Homework 3 **PSY8219**
Due September 17 in class **Fall 2018**
20 points

Q1. Plot the probability density function of a chi-squared distribution. As I'm sure some of you know, the chi-squared distribution is widely used in categorical data analysis. The equation for the probability density function of the chi-squared distribution is

$$f(x) = \frac{1}{2^{\frac{k}{2}}\,\Gamma\left(\frac{k}{2}\right)} x^{\frac{k}{2}-1} e^{-\frac{x}{2}}$$

For anyone who doesn't know, *e* is an important mathematical constant that shows up all over the place in mathematics and statistics. Like `pi`, it's just a number. It's equal to 2.71828182846… In Matlab, there is a function `exp(x)` which computes *e* raised to some power *x* (check the documentation on `exp` in Matlab).

The gamma function $\Gamma(x)$ is also ubiquitous in mathematics and statistics. In Matlab, there is a function `gamma(x)` you can use. (Compute *Γ(x)* where *x* is an integer between 1 and 10 and see if you can figure out what *Γ(x)* is computing when you give it an integer. *Γ(x)* generalizes beyond what you might be already familiar with to all real numbers, not just integers.)

In the equation for the chi-squared, *x* if the value you are determining the likelihood of, *f(x)* is the computed likelihood, and *k* is the degrees of freedom. Unlike a distribution like the normal, for the chi-square *x* is constrained to be greater than or equal to zero (the chi-squared distribution is undefined for negative values of *x*).

Create three plots of the chi-squared distribution for three values of the degrees of freedom, *k*=2, 4, and 9; put all three on the same graph using the `plot()` command as shown in class. Make sure your plots reasonably show the full shape of the distributions (obviously, you can't show the entire distribution in its because it is defined in the range between 0 and infinity, but you can show a reasonable range to see its full shape). As a check, feel free to look at the Wikipedia page for the probability distribution for the Chi-squared to see its shape for different values of *k*.

Specifically, to do this, you will need to generate a fine-grained range of *x* values, then using the formula, calculate the corresponding fine-grained *f(x)* values. I want you to do these calculations two ways (for each method, creating a separate plot for the three values of *k*):

Q1a. Use `for` loop(s) to calculate the value of *f(x)* for each individual value of *x*.

Q1b. Instead of a `for` loop, calculate the full array of *f(x)* values from the full array of *x* values in one line of Matlab code.

Q2. Create Matlab code that performs matrix multiplication without using the built-in operators for performing matrix multiplication or calculating a dot product. Instead, I want you to use scalar multiplications and additions, `if` statements, and `for` or `while` loops. You will need to review class slides from last week and this week. Do not try to put the code in a function call (we will use functions later); a script is sufficient.

Your code should work for any matrices A and B that are defined before this code. When writing your code, it is okay to assume that it calculates the matrix operation A * B (in that order).

You should first check using conditionals that the dimensions of the matrices make matrix multiplication possible; if not, you should print an error message to the Command Window.

Then you should calculate the value of C = A * B using your code. Again, only use scalar additions and multiplications (no built-in functions or matrix or vector operations are allowed).

Compare your long-form calculated solution C = A * B to that using the built-in matrix multiplication operation `D = A * B`. C and D should be the same value. Show that they are.

Q3. In Matlab code, convince yourself (and me) that the inverse of a matrix is not the same as the matrix of the inverse (1/x) of each of its elements. First, I want you to show me that they are not equal to one another. Second, I also want you to show me that the former (the inverse of a matrix) and not the latter (the matrix of the inverse of its elements) satisfies the key property for what it means to be an inverse of a matrix.

Note that for some matrices it is impossible to calculate an inverse. Matlab will often signal this by saying that the matrix is "singular" (sometimes called "degenerate" as well). If you find that your matrix is singular (noninvertible) try another matrix until you find one that can be inverted.

Q4. I want you to a create data structure that holds the first name and last name of the subjects in an experiment:

Mary Smith
Robert Jones
Frank Tinny
Larry Jacobson

Demonstrate how you would reference only the first name of all the participants, then demonstrate how you would reference the full name of one of the participants. Do so without hard-coding the solution.

Next use an fprintf statement to print the full name of the 4th subject.

Q5. Learn more about structures (`struct` arrays) in Matlab. I went over this quickly in class; look over the class slides. In the slides I also have links to some online sources. You can also read about structures in the Attaway textbook.

Imagine that I am currently holding data for my experiment in a cell array as follows:

```
data{1,1} = 'Frank Jones';            % name
data{1,2} = 23;                       % age
data{1,3} = [24 26 21 22];            % BDI score each month
data{1,4} = [178 175 181 182];        % weight each month
data{2,1} = 'Bob Mills';              % name
data{2,2} = 26;                       % age
data{2,3} = [28 31 30 32];            % BDI score each month
data{2,4} = [154 152 156 153];        % weight each month
data{3,1} = 'Jim Russo';              % name
data{3,2} = 32;                       % age
data{3,3} = [21 22 19 21];            % BDI score each month
data{3,4} = [198 194 195 196];        % weight each month
```

a) Recreate this data using a structure (`struct array`) instead.

b) Write Matlab code that calculates the average BDI score for each participant in the `cell arrays`. It should be general enough so that the same piece of code will work even if `data` contained more than 3 subjects.

c) Write Matlab code that calculates the average BDI score for each participant in the `struct arrays`. It should be general enough so that the same piece of code will work even if `data` contained more than 3 subjects.

(For b. and c., see if you can do each of these in one line of Matlab code for a small bonus.)

Unexcused late assignments will be penalized 10% for every 24 hours late, starting from the time class ends, for a maximum of two days, after which they will earn a 0.