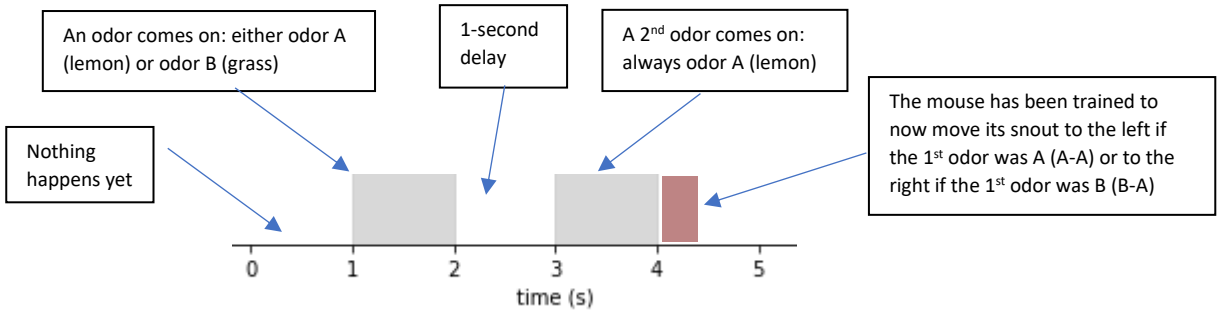
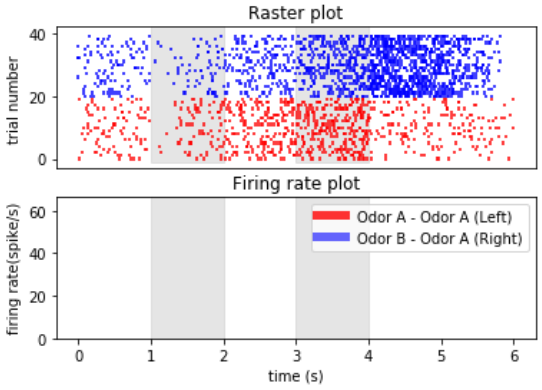
Coding & Neural Activity - April 4, 2019

In this exercise, we will manipulate neural activity data using Python. First, you need to install python 3.7 and Spyder (the interface) by installing Anaconda: https://www.anaconda.com/distribution/

Next, open Spyder and open the script called *neural_analysis.py.* This has starter code to analyze simulated data 'recorded' from the brain in mice performing a task that uses working memory:



To open the file: press the folder icon on the top right, navigate to where you saved the code, open the file explorer tab and click the file called *neural_analysis_start.py.* Press run (green triangle or F5), and then start looking through the comments, functions, and variables. 12 variables should be created – these can be examined in the 'variable explorer' window on the right side of Spyder. This script starts with a section where some useful functions (like numpy) are imported; then some parameters (like number of trials) can be set by the user. Then the data are generated and plotted. When this is run, a raster plot should be generated in the console, and look something like this:



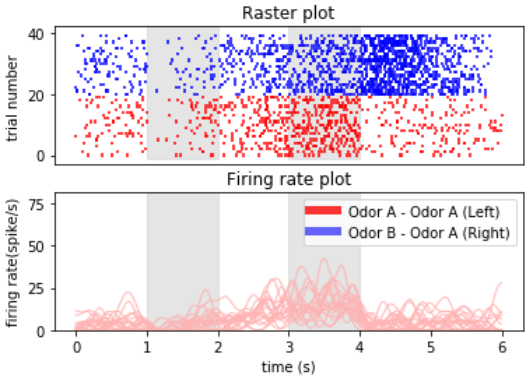
This plot shows the neural activity (action potentials, or 'spikes') in a single neuron, occurring during 40 trials of the task. Each point on the Raster plot represents one spike that occurred at that moment. This plot shows that this neuron was more active during the 2nd type of trial (Odor B – Odor A (move right)), especially from 4-5 seconds after the trial started. These tick marks are plotted using the variable called *spike_times*, which contains the times when the neuron spiked (e.g. 2.3 seconds, 3.4 seconds, etc.) on each trial.

Another useful way to display neural activity is to plot firing rates. This is just the average of the number of spikes per second in some time window – where there are a lot of dots close together, the firing rate will be high, and when there are no dots the firing rate will be zero. These values are found in the variable *firing_rates,* which is 40 rows (40 trials) x 6000 columns (6000 milliseconds).

Plotting can be done on line 64, using the imported *plot()* function:
plot(x_axis, y_axis, color = [R,G,B, transparency], linewidth = l)
In our case, you can use the following inputs (plot_colors_light[0] = [1, 0.7, 0.7, 0.9]. This will be helpful later on in the assignment) :
plot(time_axis, firing_rates[0,:], color = plot_colors_light[0], linewidth = 1)

Run the function again. You should see a line in the bottom plot showing the firing rate.

This shows the firing rate for a single trial. However, single trials are imperfect and noisy, so we would like to plot the firing rate for every trial. Use your knowledge of for loops and using indices to retrieve values from arrays, and now plot a line for all of the *number_of_trials* trials.

