How the North American Mammalian Paleofauna Got a Crook in Its Curve

36-402, Advanced Data Analysis

Due Tuesday, 21 February 2012, at 10:30 am

We continue to work with the fossil data set from homework 3. Paleontologists have suggested that the right curve relating change in log mass to ancestral log mass should be piece-wise linear: a downward-sloping line for small ancestral log masses, and flat for larger ancestral masses. In this problem set, you will fit that model, and see whether the data supports non-linear corrections.

You will first need to load the data set from homework 3, and add the column of change in log mass to the data frame. (See solutions that problem set, if necessary.)

1. (5 points) The piece-wise linear model predicts the following mean response as a function of the input x:

$$\hat{y}(x) = \begin{cases} a + bx & \text{if } x \leq d \\ c & \text{if } x \geq d \end{cases}$$

   Assuming that this is continuous at d, solve for a in terms of b, c and d. Explain why, in this application, it is reasonable to assume continuity.

2. (5 points) Write a function in R, called `deac`[1], that takes in a vector of numbers `x`, and three parameters `b`, `c`, and `d`, and returns the prediction of the model at each value of `x`.

   Check that your `deac` function is working properly by seeing that when $b = -1$, $c = 0.05$ and $d = 2$, giving `x=c(1,1.5,3)` gives outputs of

```
[1] 1.05 0.55 0.05
```

   Plot `deac`, with those parameters, over the range $(0, 4)$. Does it look right?

   Hints: `ifelse` for writing `deac`, `curve` for plotting.

3. (15 points total) Because `deac` varies nonlinearly with parameter d, we cannot estimate it by linear regression. However, we can still estimate the parameters by least squares. To do this, we need to write a function, make

[1] From the initials of the scientists who proposed this model; they didn't give it a name.

a starting guess about the parameters, and use the built-in optimization function `optim` (see recipe 13.2 in *The R Cookbook*).[2]. The following function fits the model to a data set by numerically minimizing the sum of squared errors:

```
my.start <- c(b=-1,c=0.2,d=10)
fit.a.deac <- function(data,start=my.start) {
  sse <- function(par) {
    preds <- deac(data$ln_old_mass,par[1],par[2],par[3])
    sum((data$delta_ln_mass - preds)^2)
  }
  fit <- optim(par=start,fn=sse,method="Nelder-Mead")
  coefficients <- fit$par
  fitted <- deac(data$ln_old_mass,coefficients[1],coefficients[2],
    coefficients[3])
  residuals <- data$delta_ln_mass - fitted
  mse <- mean(residuals^2)
  return(list(coefficients=coefficients,fitted=fitted,residuals=residuals,
    mse=mse,data=data))
}
```

(See online for the commented version; you'll want to source that, rather than typing this in and adding original errors.)

(a) (7) Explain what the inner function, `sse`, does.

(b) (8) What sort of output does `fit.a.deac` give — a vector, a list, an array, what? What do the various components of the output represent, in terms of the statistical problem?

4. (15 points) *Starting positions*

(a) (10) The code given above looks for a vector of initial parameters called `my.start`, if no other starting point is supplied. The line before the function makes up some values for `my.start`; they are bad ones. Find a better starting value for d by examining your non-parametric fit in homework 3. (If you couldn't get that to work, look at the solutions.) Get a rough guess for c by taking the average change in log mass over all animals whose ancestral log mass was greater than your guess for d. Get a rough guess for b by linearly regressing the change in log mass on the ancestral log mass for animals where the latter is less than your guess for d. Explain why this both guesses make sense.

[2] R has a built-in function, `nls`, for such "nonlinear least-squares" estimation, working more like `lm`. Unfortunately, `nls` can be flaky when the model doesn't have continuous derivatives, which is the case here. Besides, writing your own code builds character.

(b) (5) Re-define `my.start` to contain your initial guesses for b, c and d. Run `fit.a.deac` to get a fitted model, which you should call `nampd.deac`. Plot the fitted values as a function of log ancestral mass on a scatter-plot of change in log mass versus log ancestral mass.

5. (20 points) *Bootstrapping will continue until morale improves.* Use resampling of residuals, not cases, in both parts.

   (a) (10) Find bootstrap standard errors, and 95% confidence intervals, for the parameters b, c and d. Report all these quantities. Reporting more significant digits than is justified by statistical precision will cost you points.

   (b) (10) Find 95% bootstrap confidence bands for the fitted curve, and add them to your plot from the previous problem.

   Note: You can use the same resampled data-frames for both parts of this problem and the previous one, but it needs more clever programming. 1000 bootstrap replicates takes 1–2 minutes on my computer.

6. (30 points) *Testing parametric forms*

   (a) (5) Use `smooth.spline` to fit a smoothing spline to the data. Add the curve to the plot. Find the (in-sample) MSE of both the smoothing spline and the parametric model. Which fits better?

   (b) (3) Write a function to fit the smoothing spline to a data set. Check that it works by making sure it gives the right answer on the original data.

   (c) (6) Write a function to calculate the MSE of a fitted smoothing spline. Check that it works by making sure it gives the right answer on the original data.

   (d) (6) Write a function to take in a data set and return the difference in MSEs between the parametric model and the smoothing spline.

   (e) (8) Combine your functions to draw 1000 samples from the distribution of this test statistic, under the null hypothesis that the parametric model is right.

   (f) (2) What is the p-value of this test of the null hypothesis?

7. (10) Does this parametric model seem like an acceptable representation of the data? Justify your answer by referring to your work above.