Shay Neufeld
Gil Mandelbaum
11/5/2016

Milestone #3
Data Exploration

When we first set out to do this project, we were excited at the prospect of training a model on neural data to predict the decision or behavior of an animal. We both have developed paradigms to study decision making in mice, and are in the process of collecting neural activity data from these mice during behavior. During our effort in exploring the data, we began to realize that before we try and model the neural data to predict behavior, it is crucial we have a deep understanding of how the mouse is actually behaving. Without a way to model the behavior of the mouse, our ability to model how the neural data relates to behavior is somewhat uninterpretable.

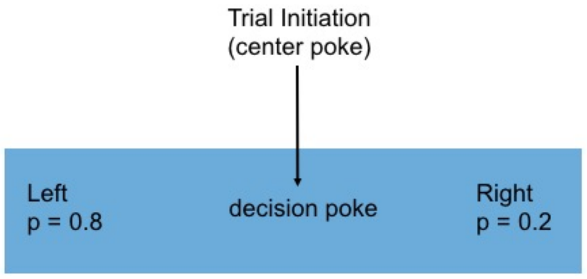
Therefore, we would like to pivot from modeling the neural data to modeling our behavioral data instead. The process we went through to arrive at such a conclusion was extremely productive. We also believe our behavioral task is particularly well suited to modeling with machine learning approaches – in fact, it touches on a topic that has been of great interest to machine learning for many decades: reinforcement learning!

The data we will model was collected during an experiment that we designed over the past few months. Briefly, mice are trained to do an adaption of a two-armed bandit task. In probability theory, the multi-armed bandit problem refers to the scenario where a gambler has to decide which slot machine to play, how many times to play each machine, and in which order to play them in an effort to maximize earnings. This has become a classic paradigm for studying reinforcement learning and the trade-off between exploring your environment and exploiting your previous knowledge of it. Considerable effort has been spent trying to implement machine learning algorithms to optimize the explore-exploit decision process within a world of probabilistic stimuli and rewards. Considerable effort has also been spent trying to understand how we and other animals behave in such situations. To date, only a couple studies have published results where mice perform an analogous behavioral task, however the attempts to actually model and predict the mouse's decision strategy have been very limited. We hope to use this project as an opportunity to improve our ability to model and interpret how a mouse strategizes to maximize reward in a world of noisy stimuli.

The behavior paradigm

Specifically, the mouse initiates a trial by poking a port in the middle of an arena, and then must choose to go to the left or right port. The left and right ports deliver water with varying probabilities. Therefore, the mouse must sample both ports, build an expectation for which port has a higher probability of reward, and bias their activity towards that port. The mouse must also learn to flexibly change their bias in the face of new evidence (i.e a change in probability of a reward). Here, the left and right ports are analogous to two slot machines with varying pay-out probabilities. Below is a picture of the three ports and a simple schema of the trial structure.

2-armed bandit behavior task for mice



The data

The behavior is run using an Arduino that communicates in real-time with Matlab. A custom behavior box was made with IR-beam detectors in each nose port, so that we can detect exactly when and where the mouse is poking. The water delivery is accomplished using solenoids controlled by the Arduino (via commands from Matlab).

The Matlab code used to run the behavior was a project Shay did over the summer, and the code can be found here: https://github.com/shayqn/TwoArmedBandit-3PortNosePoke-behaviorCode

Each behavior session runs ~30 minutes, during which time a trained mouse performs ~400 trials. With each behavior session we construct a Matlab structure that contains every poke detected, along with the corresponding data:

- Time poked
- Which port was poked
- The port probabilities
- Whether a reward was given.

Data processing

Since we want to model the decisions of the mice, we want to organize the data so that each 'observation' is a single 'decision' – that is, a trial where the mouse chose left, or a trial where the mouse chose right.

Not every 'poke' corresponds to a trial, however. Sometimes the mouse will incorrectly poke on the side before going to the center (we call these error pokes). Additionally, there will be pokes detected corresponding to when the mouse is happily licking up the water reward.

Therefore, we want to process the data in such a way where we find two pokes (a center poke and a left or right poke) that corresponds to the mouse correctly performing a trial. To do this, we wrote a Matlab script that searches through all the pokes, and based on the structure required by the task (i.e. The mouse must poke in the center and then go to one of the side ports) creates an array where each row corresponds to a single trial. Specifically, after processing the data in this way, each row contains the following columns.

