Homework 4

Due: Thursday Oct 31, 2019, 9:55 am

To submit:

- Save your work as a .ipynb file. Use the filename: **LASTNAME_HW4.ipynb**
- Upload the file to the *Homework 4* Activity in Moodle

Collaboration:

You may work with other students, but each student should write their own code for the assignment separately. In a comment at the top of your homework please indicate the people with whom you worked on the assignment.

Assignment goals:

In this assignment you'll use random number generators to randomize the conditions of an experiment; continue analyzing the categorized free recall data we used in the Midterm and in class; and program a more complete version of a recognition memory experiment.

Problem 1: Randomization of experimental conditions (30 pts)

Psychologists often need to randomize the sequence of conditions in an experiment to avoid *order effects* in subjects' responses. Suppose you had an experiment with 3 blocks and you wanted 3 conditions within each block to be randomized in terms of presentation order. Suppose further that you have the requirement that your design have no consecutive repeats. The following design would be considered valid: `1 2 3 2 3 1 3 1 2` but this one would be invalid: `1 2 3 2 1 3 3 2 1`.

Write code that takes in two parameters (`nblocks` and `nconds`) and creates a randomized sequence of `nconds` across `nblocks`, subject to the constraints described above.

Problem 2: Analysis of similarity in category recall (35 pts)
We have already spent some time analyzing the data in `midterm_catfr_data.txt`, looking at how performance on categorized free recall is related to performance on standard free recall in participants that performed both tasks. In this problem, what you will do is determine the **pairwise relationships** in recall performance between categories.

For each unique category in the dataset, calculate the across-participant correlation between recall performance for that category and each other category. For example, for Trees you would calculate the correlation in recall performance between Trees and Landscapes, Trees and Zoo animals, Trees and Instruments...Repeat this process for each pair of categories.

Since there are 26 unique categories, you should store your correlation values in a 2-dimensional numpy array that has size 26 x 26. Each element of the array will contain the correlation between proportion recall for the corresponding pair of categories.

In this analysis, since we are **not** comparing to standard free recall, you should use all the data from all subjects that participated in the categorized free recall experiment.

Which pair of categories has the highest correlation in recall performance? Which pair has the lowest?

Problem 3: Program the encoding and test phases of a recognition memory experiment (35 pts)
Write code for an experiment that will present a list of words one at a time during the encoding phase of a recognition memory experiment, and will then present the same words during the test phase. To do this, read in the words from `wordpool.txt`. Set two variables at the start of the experiment (`ntargets` and `nlures`) that I can modify when I run your code to change how many targets are presented during encoding and how many targets + lures are presented during test. After all of the words are presented during encoding, your experiment should then present the user with a screen asking the user to press a button to continue on to the test phase of the experiment. The code should wait at this screen indefinitely until a key is pressed.

Your code should use `ntargets` and `nlures` to **randomly** sample from the set of words in `wordpool.txt` and use the random sample as stimuli in the experiment. Be sure that your random samples of targets and lures do not overlap!

Set a variable at the beginning of your code to define how long to present each stimulus during encoding (use 1 second as the default). During the test phase you should have the participant press either `j` (old) or `k` (new) to indicate their response. You should store the participant's response to each item in a `numpy` array.

Once the experiment is finished, your code should (1) close the experiment window; (2) calculate the hit rate and false alarm rate for the participant; and (3) display the hit and false alarm rates as output.