Homework 4: But Does It Scale?

CS 201-A01, Spring 2021

This Homework was adapted from Homework 4 of Carnegie Mellon University's 36-350 course[1].

General instructions for homeworks: Upload both (i) the R Markdown file and (ii) the `nb.html` file to eCampus. You should give the commands to answer each question in its own code block. Each answer must be supported by written statements as well as any code used.

Background: In the previous lectures and lab, we began to look at user-written functions. For this assignment we will continue with a look at fitting models by optimizing error functions, and making user-written functions parts of larger pieces of code.

In lecture, we saw how to estimate the parameter a in a nonlinear model,

$$Y = y_0 N^a + \text{noise}$$

by minimizing the mean squared error

$$\text{MSE}(y_0, a) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - y_0 N_i^a)^2.$$

We did this by approximating the derivative of the MSE, and adjusting a by an amount proportional to that, stopping when the derivative became small. Our procedure assumed we knew y_0. In this assignment, we will use a built-in R function to estimate both parameters at once; it uses a fancier version of the same idea.

You will estimate the power-law scaling model using the data alluded to in lecture, available in the file `gmp.dat`, which contains data for 2006.

0. Store the data in `gmp.dat` in a variable called `gmp` and create a new column in `gmp` called `pop` with the populations of the cities using their Gross Metropolitan Product (`gmp`) and Per Capita Gross Metropolitan Product (`pcgmp`).
1. Plot the data as in lecture, with per capita GMP on the y-axis and population on the x-axis. Add the curve function with the default values provided in lecture. Add two more curves corresponding to $a = 0.1$ and $a = 0.15$; use the `col` option to give each curve a different color (of your choice).
2. Write a function, called `mse()`, which calculates the mean squared error of the model on a given data set. `mse()` should take three arguments::
   - a numeric vector of length two, the first component standing for y_0 and the second for a;
   - a numeric vector containing the values of N; and
   - a numeric vector containing the values of Y.

   The latter two arguments should have as their default values the columns `pop` and `pcgmp` (respectively) from the `gmp` data frame from lecture. The function should return a single numerical value.

[1]Shalizi, C. R. and Thomas, A. C. (2014), "Statistical Computing 36-350: Beginning to Advanced Techniques in R", http://www.stat.cmu.edu/ cshalizi/statcomp/14

Your function may not use `for` or any other loop. Check that, with the default data, you get the following values.

```
> mse(c(6611,0.15))
[1] 207057513
> mse(c(5000,0.10))
[1] 298459914
```

3. R has several built-in functions for numerical optimization. These use ideas similar to (but more sophisticated than) steepest descent to find a minima of a function. For this homework, we will use `optim()`. `optim()` takes two required arguments: a starting value for the arguments of the function and the function itself. You should also set the `method` argument of `optim()` to `"BFGS"`, which will use the Broyden–Fletcher–Goldfarb–Shanno algorithm to perform the minimization.

   Run `optim()` three times with your function `mse()` and three **different** starting value pairs for `y0` and `a` as in

   ```
   optim(c(y0=6611,a=1/8), mse, method = "BFGS")
   ```

   What do the quantities `par` and `value` represent? What values does `optim()` return for these in each call you made?

4. Using `optim()` and the `mse()` function you wrote, write a function, `plm()`, which estimates the parameters y_0 and a of the model by minimizing the mean squared error. It should take the following arguments:

   - an initial guess for y_0;
   - an initial guess for a;
   - a vector containing the N values; and
   - a vector containing the Y values.

   All arguments except the initial guesses should have suitable default values. It should return a **named list** with the following components:

   - the final guess for y_0;
   - the final guess for a; and
   - the final value of the MSE.

   Your function must call the functions you wrote in earlier questions (it should not repeat their code), and the appropriate arguments to `plm()` should be passed on to them.

   What parameter estimate do you get when starting from $y_0 = 6611$ and $a = 0.15$? From $y_0 = 5000$ and $a = 0.10$? If these are not the same, why do they differ? Which estimate has the lower MSE?

5. The file `gmp-2013.dat` contains measurements for 2013. Load it, and use `plm()` to estimate the parameters of the model for 2013. Have the parameters of the model changed significantly[2]?

[2] In the colloquial, not the statistical, sense of the word 'significantly'.