1. The time (in seconds) since the beginning of the behavior session.
2. The time (in seconds) since the previous trial
3. The time (in seconds) between the initial poke and the decision poke
4. Which port was chosen (left or right)
5. The reward probabilities of the two ports at that time (two values between 0 and 1)
6. A Boolean representing whether a reward was given (1 or 0).

An example of the resulting csv file imported as a pandas dataframe is given below:
(note the column names provided here are to maximize clarity, but we used shorthand names to make coding easier)

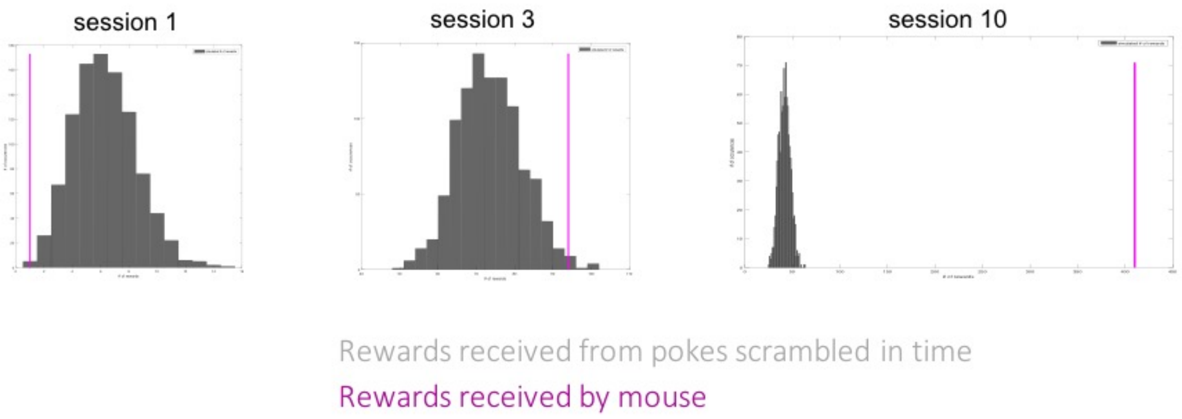
	decision poke time (s)	time since last trial(s)	time between trial initiation and decision(s)	port(1=R,2=L)	right port reward probability	left port reward probability	reward given yes=1,no=0
0	22.024	1.478	0.346	2	0.3	0.7	1
1	24.812	2.287	0.501	2	0.3	0.7	0

Data exploration

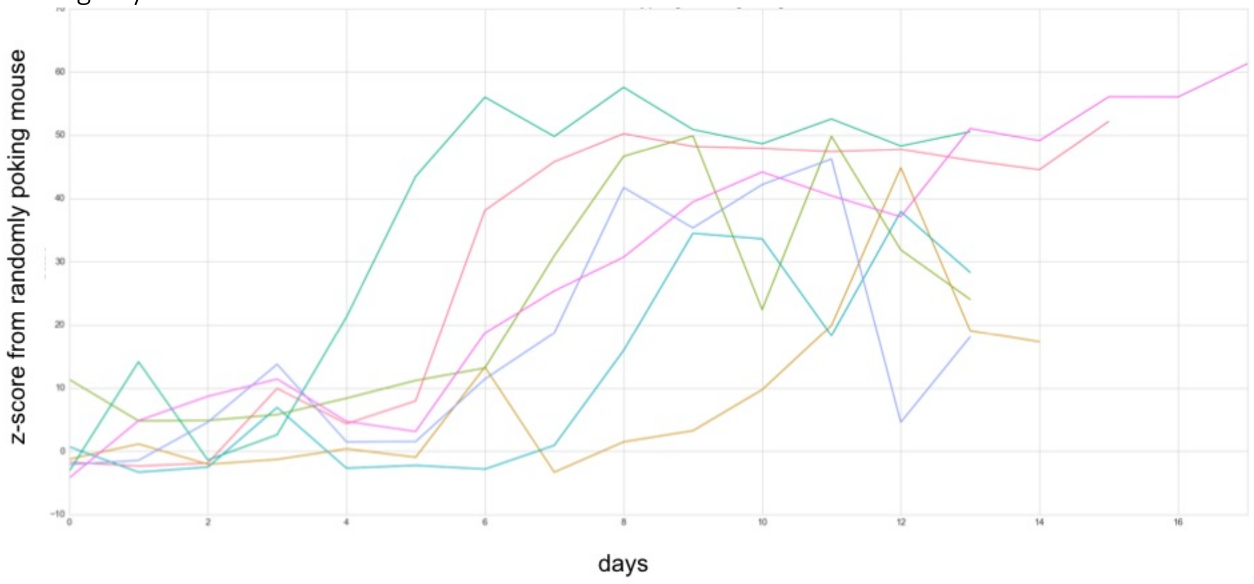
Is the mouse performing the behavior above chance?

