, objectively correct approach to variable selection – we must answer two questions:

1. *What is good?*
   - How do we determine when one model is better than another? Many possible answers
   - There is a fit-simplicity tradeoff -> both good things, but in conflict with one another

   Occam's Razor
   Explanations should be as complex as they need to be and as simple as possible
   Aka find the model that fits well enough but is as simple as we can make it

- Occam's Razor Tool: **AIC** (akaike information criteria)
  - Something we can compute about any model that negotiates the fit-simplicity tradeoff
  - A lower AIC score is better

$$\text{AIC} = n \log\left\{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2\right\} + 2p\,.$$

  * n = sample size; p = # of variables in model
  * n = sample size; p = # of variables in model
  * recall that $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ is the sum of squared residuals
  * dividing by *n* makes the first term equal to $\hat{\sigma}^2$
  * the first term measures fit by reducing the errors
  * the second term (2p) penalizes complexity

- Another Occam's Razor Tool: Adjusted R^2
  - this value can actually get smaller when you add more predictors; adjusted R^2 penalizes overfit

2. *How do I find the good?*
   - if AIC is a formula that can be applied to any model, we now have to find the algorithm that has the optimal tradeoff off of fit and simplicity

Exhaustive enumeration aka "check them all" (like looking for a chocolate bar under many platters)

- Take all possible models, calculate AIC for all, and choose the lowest value
- your computer can do this for many variables; however, exhaustive enumeration only works well for smallish problems (thousands to a few million possible models)

Greedy search is used for a situation with lots and lots of variables
- Greedy search is a general idea and **stepwise selection** is a specific way of carrying it out

Back to the hourly wages walkthrough:
We try to fit a model that includes all variables:

```
lm2 = lm(wage ~ educ + sector + age + sex + south + married + race + union, data=cps)
anova(lm2)
```

- consider all possible variables drops one by one to see its effect on residual sum of squares and AIC:

```
drop1(lm2)
```

- the model in which the "married" variable is dropped leads to the smallest AIC
- this is what it means to be greedy – greedily pick the one model that looks best and minimizes AIC
- the best thing to do is drop the married variable; then we are left with a model of wage based upon all variables except 'married'
- we then did the same thing again to consider deleting more variables

```
lm3 = lm(wage ~ educ + sector + age + sex + south + race + union, data=cps)
drop1(lm3)
```

- there is a slight improvement if we drop race as well
- This procedure—sequentially pruning variables in a way that makes AIC as small as possible—is often called "greedy backward selection."
- It's "backward" because we start from a large model and prune downwards, and it's "greedy" because at each stage we delete the single worst variable, as measured by AIC.

*we can reach a point where no matter what variable we drop, AIC will increase
-> step command

```
lm4 = lm(wage ~ educ + sector + age + sex + south + married + race + union + sex:age, data=cps)
lmstep = step(lm4, direction='backward') <- starting with the full model
```

-returns a ranked list of AIC
- after dropping so many variables, staying where we are is the best AIC (the point on the output where AIC begins to increase)
- caveat: in the presence of interaction variables, stepwise selection will always keep the main effects as long as there's an interaction term in the model; aka step(lm4) considers dropping age:sex but neither age nor sex, both corresponding effects must still be included as long as interaction terms are present

Now to the google flu trends walkthrough:
- our goal is to forecast flu activity based on what people are googling about the flu
Let's do three things:
1) Split the data into a training and testing set, for the purpose of building and testing a predictive model
2) Fit a model to the training set
3) Make predictions on the testing set and check our performance

```
flu_week = flu[,1]
flu = flu[,-1]
#eliminates the week variable from data set

flu_train = head(flu, 182)
flu_test = tail(flu, 26)
#the training set is to fit the model; testing set is to test the model

lm1 = lm(cdcflu ~ flu.and.fever + over.the.counter.flu.medicine + treat.a.fever,
data=flu_train)
yhat_test = predict(lm1, newdata=flu_test)
plot(yhat_test, flu_test$cdcflu)
# Calculate correlation and root mean-squared error
cor(yhat_test, flu_test$cdcflu)
#cor measures how well we do when we predict the set
RMSE = sqrt(mean( (yhat_test-flu_test$cdcflu)^2))
```

Can we do better than this simple three-variable model? We can use every search term in the data set
- helpful hint: "." means R will include all variables not otherwise named

```
lm_all = lm(cdcflu ~ ., data=flu_train)
yhat_all = predict(lm_all, newdata=flu_test)
cor(yhat_all, flu_test$cdcflu)
coef(lm_all)
```

But, can we do at least as well (or even better!) with a simpler model? Let's use stepwise selection selection to check

```
lm_step = step(lm_all, direction='both')
yhat_step = predict(lm_step, newdata=flu_test)
cor(yhat_step, flu_test$cdcflu)

length(coef(lm_all))
length(coef(lm_step))
```

*Correlation actually increases from 0.84 to 0.85 with less predictors; a good example of the goal of Occam's Razor: to balance simplicity and fit

Lastly, let's look at the carauction_training data set.
- All teams are in a contest to find the lowest test set error in building a predictive model that forecasts price, 'MMRAcquisitionAuctionCleanPrice.'
- Think about interactions, nonlinear effects, etc.
- Be original; don't just run stepwise selection because then we would all win Tiff's Treats, how terrible
- we'll continue on this next Monday