IPBR Problem Set 3

due July 5th, 2017

For feedback, submit this problem set by e-mail to squyres@g.harvard.edu by the start of lecture on July 5. You should submit all of the code (scripts and functions) that you write, as well as the figures that your code generates. Remember to comment your code!

Question 1: Data Handling and Plotting

Make a new script and save it as pset3_q1_(your initials).

a. Using the equation `y = (1-x.^2)+(x.^2).^(2/3)`, determine the value of y for x values between -1 and 1, counting by increments of 0.05. Store the results in a vector called y. (Note that in MATLAB, you need to use the notation .^ to raise each element in an array to an exponent. The notation ^ refers to taking the exponent of the entire matrix.)

b. Use the find function to find the index of the element -0.55 in x. Use this index to find the value of y when x = -0.55.

c. Use logical indexing to find the elements in y that are greater than 1.1. Use this information to identify the elements in x that correspond to these y values.

d. Plot y vs x. Add a title, x-axis label, and y-axis label to your graph.

e. Change the color of the plotted line to red, and add triangle markers to your graph. (Hint: https://www.mathworks.com/help/matlab/ref/linespec.html)

f. Save the images as figure_1f_(your initials).tiff

g. Using the equation `k = abs(x)`, determine the value of y for x values between -1 and 1, counting by increments of 0.05. Store the results in a vector called y.

h. Plot the result on the same graph as the previous plot. Make sure to use the MATLAB command `hold on` so that the previous plot is not overwritten.

i. Change the axis so that the x-axis ranges from -2 to 2, and the y-axis from -0.1 to 1.2. Add a new title.

j. Save this as figure_1j_(your initials).tiff

Question 2: Function Design

In this section, you will be constructing a function to take the square root of any positive, rational number greater than 1. The function should have a single input and output; the output should have accuracy to the thousandths place. You **cannot** use MATLAB's **`sqrt`** function in your function.

Example:

```
out = mattsqrt(30.45)
out = 5.518
```

Per usual, there are many different approaches to arrive at the same solution here. The approach that I used consisted of **For** loops and **conditional** statements to first find the integers that the solution is between. For an input of **30.45**, my function would first determine that the solution is between 5 and 6. Then using the same approach (**For** loops and **conditionals**) my function would find the tenths place, then hundredths, and finally the thousandths place.

You may want to use MATLAB's **`break`** command in this exercise. The **`break`** command will terminate the execution of a loop- see the example below. You could also consider using a **While** loop to achieve the same thing.

```
for i = 1:10
    if i*5 == 20
        solution = i
        break
    end
end
```

Save your function as pset3_q2_(your initials).

Challenge!

This function requires that the user give an input of a positive, rational number greater than 1. But what if the silly user tries to give an incorrect input, like a negative number? You might want to give them an error message to let them know that their input was incorrect. Try using **conditional** statements to test for bad inputs, and use the **`error`** function to tell the user how silly they are. Type **`help error`** into the command window or read the MATLAB documentation (https://www.mathworks.com/help/matlab/ref/error.html) for details.

Question 3: Challenge!

In this section, you will implement your solution to question 4 from problem set 1 as MATLAB code. The problem is as follows:

> Suppose you are interested in the stress response of yeast cells to different concentrations of sodium. You want to know how the expression of 1000 different genes changes across 5 different sodium concentrations (4 experimental and 1 control). To do this, you perform single cell RNA-seq from 100 cells in each condition to get the expression levels of each of the 1000 different genes. Suppose you want to know for each condition which genes have expression levels 10x greater than in control conditions.

Note that my solution to this problem is posted in the answer key for problem set 1. You may use that algorithm as a basis for your function, or you may develop your own.

First, you can generate some fake data to analyze- this is a useful way of making sure that your code is working properly. Use the **rand** function to generate a 3D matrix of dimensions 1000x5x100, considering what those dimensions represent.

Then, write a function that will take this matrix of RNA-seq data as its input and return as its output the identities of the genes that have a 10x higher expression level compared to the control condition for each of the four experimental conditions. The genes can by identified by their index (#1-1000).

It is up to you to decide how you want to return this information to the user, but you may find it useful to have your function return multiple outputs. A function with multiple inputs and multiple can be defined as follows:

```
function [output1, output2] = functionName(input1, input2)
```

This function can then be run from the command line with the following syntax:

```
[a,b] = functionName(6, [2 3 1])
```

Save your function as pset3_q3_(your initials).