The first thing we want to know is whether or not the mice are actually learning the behavior! It's possible that a mouse could collect some number of rewards by simply randomly poking the ports at a high rate. In fact, we can calculate exactly how many rewards a 'randomly poking mouse' would earn. To do this, we took the length of a behavior session (~30 pokes) and the total number of pokes the mouse did during that time, and then randomly distributed those pokes within those 30 minutes. We then imposed the structure of the behavior (you need to poke in the center, and then choose a side within a certain amount of time) and calculated how many rewards would have been awarded. We then repeated the randomization of pokes many times to get a distribution of how many rewards a randomly poking mouse would have got. Finally, we compared how many rewards the mouse actually received, and calculated a z-score from the distribution of rewards earned by the shuffled data.

Below are the histograms of rewards earned by the shuffled data, and in purple is the true number of rewards scored by a single mouse on that day. Apologies for the unreadable numbers on the axes here, but the trend and result is clear – the mouse starts off performing well below chance, but quickly improves and by session 10 is many standard deviations above chance.



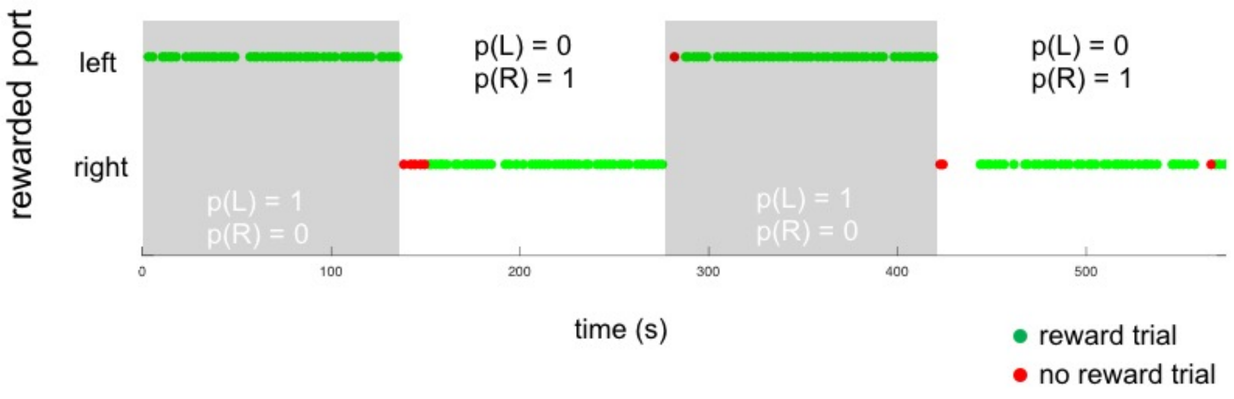
How does this learning look across mice? Below are the z-scores for animals starting with the first day of training. Each line corresponds to a single mouse, and each x point is a consecutive training day.



We can appreciate that mice start out non-significantly different than a random poker (indeed, sometimes much worse!), but quickly progress to several standard deviations higher, and finally attain z-scores > 10 indicating the mouse are receiving many more rewards than would be expected if it was simply randomly poking.

Example of well-trained mouse performing behavior

To get a sense of what this behavior looks like, we can plot each trial the mouse performs (left or right) and color code it according to whether it got a reward (green for yes, red for no). In this session, the reward probabilities of the two ports switched between 0 and 1 – with these extreme probabilities, the task actually becomes deterministic (i.e. there is absolute certainty that an unrewarded trial indicates a change in the reward probabilities of the ports).

