Homework 4: Free Soil

36-402, Spring 2015

Due at 11:59 pm on Monday, 9 February 2015

AGENDA: Practice writing, testing, and debugging simple R functions. Practice decomposing a big computational problem into a bunch of small, inter-locking functions. Practice estimating a categorical contrast. Practice with weighted least squares. Practice with bootstrapping. Finally, an early observance of Lincoln's birthday.

GRADING: The problems add up to 90 points. The remaining 10 points are reserved for style and clarity. (See rubric at the end of this assignment.) This will apply to all future homework until further notice.

Recall that equation for the standard error of a proportion, when we observe a binomial with n trials and success probability p:

$$\sqrt{\frac{p(1-p)}{n}} \tag{1}$$

Further recall the estimated standard error in an observed proportion $\hat{p}$:

$$\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \tag{2}$$

Recall, finally, that the `Mobility` variable from homework 1 was an observed proportion, the fraction of children born into the bottom fifth of the income distribution who make their way to the top fifth of the distribution by age 30.

Load the data set from homework 1 as a data frame named `mobility`. We will only need three columns, `Mobility`, `Population` and `State`, though you may also want to keep `Name` for debugging purposes. Do not remove any row from the data frame which has complete values for these variables.

1. (15) Write a function, `se.prop`, to calculate the standard error for proportions. It should take a vector of proportions, `p`, and a vector of trial numbers, `n`, and return a vector of standard errors.
   (a) (2) Construct a test case to check that `se.prop` gives the right answer when $p = 0.5$, $n = 1$.
   (b) (2) Construct a test case to check that when `se.prop` is given a vector of different `n`'s, all with the same p (not equal to 0 or 1), the answers are proportional to $1/\sqrt{n}$.

(c) (2) Construct a test case to check that when $p = 0$, the returned value is always `0`, for multiple n.

(d) (2) Construct a test case to check that when $p = 1$, the returned value is always `0`, for multiple n.

(e) (2) Construct a test case to check that when given a vector `p` of mixed `0`s and `1`s, the returned vector has all `0`s, for multiple `n`.

(f) (2) Construct a test case to check that when given a vector of different, non-extreme values for `p`, and a constant `n`, the entries of the returned vector are proportional to $\sqrt{p(1-p)}$.

(g) (2) Check that `se.prop` works properly when `p=c(0.3,0.8)` and `n=c(12,72)`. This includes working out what the proper answers should be.

(h) (1) Explain whether your code implements Eq. 1 or Eq. 2.

2. (10)

   (a) (3) Use `se.prop` to calculate the standard error of the mobility for each community in the data from homework 1; report the summary statistics.

   (b) (1) Plot the histogram of the standard errors.

   (c) (2) Make a scatter-plot of the standard errors vs. population.

   (d) (2) Make a scatter-plot of the standard errors vs. mobility.

   (e) (2) How reliable were the inferential statistics you calculated in homework 1?

3. (15)

   (a) (5) Write a function, `WSE`, to calculate weighted mean squared error. It should take as arguments `predicted`, a vector of predicted values; `observed`, a vector of observed values; and `weights`, a vector of weights. It should return a single real number, the weighted mean squared error. Mathematically, that is to say, it should find

$$\frac{\sum_{i=1}^{n} w_i \left(y_i - \hat{y}_i\right)^2}{\sum_{i'=1}^{n} w_{i'}}$$

   Make the default value for `observed` the `Mobility` column of the data, and the default values for `weights` equal to one over the squares of the standard errors in `Mobility` from the previous problem. *Hint:* You could write this using a `for` loop, or even two of them, but there are more elegant ways.

   (b) (3) Check that `WSE` works properly when `predicted` is `c(0.15,0.05)`, `observed` is `c(0.14,0.07)`, and `weights` is `c(0.01, 0.42)`. (This includes working out what the right answer should be.)

   (c) (2) Create three modified versions of this test case, each changing one of the three arguments, and make sure that your function works correctly on all three.

(d) (2) Explain why, for modeling mobility, the weights should be the inverse square standard errors.

(e) (3) Check that `WSE` returns the MSE when all the weights are equal. (They will *not* be equal for those default values.)

4. (10)

   (a) (5) Write a function, `dixie`, which reads in a vector of state names (in the form used in the mobility data set), and returns a binary vector, 1 if the state was part of the Confederacy during the US civil war, and 0 otherwise.

   (b) (5) Check that it gives the correct results when applied to a vector of the 50 state names and the District of Columbia.

5. (10) Write a function, `dixie.fit`, which takes two arguments: a data frame with a column named `State`, and a vector of length two, `levels`. It should test, for each row, whether the state was in the Confederacy (using `dixie`), and if so return the first element of `levels`, and if not, return the second element. Check that it works correctly when `levels=c(1,0)`. Explain how you know that is the correct behavior.

6. (10) Write a function, `dixie.WSE`, which takes as input `levels`, without default, and a data frame, defaulting to `mobility`. It should predict the mobility level for each city based on whether it was in the Confederacy or not, using the function `dixie.fit`, and return the weighted squared error, using `WSE`, with the actual values of `Mobility` as the response and weights based on their standard errors. For full credit, call, do not re-write, the functions from the earlier problems.

   Construct a test case using a data frame of four rows to check that is working properly, when `levels=c(0.01,0.15)`.

7. (5) Optimize the weighted squared error for this two-parameter model, starting from the initial guess that the mobility level for the former Confederacy is 0.01, while that for the rest of the country is 0.15. Report the best-fitting values of `levels`.

   Hint: See recipe 13.2 in *The R Cookbook*.

8. (10) Turn the optimization from the previous problem into a function, which takes as arguments a data frame (with default equal to `mobility`) and an initial guess at `levels` (with default equal to `c(0.01,0.15)`), and returns the fitted values of `levels` (and nothing else). Check that running it with the defaults reproduces your answer from the previous problem. Check that you get a different answer if you remove the first half of the data frame.

9. (5) Use resampling of rows to give standard errors for `levels`.

RUBRIC (10): The text is laid out cleanly, with clear divisions between problems and sub-problems. The writing itself is well-organized, free of grammatical and other mechanical errors, and easy to follow. Figures and tables are easy to read, with informative captions, axis labels and legends, and are placed near the text of the corresponding problems. All quantitative and mathematical claims are supported by appropriate derivations, included in the text, or calculations in code. Numerical results are reported to appropriate precision. Code is either properly integrated with a tool like R Markdown or knitr, or included as a separate `.R` file. In the latter case, the code is clearly divided into sections referring to particular problems. In either case, the code is indented, commented, and uses meaningful names. All code is relevant to the text; there are no dangling or useless commands. All parts of all problems are answered with actual coherent sentences, and never with raw computer code or its output.

EXTRA CREDIT (10): Show, mathematically, that the optimal values for `levels` are always given by two weighted averages of `Mobility`. Show how to find them by two calls to `weighted.average`, without using `WSE`, `dixie.fit`, `dixie.WSE`, or any optimization function. For full extra credit, check that code implementing this matches the answer you obtained above.