4 · Quantifying uncertainty

Due Monday, February 16, 2015

(1) Bootstrapping practice

Refer to the "Creatinine, revisited" walkthrough on the class website.[1] This will show you how to implement bootstrapping as a way to approximate a sampling distribution when you cannot directly simulate samples from the population (i.e. pretty much all the time).

[1] http://jgscott.github.io/teaching/r/creatinine/creatinine_bootstrap.html

Once you're comfortable with this walkthrough, use what you've learned to address the following questions. None should require more than a single paragraph.

(A) In an earlier set of exercises, an interesting issue came up on the problem about the CAPM (capital-asset pricing model): was your estimated "beta" for Wal-Mart the same as that reported by Yahoo Finance? You may have seen a slight discrepency here. Can this discrepancy be explained by sampling variability? After all, you only looked at a sample of data, rather than the entire history of Wal-Mart's returns. Could it be that you got a different beta simply because you used a different sample than Yahoo did? Use what you know about sampling distributions and bootstrapping to address whether this seems like a plausible explanation in light of the data. Show your evidence in a picture.

(B) Return to the data set "ut2000.csv" on SAT scores from UT students across all 10 undergraduate colleges. Calculate an approximate 95% confidence interval for the difference in mean SAT math (SAT.Q) scores between students in the colleges of architecture and liberal arts. (This should be easy if you remember about dummy variables.) Concisely describe what you did, and report the interval.

(C) In your own words, briefly describe the idea of bootstrapping (both what we do and why we do it).

(2) Splines and cross-validation

Return to the data on ethanol emissions (`ethanol.csv`) from a previous exercise. If you recall, I asked you to fit a polynomial regression model to NOx emissions versus the equivalence ratio, choosing the degree of the polynomial "by eye." As you discovered, this can be tricky: it's hard to balance fit (which is good) with simplicity (which is also good) without some firm guidance as to what principle should guide the trade-off. Moreover, your favorite polynomial may have still had some undesirable features, like going crazy near the edge of the plot. (This shortcoming is, alas, pretty typical of polynomial regression models.)

In this question, you'll learn two new ideas to address these issues: spline models and cross-validation.

(A) *Splines* are yet another way to estimate a nonlinear relationship of the form $y_i = f(x_i) + e_i$, where $f(x)$ is called the regression function. Obviously linear and polynomial functions are special cases of regression functions, but spline models are more general.

The simplest way to define a spline is that it is a piecewise polynomial function.[2] Splines classified by their *degree*—for example, piecewise linear, quadratic, or cubic—and by their *knots*, which are where the polynomials connect.[3]

Splines are best understood by example. Load up the ethanol data set, install and then load the `splines` library, and try the following commands in R:

```
plot(NOx ~ E, data=ethanol)
model1 = lm(NOx ~ bs(E, df=2, degree=1), data=ethanol)
points(fitted(model1) ~ E, data=ethanol, col='red', pch=19)
```

The `bs` command inside the model statement tells R to fit a piecewise linear function (degree=1) with a single knot and therefore two separate pieces (df=2). (Here `bs` stands for "b-spline.")

To fit a piece-wise linear with two knots (and therefore three pieces) try this:

```
plot(NOx ~ E, data=ethanol)
model2 = lm(NOx ~ bs(E, df=3, degree=1), data=ethanol)
points(fitted(model2) ~ E, data=ethanol, col='red', pch=19)
```

In general, the fitting routine chooses (`df` − `degree`) knots using the quantiles of the predictor variable, and then fits the functions using a variation on least squares. For example, to fit cubic spline (i.e. a piece-wise cubic function) with 2 knots, use

[2] This is not an entirely complete definition, but it will do for now. You can find out lots more about splines by Googling.

[3] Therefore a spline with no knots is just an ordinary polynomial.

```
plot(NOx ~ E, data=ethanol)
model3 = lm(NOx ~ bs(E, df=5, degree=3), data=ethanol)
points(fitted(model3) ~ E, data=ethanol, col='red', pch=19)
```

The degrees of freedom (df) is 5, and the degree is 3, therefore there are two knots.

There is nothing to turn in for Part A.

(B) To fit a spline model, we obviously need to choose the number of knots. The general principle we hope to follow is that our model should predict well. But "predicting well" is not the same as "fitting the data well"—if it were, we'd just draw a very squiggly line through all the points and call that our model. But this model would be horribly over-fit, and would generalize poorly to future data. That makes it a bad model.

If someone gave you a second, independent data set, you could actually check how well each model predicts the y's, given the x's. This is called *out-of-sample predictive validation*. With only one data set, of course you can't do this. (Plus, if you actually possessed that future data set, you'd want to use it for fitting the model!) But you can get some idea of how a model would generalize to future data by the process of *cross-validation*, which mimics out-of-sample predictive validation.

The idea is this. First, randomly split the rows (cases) of your data set into two chunks: a "training" set, and a "test" set. Use the training set to fit the model, and then make predictions for the likely y values corresponding to x values in the test set. You can then check the actual y values in the test set, to see how well you did at predicting them.[4]

[4] Remember not to use the test set points in fitting the model—that would be cheating.