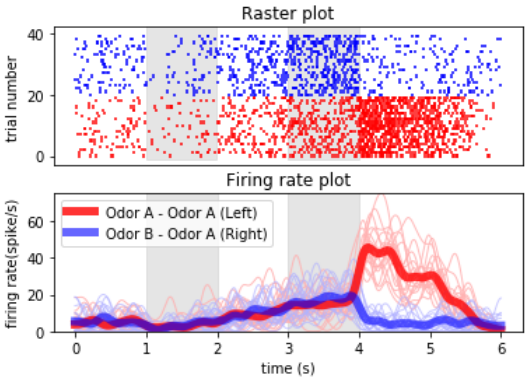


This only shows the firing rates from one of the trial types. Use another for loop to plot the firing rates for both trial types. Note that *plot_colors_light[1]* will give you the correct color for plotting the remaining trial type. Hint: in the case shown above, the first trial type occurs during trials 0 – 20, and the trial type occurs during trials 21 – 40; and the first axis of *firing_rates* is 40 elements long.

Plotting 40 lines is useful to some degree, but the average of all of those lines will be much easier to grasp. Use the np.mean() function to take the average firing rate for each trial type:
np.mean(array, axis = a) where a is the axis along which the mean is taken (e.g. if array is a 10 x 20 array of numbers, *np.mean(array, axis = 0)* would return a 1 x 20 array)
You will also need another for loop to get the average for both the red and blue trial types.

Once you get the average firing rates (an array of length 6000), you can plot them with:
plot(time_axis, average_firing_rate, color = plot_colors[trial_type], linewidth = 6)

You can then save this plot to a folder of your choice, with lines like:
file_location = 'C:\\Users\\philip\\neural analysis\\figures\\neural_acitivity.png'
savefig(file_location)



Great! This is basically how neuroscientists visualize neural activity for a single cell. However, computations are performed in the brain using many neurons. Set a new parameter in the beginning to set how many neurons you'd like to examine. Then create a for loop that can repeat the steps performed above for many cells. Save these figures as separate files and look through these different cells. How many types of responses do you find? What do you think this brain region might be doing in the task?

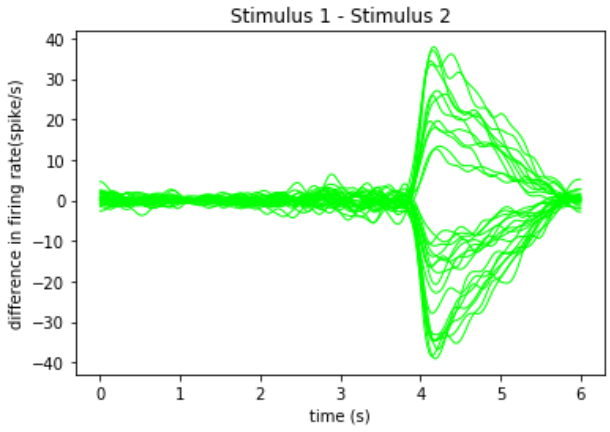
To test a hypothesis you might have, you can perform more analysis on the average firing rate data. One basic but important analysis we use is to compute the difference between the average firing rates of a single neuron during trial type 1 and trial type 2. This tells you how much that neuron is *tuned* to whatever is different between those trials. In this case, that could be odor A vs. odor B, or turning left vs. turning right.

First, create a new figure and subplot before the for loop that loops across neurons:
differences_figure = figure()
differences_subplot = subplot(111)

For each cell, subtract the average firing rate for one trial type from the average firing rate from the other trial type. This will give you an array of length 6000. Once this is done, plot this line on the figure. Since this code uses multiple figures, before invoking the plot() function, you must bring up the figure you want to use for this plot:
figure(differences_figure.number)
differences_subplot = subplot(differences_subplot)

You can also format this plot with a line like:
differences_subplot.set(title = 'Stimulus 1 - Stimulus 2', xlabel = 'time (s)', ylabel = 'difference in firing rate (spike/s)')

Finally, save this figure to the folder you've selected for saving figures.



What does this plot tell us about what this brain region seems to 'encode'? Think about what high vs. low values on this plot mean, and think about how the timing of high values compares to the timing of the task events shown on the 1st page of this document.

Let's look at another brain region now – change the *brain_region* parameter to 'brain region 2', and run this script again. (Make sure that when you are saving the figures from this brain region using *savefig()*, the file path is different from the one from brain region 1 – otherwise you will overwrite those files).

Examine the raster plots and firing rate plots, as well as the plot showing the difference in firing rates between the two trial types. How many types of responses do you find? What might these different neurons be doing in the task? What does the difference in firing rate plot tell you about what these neurons are ‘tuned’ to?

Finally, look more into the different neuron types in brain region 1 and 2. Look at your saved images of the neural activity plots and write down all the different activity patterns you see. Use if statements to separate the neurons into different groups depending on what kind of activity they have, give each group a name, and count how many neurons are in each group. If you have any other hypotheses about these neural data, how could you test them using python?