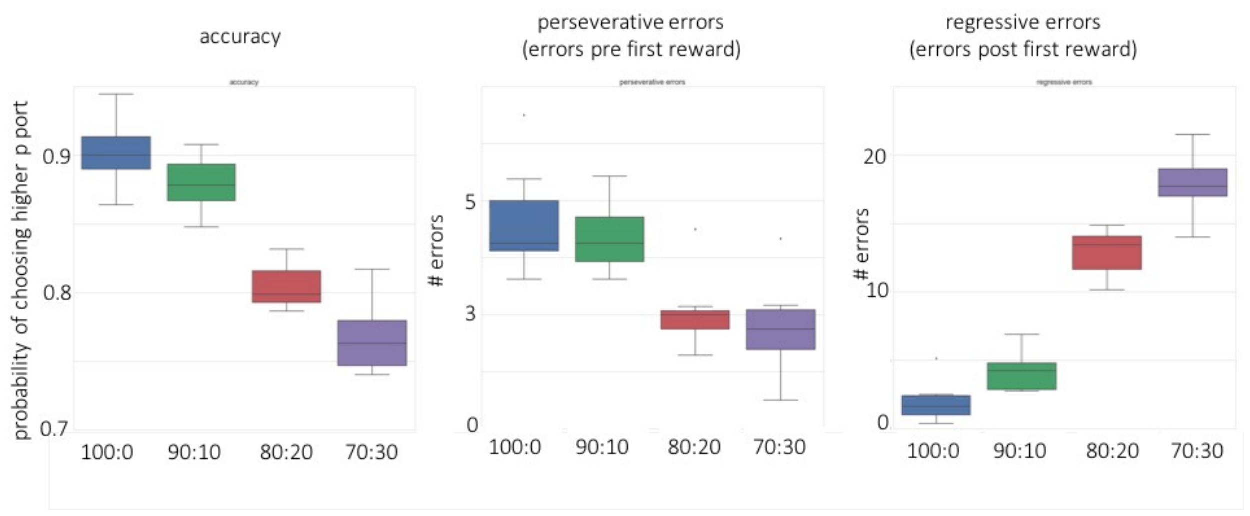


Given these conditions, the mouse is performing with extremely high accuracy, and flexibly switches between the two ports almost exclusively after it receives no reward.

Summary performance statistics averaged across mice and days for multiple reward probability conditions.

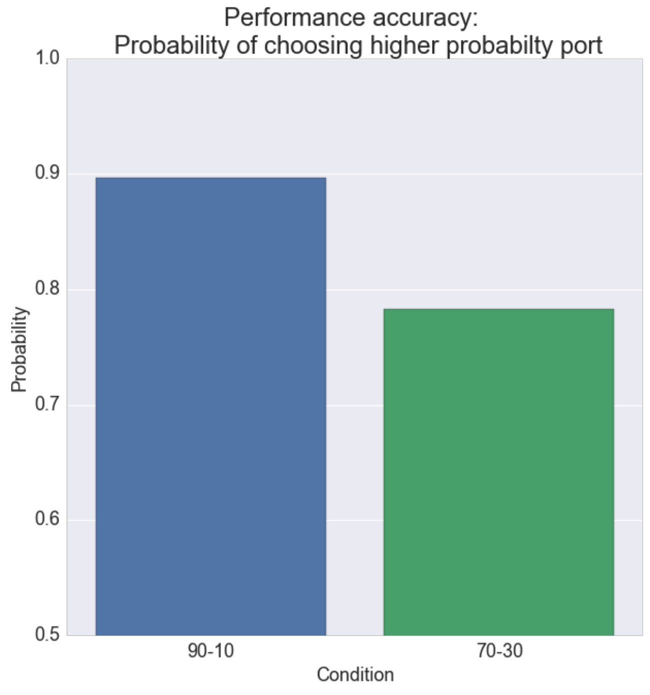
What happens when we change the relative reward probabilities from being deterministic (1 vs 0) to probabilistic (0.9 vs 0.1, 0.8 vs 0.2, or 0.7 vs 0.3)?

To get a sense of how mice change their behavior in response to different reward probabilities, let’s first define some metrics for measuring their performance. One metric is the fraction of times the higher probability port is chosen – we’ll simply call this ‘accuracy’. Another way to assess performance is by looking at the ‘error’ trials, where the mouse chose the low probability port. We can divide these error trials into 2 groups: the errors that occur before the first correct trial (after the reward probabilities change), and the errors that occur after the first correct trial. The first type of error we refer to as ‘perseverative’ errors, since it reflects the mouse perseverating on a behavior that previously resulted in a reward but no longer does. The second type of error is called ‘regressive’ errors since it reflects the mouse regressing back to a previous strategy that resulted in a reward but no longer does. The following data is averaged over 8 mice and over 3 different behavior sessions per condition. Error bars are SEM.