For example, try the following code.

```
# Split the data into training and testing set
n_total = nrow(ethanol)
n_train = floor(0.8*n_total)
training_points = sample(1:n_total, size=n_train, replace=FALSE)
training_data = ethanol[training_points,]
testing_data = ethanol[-training_points,]
# Fit the model to the training set
model5 = lm(NOx ~ bs(E, df=5, degree=3), data=training_data)
# Predict on the testing set and calculate mean squared error
yhat = predict(model5, newdata=testing_data)
MSE = mean( (yhat - testing_data$NOx)^2 )
MSE
```

Using this code as a starting point, fit a spline to the ethanol data, where the combination of degree and knots is chosen by cross-validation. To keep things simple, you may limit your search for the "optimal" spline to three candidate models of your choosing. Your candidate models might be chosen Goldilocks style: one that underfits, one that fits, and one that overfits, as judged by eye.[5] Turn in your code (printed in a fixed-width font like Courier), an estimate of the mean-squared error for each of the three candidate models, and a plot of your final model with the fitted values superimposed on the data.

[5] Feel free to be more systematic if the fancy strikes.

Some notes:

1) There is a lot of randomness in terms of which points get selected to be "training" versus "testing" points. Thus you'll want to use a Monte Carlo simulation to average the results of many different train/test splits.

2) There is also some arbitrariness in the ratio of training set size to the test set size. We've used a rough 80/20 split of the points in the data set. You may play around with this if you wish, but don't have to.

3) In the code block above, we're only fitting the cubic spline with 2 knots (df=5). In reality, you should actually fit *all three* of your candidate models to the training set, and then generate predictions from each model for the same test set. This is important to ensure that the three candidate models don't make predictions for different train/test splits (which would introduce an extra, unnecessary source of randomness.)

4) How to measure the quality of a model's out of sample prediction? Since you're fitting each model by least-squares, a natural measure of predictive ability is the sum of squared errors for the out-of-sample points. If $y_1, \ldots, y_m$ are the test-set points, and $\hat{y}_1, \ldots, \hat{y}_m$ are the predictions for these points generated by some model fit to the training data, then the mean of the squared out-of-sample errors is

$$\text{MSE} = (1/m) \sum_{k=1}^{m} (y_k - \hat{y}_k)^2 .$$

   Smaller values of MSE (mean squared error) are better!

5) MSE varies from one split to the next. Ideally you should characterize the between-model and within-model variation in MSE.

(3) Statistical adjustment, continued

On this problem, you will get to play amateur ecologist and investigate some hypotheses about speciation in the Galapagos Islands.

Consider the following three variables: (1) the number of distinct plant species that inhabit an island (Y); (2) the total area of the island (X_1) in square kilometers; and (3) the degree to which the island is flat or mountainous, as measured by the elevation of the highest point of the island (X_2) in meters. In the file "gala.csv" from the course website, you will find information on these variables (along with a few others) on 30 different islands in the Galapagos chain. (The three variables above are labeled Species, Area, and Elevation.)

(A) Before looking at the data, form your own hypotheses in response to the following two prompts. In each case, write a sentence or three explaining your reasons for thinking so.

1) As the land area of an island increases, *holding elevation constant*, I expect that the number of plant species that live there will . . ." Increase or decrease?
2) As the elevation of the highest point on an island increases, *holding total land area constant*, I expect that the number of plant species that live there will . . ." Increase or decrease?

(B) Now look at scatter plots of species count versus area, and species count versus elevation. Apply any transformations you deem necessary in order to make the assumption of linearity appropriate. Then fit individual linear models for each bivariate relationship (with species count as the response in each case), and summarize what you find. Report your model estimates and R^2 for each model.

(C) Suppose we wanted to estimate the following two quantities: i.) the marginal effect of area on species count, adjusting for the effects of elevation; and ii.) the marginal effect of elevation on species count, adjusting for the effects of area. You know that the residuals from a regression model implicitly adjust for the predictor variable. Thus you decide to try the following strategy.

i.) To estimate the marginal effect of area on species count, adjusting for elevation, you first compute the residuals from the species-versus-elevation regression (thereby estimating the "elevation effect" and also adjusting the species count for elevation). You then regress these residuals on area to get the "area effect" adjusted for the elevation effect.

ii.) To estimate the marginal effect of elevation on species count, adjusting for area, you first compute the residuals from the species-versus-area regression (thereby estimating the "area effect" and also adjusting the species count for area). You then regress these residuals on elevation to get the "elevation effect" adjusted for the area effect.

Do these two procedures provide the same estimates of the area and elevation effects, or different estimates? (That is, are they consistent with each other?) Explain what you think is going on here.

(Remember that all these regressions should happen on the appropriate transformed scale.)

(D) Can you think of a better way, using only regression models involving a single predictor, to get estimates of the quantities you care about? (That is, the elevation-adjusted area effect and the area-adjusted elevation effect.) Describe your approach in sufficient detail so that someone else could actually implement it. Hint: think about regression models that relate the two predictor variables.