Lab 2: Only the Answers Have Changed

36-350, Statistical Computing

6 September 2013

Agenda: Practicing iteration; practicing re-writing code; checking how reliable random methods are.

General instructions for labs: Upload a single plain text file (not Word), named with your andrew ID, to Blackboard. For questions which ask you to do something, give the command you use to do it. For questions which ask you to calculate something, give both the answer (to reasonable precision), and the commands you used to get it. For questions which ask you to re-write code, give the complete code, not just a description of the changes you'd make. For questions which ask you to explain, write out an explanation in coherent, complete sentences. (You will be graded on your written explanation, not what you might say to the TA.) Clearly indicate which answer goes with which question.

The file `http://www.stat.cmu.edu/~cshalizi/statcomp/13/hw/02/hw-02.R` on the class website contains the code for the resource allocation example from lecture 3. Download it.

1. (30) *Reproducibility?*
   (a) (10) Set the initial output levels to 30 cars and 20 trucks (as in the slides) and run the code. What is the final output plan (`output`)? What is the final demand for resources (`needed`)? Is the plan within budget and within the slack? How many iterations did it take to converge (`passes`)?
   (b) (10) Re-set the initial plan to 30 cars and 20 trucks, re-run the code and report the final values of `output`, `needed`, and `passes`.
   (c) (10) Explain why your answers for 1a and 1b differ from those given in the slides, and from those given on the slides, even though you're running exactly the same code. *Hint:* Your answers had better differ.
2. (40) *Layering iteration* We want to be able to re-run the resource allocation code multiple times and store the results, to get a sense of the average behavior and how variable it is.

(a) (20) Write a `for` loop which re-runs the resource allocation code a user-set number of times, and store the results. Store all the `output` vectors in one array, all the `needed` vectors in another array, and all the `passes` numbers in a vector.

Your could should look something like this:

```
runs <- 5
# Set up the arrays and vectors for storing results
for (run in 1:runs) {
   # Reset the initial conditions
   # Run the main loop
   # Store the results in the arrays
}
```

Your code needs to fill in the lines marked with `#`.

(b) (10) Run your `for` loop. What are the mean and standard deviations of the output levels? (You should report two means and two standard deviations.)

(c) (10) What are the means and standard deviations of the resources needed? (Again, you should report two means and two standard deviations.) The standard deviations should be less than the `slack` levels — why?

3. (30) *Changing starting positions*

(a) (10) Modify your `for` loop, so that the initial output levels are 5 cars and 5 trucks, and re-run it. What are the mean and standard deviations of the final output levels?

(b) (10) Re-run your code with the initial output levels set to 12 cars and 19 trucks. What are the mean and standard deviations of the final output levels?

(c) (10) Does the final plan depend significantly on the initial plan? Explain, using your answers to earlier questions.