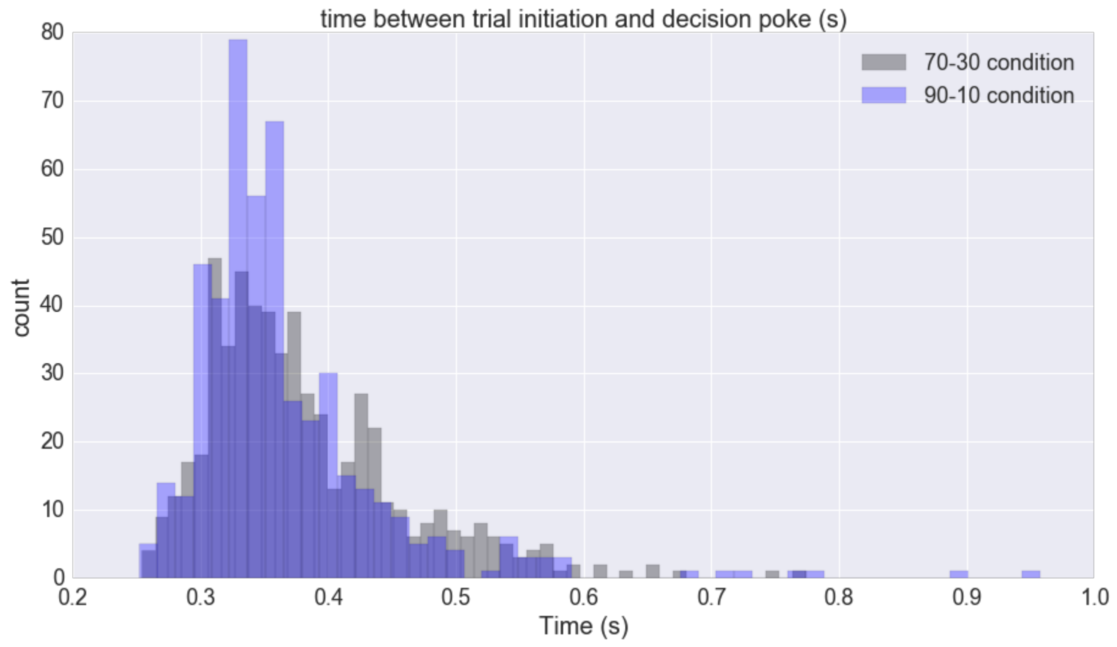
Exploring the data from a single mouse under different reward probabilities:

We want to model the decisions of a mouse on a trial by trial basis – that is, perhaps given some history of behavior, what will the mouse do next? In order to explore how our data might be amenable to such a model, we first explored two behavior sessions from the same mouse which contained different reward contingencies. One day, the mouse had to switch between reward probabilities of 0.9 and 0.1, and the other day between 0.7 and 0.3. Clearly maximizing reward optimally becomes more difficult as the difference in reward probabilities becomes smaller. First, let's look at the performance accuracy of this mouse across the two different conditions:



