**Folder purpose:**

These scripts/functions are designed to format and do major analyses in a consistent manner across the lab. However, it will not analyze everything. For example, it may compute coherence between two signals, or multivariate granger with 3+, but it may not tell you where significant differences lie. You should have scripts in your data storage folder that do this so anybody can analyze them at a future date.

**How to use this folder:**

1. Determine if you want to bin by space or by time. If you want to look across the entire session, you will be binning by space. Alternatively, if you want to bin
2. Open up that folder and find the function you wish to use.
3. Run the function and when it is finished, right click and save your data into your own special folder. Please follow the follow guideline for naming your data: *data\_\** where the \* indicates the rest of the name that you desire. It is helpful to add lowercase and capital letters like so: *data\_PfcHpc\_grangerStem*  or *data\_coherenceTimeBins\_stem*. The same is true of figures. Please store any figures like so: *fig\_\**
4. Once your data is saved, you can plot it using the functions from this folder, but you will most likely want to plot it yourself.

**Folder details:**

There are broadly two separate approaches for the data set. You can bin by time or by space. Each has an advantage over the other one and may be employed based on your rational.

Binning by time allows you to control for the amount of time being considered. This is especially important for LFP analyses where differing amounts of time could influence your analyses since they will inherently have different cycles of the frequency of interest. For example, say you have a within subjects design where the rat experiences context A, then B. Then you realize that the rat spends more time on object A than B. You may want to examine the first second of their occupancy at the goal (or T-junction, or whatever else you are interested in). Thus, binning by time would allow you to assess differences between A and B in terms of LFP (say power or coherence, etc…) while controlling for differing amounts of cycle counts (of say theta). Additionally, binning by time makes LFP analyses extremely simple, and rarely will you have to subsample or interpolate data if you clean using the rmlines version of chronux cleaning functions.

Binning by space is equally, if not more strong, however. An example of binning by space is if you have a rat running on the T-maze on context A, then B, and you are interested in a specific location across contexts (see Hallock et al., 2016 linear classifier for a clear example of binning by space). This approach controls for the rats location in an environment.

Note: what do you do if you don’t have an Int file? You will default to the Binned by Events folder. This folder will contain everything that doesn’t do some kind of binning procedure. While the Binned by Events folder could technically fall under either binned by time or by space (depends how you defined your events), it was easier to create a new folder.

**What to do after you have your data?**

This folder was designed to analyze data in a consistent manner. However, most of the time, it will not plot things for you. Once you have your data, visualize it! Save a script in your data storage folder that calls it up and analyzes it for you!

**Current code in this folder:**

*ClassifierPermutate:* This code is the most well-rounded SVM classifier the lab has to date. However, it is currently only suited for a pseudo-simultaneous methodology (see Spellman et al., 2015). If you would like to include your entire session (including incorrect trials) and all sessions contain the same number of trials, you can do so. But what makes this code so powerful is three things: 1) You can have sessions with different number of trials or only extract correct trials because this code will run the classifier on 1000 different combinations of firing rates and control for different sized sessions. Additionally, this controls for any effect that you may have seen due to chance by the way your data was organized. For example, lets say you run the classifier on 10 sessions, each session contains 12 trials and you find that it predicts your two classes well. How do you know the classifier didn’t happen to do well due to the organization of your data? In other words, since trial one is always row one and trial 12 is always row 12, your effects could be spurious. So this shuffles that ordering. 2) You can take two separate approaches to classification, one being the leave 1 out approach, and the other being the train on 75% test on 25% approach. No matter what, it is recommended that you iteratively estimate this 1000 times (see code for how to do so). 3) Data will always be compared against a chance level distribution that is calculated by shuffling your labels and running the classifier 1000 times. This removes any arbitrary assumption that 50% is true chance.

*get\_sessionFR:* This will extract the firing rate across the entire session. Therefore, as long as you have a TimeStamps variable, you can run this.

*Instant\_speed*: A function that calculates instantaneous speed (