STAT 339: HOMEWORK 0 (CODING AND MATRIX ALGEBRA WARMUP)

DUE ON BLACKBOARD BY START OF CLASS ON MON. FEB. 6

Instructions. This assignment is optional and does not count for a grade. You should definitely do it if you do not have programming experience or if you are rusty / could use a refresher. You are welcome to pick and choose the parts that you think will be useful to you (e.g., just the coding parts or just the linear algebra part).

If you want feedback on your work, turn in your writeup and code to Blackboard as an archive file (e.g., `.zip`, `.tar`, `.gz`) by the start of class on Monday, 2/6.

You may use any language you like to do this assignment — the tasks are stated in a language-neutral way. I recommend one of MATLAB/Octave, Python, or R, with MATLAB/Octave as a default choice if you have no experience in any of them, as it is in some ways the most 'user-friendly' of the three, and there is example code provided with both the main textbook and the supplementary textbook that we will rely on most. Alternatively, you could teach yourself Python, which is probably a better long-term investment, but it will require more time up front. There are some Python resources on the course webpage (under "Resources").

If you are starting from scratch and using Octave, I suggest starting by downloading Octave for your operating system (unlike MATLAB, it is free) and working through the tutorial, including exercises. Links for both are on the course webpage (under "Resources"). The complete documentation is also linked there if you need to look up something specific (or you can Google `octave thingyouwanttodo`, which most often points you to the manual — for example `octave histogram` to find the function for a histogram). If you know the name of the function you want (e.g., `hist`) you can type `help hist` at the Octave prompt to see documentation on how to use the function.

You may also use any typesetting software to prepare your writeup, but the final document should be a PDF. LaTeXis encouraged; a reproducible research format in which code is embedded into the document (e.g., knitr, RMarkdown, Jupyter or IPython Notebook) is even more encouraged. Sadly I am not aware of a reproducible research format that supports MATLAB/Octave code.

Date: Last Revised: January 26, 2017.

0. Create a project directory for this homework to keep your files organized, and set it to be your working directory. Your archive will be this directory.

1. **Creating some simple plots.** Create each of the following plots and embed them in your writeup (if you are not using a reproducible research format, save them as a file first and import the file into your document).

   (a) Produce a plot of the floor function, $f(x) = \lfloor x \rfloor$ for x values in the range 0 to 10. (The floor function "rounds down" to the nearest integer.) Include axis labels.

   (b) Plot three different lines on the same plot (pick your own intercepts and slopes). Distinguish them by color and include a legend.

   (c) Produce a plot of the unit circle. (Hint: Create the $y > 0$ and $y < 0$ parts separately)

2. **Matrix manipulation and loops**

   (a) Get data for this exercise from my website: `http://colindawson.net/data/big_matrix.txt`. You can either read it directly into your program from the web, or first download the file into your working directory and read it in locally. One of these may be easier than the other depending on the language you are using. Save it as a variable called `A`. (In Octave you can use `load` to do this.)

   (b) Create variables `R` and `C` that hold the number of rows and columns in the matrix.

   (c) Normalize `A` so that the values are all between 0 and 1.

   (d) Visualize `A` as a greyscale image where 0 is black and 1 is white. You may find that most of the image shows up as black; if so, apply a nonlinear transformation to the values in the matrix to stretch the values near 0 and compress them near 1.

   (e) Create a copy of `A` (e.g., `B`), and use a for loop to replace the (i, j) entry of `B` by 0 (or whatever your transformation returns for 0) whenever i and j are both evenly divisible by 5. Visualize `B` (you should see a lattice of black pixels that occupies $1/25$ of the image in total).

(f) Make the entries that were previously 0 (or transformed 0) into 1s (or transformed 1). Do this without writing any explicit loops (Hint: select the entries of interest using an index expression).

(g) Save the new matrix as a file.

3. **Creating and running a script.** Create a script file in your working directory and place in it all the code you need to read in and write out the matrices in problem 2. Verify that it runs by sourcing it from a fresh workspace (or from the shell command line).

4. **Creating a function.** In a separate file, define a function that does what you did in problem 2 with some more flexibility. Your function should take three arguments: the name of the file to read data from, the name of a file to write output to, and the step size for the lattice. It should also contain a clear docstring (a special comment inside the function definition the text of which displays if a user asks for help). Create a script file that gets the definition of your function from the first file and calls it with two different step sizes, writing to two different output files.

5. **Random numbers and Counting.** Define a function that simulates throwing two d-sided dice n times and returns the proportion of the time that the values on the dice sum to y over the n rolls. Your function should take d, n, and y as arguments, and return the sample proportion. (Optional: have your function provide default values for d and/or y) In a script, first set the seed of the random number generator to 42, then call your function 10 times in a row with $d = 6$, $n = 100$ and $y = 12$. How often did you get the same value? Set the seed to 42 again and repeat the 10 calls. How did the 10 values you got compare to the 10 values you got the first time? Explain why it might be useful to be able to control the random seed when testing and debugging a machine larning algorithm.

6. **Vector Arithmetic.** Define a function that generates two 3×1 vectors, a and b, with random entries (your choice of distribution), (if using Python, use `numpy` arrays not plain arrays), and returns a list/tuple/struct consisting of: a, b, $a + b$, $a \circ b$, and $a \cdot b$ ($\circ$ denotes the elementwise product and $\cdot$ denotes the dot product;

$a \cdot b = a^\mathsf{T} b := \sum_{i=1}^{3} a_i b_i$). Set the random seed to 37 and call your function, including the five output values in your writeup as verbatim monospace text and in formatted mathematical notation.

7. **Matrix Arithmetic.** Create a modified version of your function from the last problem that also creates a random 3×3 matrix, X, and returns a list/tuple/struct consisting of a, b, X, $a^\mathsf{T} X$, $a^\mathsf{T} X b$, X^{-1} and $(X^\mathsf{T} X)^{-1}$. Set the seed to 37 again and call your function, including the output values in your writeup in two ways as above. Verify for one representative element of the output that matrix multiplication worked as it should; i.e. $(AB)_{ij} = \sum_k A_{ik} B_{kj}$, where A_{ij} is the entry in row i and column j of A.

8. **Linear Algebra Warmup.** (No coding for this problem)

   (a) (FCML Exercise 1.3) Show that:

   $$\mathbf{w}^\top \mathbf{X}^\top \mathbf{X} \mathbf{w} = w_0^2 \left(\sum_{n=1}^{N} x_{n1}^2 \right) + 2 w_0 w_1 \left(\sum_{n=1}^{N} x_{n1} x_{n2} \right) + w_1^2 \left(\sum_{n=1}^{N} x_{n2}^2 \right),$$

   where

   $$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \mathbf{X} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \\ \vdots & \vdots \\ x_{N1} & x_{N2} \end{bmatrix}.$$

   Recall the definition of matrix multiplication: $(AB)_{ij} = \sum_k A_{ik} B_{kj}$, where, for example, A_{ij} represents the entry in the ith row and jth column of the matrix A.

   (Hint – it's probably easiest to do the $\mathbf{X}^\top \mathbf{X}$ first, which will allow you to work with smaller matrices!)

   (b) Using $\mathbf{w}$ and $\mathbf{X}$ as defined in the previous exercise, show that $(\mathbf{X}\mathbf{w})^\top = \mathbf{w}^\top \mathbf{X}^\top$ by multiplying out both sides.