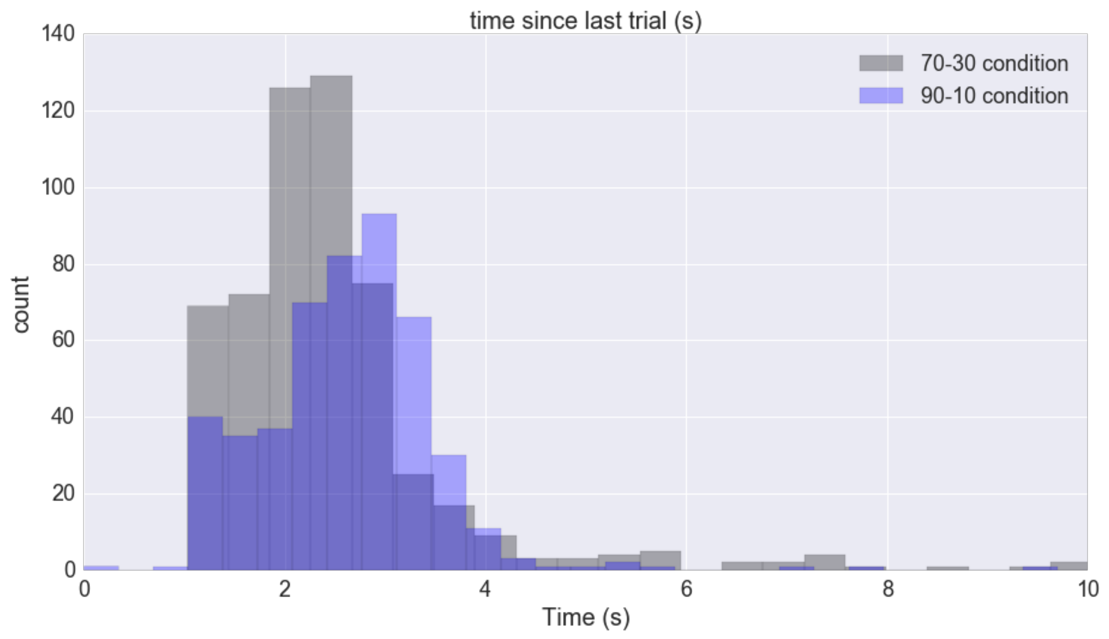
For the 90-10 condition, the mouse is basically performing at the reward probability – 90% of the time it chooses the 90% reward probability port. However, for the 70-30 condition, even though the mouse is overall less accurate, it is actually biasing it's behavior more in the sense that it is well above 70%. This nicely follows from the average data across many mice and days shown above.

One metric we can take a look at is the time the mouse takes between the center-port initiation poke and the decision poke. During this time, the mouse must move ~5cm to the left or the right and stick it's nose in the port to see if it gets water. Below is the distribution of those times for the two different conditions:

