Week 2 Homework

- Due date: Tues, June 23 by 11:59pm EST. If you need extra time, just let me know.
- Complete exercises in a single script named "lastname_firstname_week##_hw.m." Functions should each go into their own .m file and be sent along with your HW script.
- You can call a function you wrote just like you call a built-in function: from the command line or from a script. For this HW, call functions from your HW script. NB: the function definitions and your script all need to be located in your working directory or else MATLAB won't be able to access them. There's a way around this, pathing, which we will get into at in a later lecture.
- **REMEMBER GOOD CODING PRACTICES**. **WRITE DOCUMENTATION FOR ALL YOUR FUNCTIONS.**
- Turn in the assignment by email at mlnguyen@princeton.edu

Problem 1: In class, we did an exercise writing a function, `plot_sagittal_slice(slice_ind)`, which plotted the sagittal brain slice specified by `slice_ind`. In this problem, you will write a more general plotting function that will plot a brain slice in any view: sagittal, coronal, or axial.

a. Write your function. Open a new file and name it `plot_brain_slice.m`. Write your function header with the function name `plot_brain_slice()`
   - The function should accept two arguments, `view` and `slice_ind`. "`view`" is a string that specifies the view of the slice, sagittal, coronal, or axial. "`slice_ind`" is a number specifying a slice in the brain.
   - Use a `switch` statement with "`view`" as your switch operator. For each `view` case, get the brain slice from `slice_ind` and visualize.
   - If "`view`" is not sagittal, coronal, or axial, it is not a valid case. Print a message saying so.
   - **DOCUMENT YOUR FUNCTION.** Review lecture slides for best practices.

b. Test your function by calling the following lines in your HW script.
   - `plot_brain_slice('sagittal', 71)  % display sagittal slice #71`
   - `plot_brain_slice('axial', 71)  % display axial slice #71`
   - `plot_brain_slice('coronal', 40)  % display coronal slice #40`
   - `plot_brain_slice('ax', 38)  % 'not a valid case' message`

Problem 2: In studies with human subjects, you will often rely on your subjects entering data in an online form or responding to prompts on a computer. Because subjects are fallible or sometimes don't read your instructions (or for online studies, are bots), you need to do some error or input checking to make sure they entered a valid response.

a. Write a function `check_age()` that prompts the subject to enter their age and checks if they are eligible to participate in the experiment.

- The function does not take any arguments, but returns an output variable called `eligible`. "`eligible`" is either 1 or 0 depending on whether the answer is valid and the subject is eligible (1 = good, 0 = bad).
- Use the built-in function `input()` to prompt the user to enter their age. Use the help function to figure out how to use `input()`.
- The function should first check that the subject entered a number (not a char or string). Use the `isnumeric()` function. If it is not a number, print a relevant message to the command window and set eligible to 0.
- Check that the number is a valid age: greater than 0 but less than 120. If it is not a valid age, print a relevant message to the command window and set eligible to 0.
- Check that the subject is an adult (at least 18 years old). If they are not, print a relevant message to the command window and set eligible to 0.
- Otherwise, the subject is eligible to participate in the experiment. Print a relevant message to the command window and set eligible to 1.
- **DOCUMENT YOUR FUNCTION.** Review lecture slides for best practices.

b. Test your function. Run `check_age()` and when prompted, enter:

```
i.   22          % eligible participant
ii.  'nineteen'  % not a number, be sure to enter with quotes
iii. 0           % not a valid age
iv.  'asdf'      % not a number
v.   18          % eligible participant
```

Problem 3: In this problem, you will analyze a subset of behavioral data from one my recent fMRI projects. The data consists of two groups of subjects: Intact and Scrambled. The subjects in the Intact group watched a lecture on a novel topic in the fMRI scanner and then took a multiple choice quiz assessing their learning. The Scrambled group watched a temporally scrambled version of the lecture which interfered with their learning. They then took the same multiple choice quiz. Here, you will load this data, threshold it based on their level of attention, and then compute summary statistics for each group.

a. Load the learning data: `load behav_data.mat`. Use `whos` to see the three new variables loaded to your workspace: `attention`, `intact_data`, and `scrambled_data`.
   - `attention` is an nSub x 2 matrix. Column 1 contains each subject's attention rating, from 1 = very inattentive to 5 = very attentive. Column 2 contains a 1 or 0 indicating each subject's experimental group, 1 = Intact and 0 = Scrambled. Thus row 1 contains the attention rating and experimental group for subj1 in columns 1 and 2 respectively, row 2 for subj2 and so on.
   - `intact_data` is an nSub x nQuestion matrix containing 1s and 0s for subjects in the Intact group. A 1 indicates the subject answered that question correctly, while a 0 means they answered incorrectly. Row1 contains the responses from subj1, row 2 from subj2, and so on. Thus a 1 in row 2, col 5 means that subj2

(row number) got question 5 (column number) correct, while a 0 in row 4, col10 means that subj4 (row number) got question 10 (column number) incorrect.

- `scrambled_data` is the same as above but for the Scrambled group.

How many subjects are in each group? How many questions are in the quiz?

b. Use logical indexing to split `attention` into two separate matrices: `attention_intact` contains the attention ratings for the Intact group and `attention_scram` for the Scrambled group.

c. Next write a function that uses attention ratings to threshold the subject data. Thresholding means removing subject data that doesn't meet a minimum standard, or threshold. In this case, we want to remove data from subjects who didn't pay attention during the task and had low attention ratings. We're not sure what threshold to set though, so we're going to write a function that lets us vary the threshold.

- Write a function called `threshold_data()` that returns the data and subject number (which is the row number) from subjects whose attention scores are above a certain threshold. For example, if the threshold is set to 2, this function should return only the data and subject number of subjects who have attention scores of at least 2.
- The function should accept 3 arguments: `scores`, `threshold_var`, and `threshold`. `scores` is an nSub x nQuestion matrix containing subject data. `threshold_var` is an nSub x 1 matrix containing the threshold variable (attention ratings). `Threshold` number between 0 and 5 (0 = no thresholding).
- The function should output 2 variables: `subj_num` and `data_thresh`. `subj_num` should contain the row number of subjects who passed the threshold and `data_thresh` should include the scores that passed the threshold.
- Test your function. Do the results look correct?
  i. `[subj_n, data] = threshold_data(scores, attention, 2) %use Intact group data`
  ii. `[subj_n, data] = threshold_data(scores, attention, 3) %use Scrambled data`
  iii. `[subj_n, data] = threshold_data(scores, attention, 0) %use either`
- **DOCUMENT YOUR FUNCTION.** Review lecture slides for best practices.

d. Run the `threshold_data()` function on both the Intact group and the Scrambled group with `threshold = 2`. Calculate the percent correct for each subject in each group. For each group, report the number of subjects who passed thresholding, the mean percent correct, the standard deviation, and the range (min/max). Use google and the MATLAB documentation to figure out how to calculate these descriptive statistics.

Class questions [optional but helpful for me]

1. How difficult was this assignment from 1-5 (1=very easy, 5=very difficult)?
2. Approximately how long did this assignment take?
3. How useful for learning was this assignment from 1-5 (1=not very useful, 5=very useful)?
4. How is the overall pace of the course from 1-5 (1=way too slow, 5=way too fast)?
5. How helpful/useful are in-class exercises (1=not very useful, 5=very useful)?
6. In general, how's the class going? What's working and what isn't? Do you like the in-class exercises? Would you prefer more programming together (I write out the code and explain what I'm doing as I go)? If you'd prefer to comment anonymously, feel free to email Ed Clayton (ec12@princeton.edu) instead and he'll summarize your comments for me.