Lab 1

Statistical Computing, 36-350

Friday September 4, 2015

Today's agenda: manipulating data objects; using built-in functions, doing numerical calculations, and basic plots; reinforcing core probabilistic ideas.

General instructions for labs. Upload an R Markdown file, named .Rmd", to Blackboard. You will give the commands to answer each question in its own code block, which will also produce plots that will be automatically embedded in the output file. Each answer must be supported by written statements as well as any code used. Include the name of your lab partner at the top of the file.

R Markdown setup. Open a new R Markdown file; set the output to HTML mode and click "Knit HTML". This should produce a web page with the knitting procedure executing your code blocks. You can edit this new file to produce your lab submission. Alternatively, you can start from the lab's R Markdown file posted on the course website, as a template.

Background

The binomial distribution $\text{Bin}(m, p)$ is defined by the number of successes in m independent trials, each have probability p of success. Think of flipping a coin m times, where the coin is weighted to have probability p of landing on heads.

The R function `rbinom` generates random variables with a binomial distribution. E.g.,

```
rbinom(n=20, size=10, prob=0.5)
```

produces 20 observations from $\text{Bin}(10, 0.5)$.

Part I

1. Generate 250 random values from the $\text{Bin}(10, 0.5)$ distribution, and store them in a vector called `bin.draws.0.5`. Find the mean and standard deviation of `bin.draws.0.5`.

2. Repeat, but change the probability of success to 0.2, 0.3, 0.4, 0.6, 0.7, and 0.8, storing the results in vectors called `bin.draws.0.2`, `bin.draws.0.3`, `bin.draws.0.4.`, `bin.draws.0.6`, `bin.draws.0.7` and `bin.draws.0.8`.

3. The function `plot()` is the generic function in R for the visual display of data. `hist()` is a function that specifically produces a histogram display.

    a. Use the `hist()` function to produce a histogram of the random draws from your fair binomial distribution, stored in `bin.draws.0.5`.
    b. Use `plot()` with `bin.draws.0.5` to display the random values from your standard distribution, in order they appear in the vector.
    c. Now use `plot()` with two arguments—any two of your other stored vectors of random draws—to create a scatterplot of the two vectors against each other.

4. We'd now like to compare the properties of each of our vectors. Begin by creating a vector of length of length 7, called `bin.means`, and then fill its entries with the means of the 7 distributions we've created, in increasing order according to their underlying probabilities of success. Using this and other similarly created vectors, create the following scatterplots. Explain in words, for each, what's going on.

a. The 7 means versus the 7 probabilities used to generate the draws.
b. The standard deviations versus the probabilities.
c. The means versus the standard deviations

5. Reproduce the means and standard deviations calculations (not the plots) that you performed in the previous part, but do so by first creating a matrix of dimension 250 x 7, called `bin.matrix`, whose columns contain the 7 seven distributions we've created. To calculate the means and standard deviations from `bin.matrix`, you can use `apply()`.

Part II

6. R's capacity for data and computation is very large compared to what was available 10 years ago.
   a. To show this, generate 2.5 million numbers from Bin$(1 \times 10^6, 0.5)$ distribution and store them in a vector called `big.bin.draws`. Calculate the mean and standard deviation of this vector.
   b. Create a new vector, called `big.bin.draws.standardized`, which is given by taking `big.bin.draws`, subtracting off its mean, and then dividing by its standard deviation. Calculate the mean and standard deviation of `big.bin.draws.standardized` (these should be 0 and 1, respectively; if not, you've made a mistake somewhere).
   c. Plot a histogram of `big.bin.draws.standardized`. To increase the number of histogram bars, set the `breaks` argument in the `hist()` function (e.g., set `breaks=500`).
   d. What does the shape of this histogram appear to be? Is this surprising? What could explain this phenomenon? (Hint: rhymes with "Mental Gimmick Serum" ...)
   e. Calculate the proportion of times that an element of `big.bin.draws.standardized` exceeds 1.644854. Is this close to 0.05?
   f. Either by simulation, or via a built-in R function, compute the probability that a standard normal random variable exceeds 1.644854. Is this close to 0.05? (Hint: for either approach, it would be helpful to look at the help file for the `rnorm` function.)

Part III (Bonus, optional)

7. Let's push R's computational engine a little harder.
   a. Design an expression to generate 50 million numbers from Bin$(10 \times 10^6, 50 \times 10^{-8})$, to be saved in a vector called `huge.bin.draws`, but do not evaluate this command yet. Then ask your lab partner to name one of Justin Bieber's songs and simultaneously evaluate your R command that defines `huge.bin.draws`. Which finished first, R or your partner? (Note: your partner cannot really win this challenge. Even if he/she finishes first, he/she still loses.)
   b. Plot a histogram of of `huge.bin.draws`, again with a large setting of the `breaks` argument (say, `breaks=500`). Describe what you see; is this surprising?
   c. What distribution does this look like to you? (Hint: fishy, fishy, fishy ...)