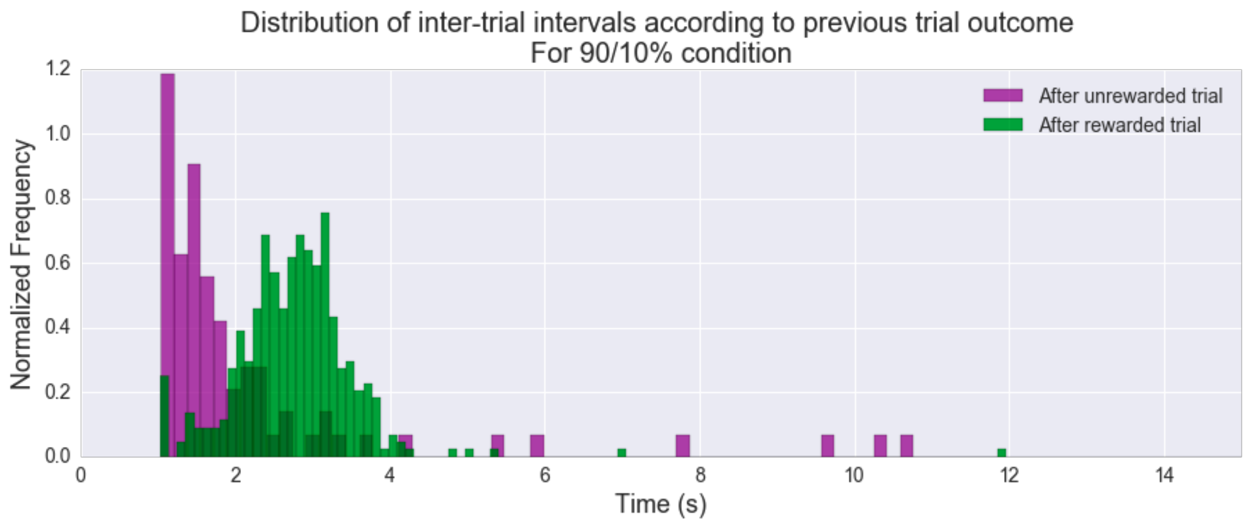


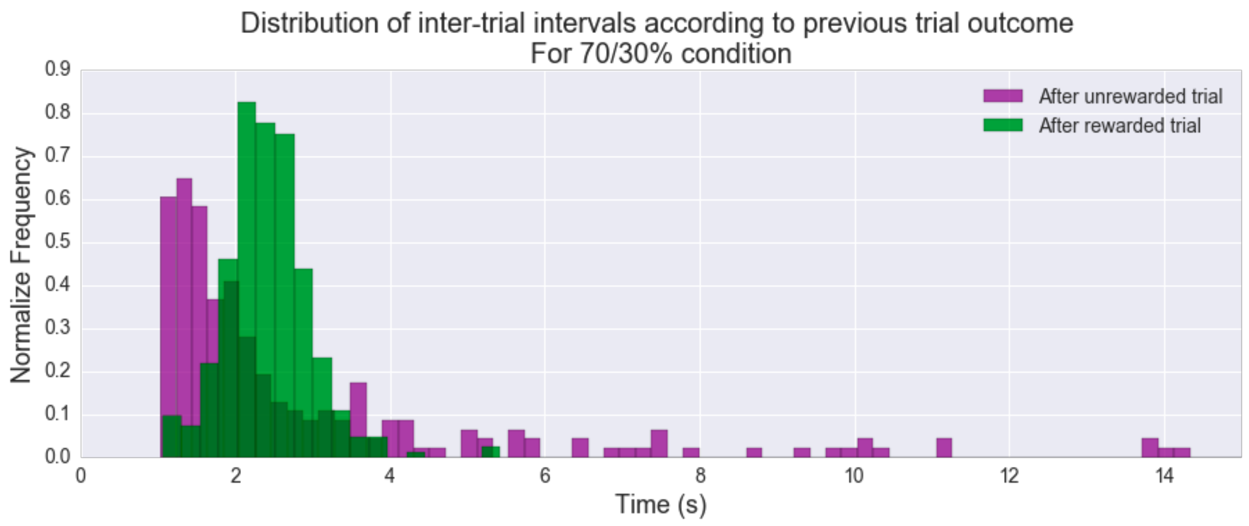
The two distributions look pretty similar – it seems as though changing the reward probabilities does not substantially change the time it for the mouse to execute a trial.

A different time variable that we intuitively guess could reflect the behavior of the mouse is the inter-trial-interval. In this behavioral paradigm, the mouse self-initiates a trial – it gets to decide when to poke in the center. Therefore, differences in how long the mouse takes between trials could reflect differences in decision making or motor planning. To start exploring how these data look, we plotted several distributions of inter-trial-interval below:



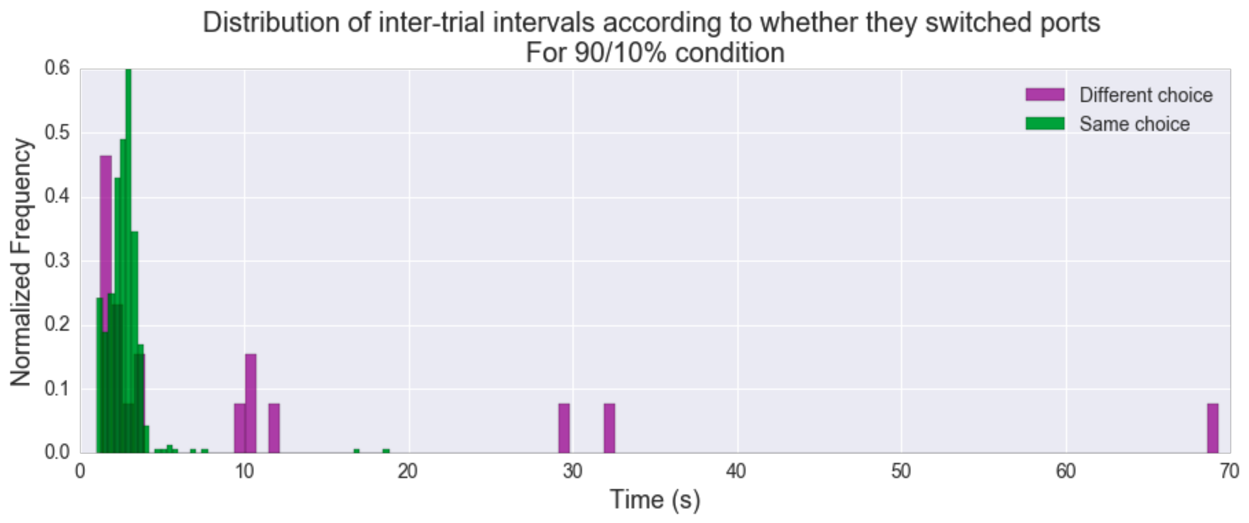
Here it looks like a difference might be emerging, where the inter-trial-interval is longer for the 90-10 condition than for the 70-30. Why might this be? Well, we know there are more unrewarded trials in the 70-30 condition – perhaps the inter-trial interval changes depending on whether the mouse got a reward on its previous trial. See below:

