Lab 2

Statistical Computing, 36-350

Friday September 11, 2015

Today's agenda: manipulation of matrices and lists; more practice with random number generation, and more reinforcing of core probabilistic and mathematical 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 homework submission. Alternatively, you can start from the lab's R Markdown file posted on the course website, as a template.

Part I: Matrices

Many important areas of statistics, machine learning, and engineering use and study matrices that are filled with i.i.d. (independent and identically distributed) Gaussian entries. The Gaussian distribution is a truly special one (and central one—get it?).

1. Generate 50,000 random values from a standard Gaussian distribution, denoted $N(0, 1)$, and from these fill out a 1000 x 50 matrix called `mat`.

2. What is the mean of the 7th column of `mat`? The mean of the 37th column? What would you expect them to be, theoretically?

3. Compute the column means of `mat`, and store as a vector called `mat.col.means`.
   Check that the mean of the column means is equal to the overall mean of the matrix.

4. Compute the sum of squares of values in each row of `mat`, and call the resulting vector `mat.row.chisq`. Given what you know about how you constructed `mat`, what distribution do the values in `mat.row.chisq` have? (Hint: take a look at the name of this vector!)

5. For each entry in the vector `mat.row.chisq`, subtract of 50 and divide by $\sqrt{100}$, and call the result `mat.row.chisq.standardized`. Plot a histogram of `mat.row.chisq.standardized`, using `hist()`, with a high value of the `breaks` argument. What does this distribution look like? Why? And what purpose did the values 50 (the amount we subtracted off) and $\sqrt{100}$ (the amount we divided by) serve?

6. Using the function `cor()`, compute the correlation of every column of `mat` with every other column, and store the answer as `mat.cor`. What dimension is `mat.cor`? What is the highest positive correlation between a pair of (different) columns? Which pair of achieves this highest positive correlation? Now compute a heatmap of the correlation matrix, using the `image()` function. Does this picture match what you'd expect? Why or why not?

Part II: Lists

The singular value decomposition (SVD) is something that you likely encountered in your linear algebra class. Though it may have seemed abstract when you learned it, the SVD actually plays a really big role in much of advanced statistical modeling. When you learn things like ridge regression, principal components analysis, smoothing splines, etc.—all of these things (and more!) can be understood from the perspective of the SVD.

7. Compute a singular value decomposition of the matrix `mat`, using the `svd()` function, and store the result in a variable called `mat.svd`. What is the class of `mat.svd`? What are the names of its components?

8. Mathematically, if we denote `mat` by a matrix M, then its SVD can be written as $M = UDV^T$, where D is a diagonal matrix, containing what are called the *singular values* of M. Using the components of `svd.mat`, form the three-term matrix product UDV^T. (Hint: you will need to use the `diag()` function in order to turn a vector into a diagonal matrix, and the `t()` function in order to perform the matrix transpose operation.) Check that is this matrix product is actually equal to the original matrix `mat`. (Hint: to check whether all entries of one matrix are equal to the other, you can use `all.equal()`.)

9. Again denoting `mat` as M, mathematically, compute the cross-product matrix M^TM, and call this `mat.cross`. Now compute an eigendecomposition of `mat.cross`, using the `eigen()` function, and store the results in a variable called `mat.eigen`. What is the class of `mat.eigen`? What are the names of its components?

10. Check that the squared singular values of `mat` are equal to the eigenvalues of `mat.cross`. Bonus: can you explain why this is the case? (This requires a bit of advanced linear algebra, so don't worry about it if you don't see why this is true...)

Part III (Bonus, optional)

11. Verify that the sum of the squared singular values of `mat` is equal to the sum of its squared entries. Can you explain why this is the case? (This requires a bit of advanced linear algebra, so don't worry about it if you don't see why this is true...)

12. Generate a 2000 x 2000 matrix filled with $N(0, 1)$ entries, called `big.mat`. *Symmetrize* this matrix by defining `big.mat.sym` to be equal to the sum of `big.mat` and its transpose.

13. We are going to want to compute an eigendecomposition of `big.mat.sym`, saved as `big.mat.sym.eigen`, but this is going to be costly; to save on computation time, since we only want to look at the eigenvalues, be sure to set the argument `only.values=TRUE` in the call to `eigen()`. Now plot a histogram of the eigenvalues, with a large setting of the `breaks` argument. Does this seem to form a definitive shape? What is it?

14. Note: the above demonstrated a very famous and fundamental result in random matrix theory, called *Wigner's Semicircle Law*. Demonstrate that this same law holds, even when the distribution of entries is changed from $N(0, 1)$ to something else (for example, try Bernoulli entries).