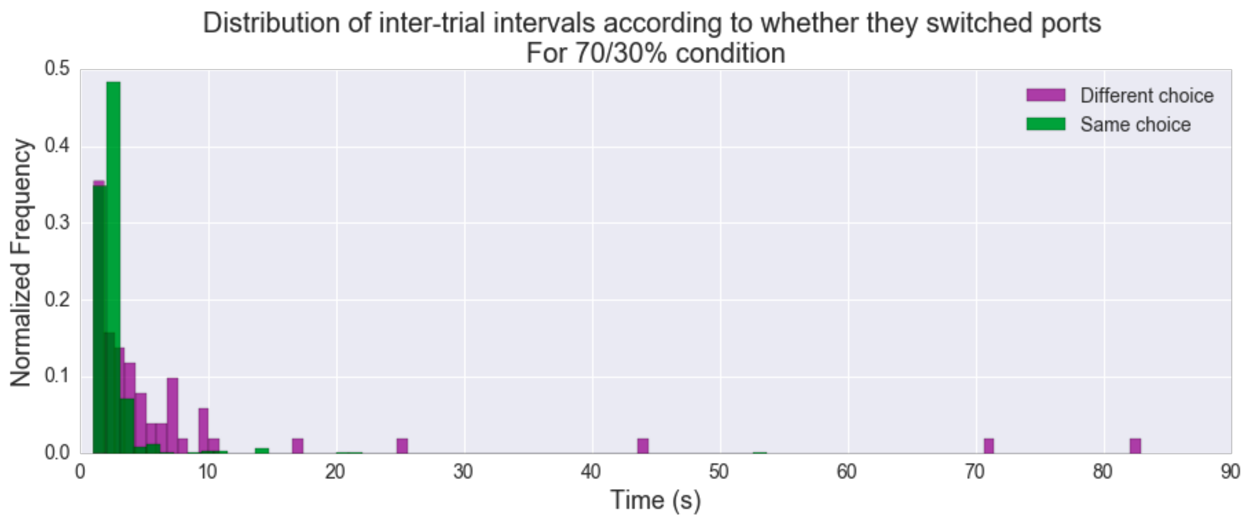




Here there is definitely a difference in the distributions! After an unrewarded trial, the mouse is much more likely to quickly initiate the next one. This could be the reason for the overall difference between the 90-10 and 70-30 inter-trial-interval distributions.

Another thing that might affect inter-trial-interval is whether the mouse changes its decision from the last trial. Perhaps the act of deciding to switch your decision requires more time to 'think' – certainly a hand-wavey idea, but we often attribute deciding between choices as taking time! See below:





There isn't as marked a difference as when we compared reward/unreward, but the different choices do certainly have a longer tail in the distribution. Do those extra long waits correspond to times where the mouse got several unrewarded trials in a row? That's something we can delve into more deeply as we continue to analyze the data.

There are lots more things we can do to explore this data more – we can explore the timing more to see if particular past or future conditions are indicative of how much time the mouse takes before initiating. We can also take a look at mouse to mouse variability. As evidenced in the early 'z-score' plots, we also have data while the mouse is mastering the task, which we could try and interpret more deeply. And, as described below, we also have a more complex probability conditions that we can analyze.

One things we are particularly interested in doing is doing a logistic regression using the mouse's previous decisions to see whether the behavior history can be used to accurately predict future choices.

Available data on hand

We currently have data for 8 mice performing the following conditions:
100-0
70-30
80-20
80-50
90-10
For each condition we have >= 3 sessions per mouse.

We also have data where the probabilities do not sum to 1, and do not reliably switch back and forth. For example, in one training example, the ports might switch between the following pairs of probabilities:
80-40
20-90
50-50
70-10
60-40
Together this data contains ~100,000 trials.

We now have preprocessed the data into the csv files described above for all the datasets mentioned here.