Ethan Sweeting
Sean Bevacqua
NE340: MazeRunner Lab
Professor Bullock

1. **Scientific Background:**

Since the earlier parts of the 20th century scientists have developed and implemented a number of mouse maze experiments to study learning, navigation, motivation, and more. These mental processes involve the acquisition and spatial localization of relevant visual cues. These cues are then are then processed, consolidated, retained, and then retrieved in order to successfully navigate an environment (Terry, 1970). In particular, scientists typically examine reward processing and evaluation in mice during these maze experiments. Fortunately due to advancements in computational neuroscience, researchers can use programs like MatLab to run thousands of maze trials concurrently in order to extrapolate meaningful reward system data.

Reward prediction error signals are driven by midbrain Dopamine neurons. In fact, previous research indicates that dopaminergic RPE's are actually also shaped by uncertainty (Starkweather et al, 2017). Therefore, mice placed in a maze will develop cognitive maps based on the RPE's which guide their learning and spatial location. This process is based on ancient systems of expectation that evolved over time which enabled mice to construct mental maps of their surroundings. These maps often relied on important cues such as food nesting location.

Our primary focus for our experiments was expected value calculations and their effect on learning and behavior in mice. According to previous research, mouse EV calculations are based on dopamine cell responses to reward-predictive cues that directly reflect the EVs of those cues. These EV's are an objective measurement, and they consider the value of an

option by multiplying the value of an option's outcome by the probability that that outcome will occur (Bullock, Lecture 5). Simulation models do have limits though, and their applicability varies from species to species. Human decision making for example "often departs from what we would expect if the objective EVs alone determined decision-making" (Bullock, Lecture 5).

For our experiments we utilized MatLab script "MazeRunner_V3" and "MazeRunner_V4" to analyze a number of parameters regarding reward cue prediction and their effects on a mouse's ability to navigate a maze. The script calls for a 15x15 matrix to be used as the maze. Within that matrix the numbers 0, 1, 3, 5, and 7 represent the path, walls, dead ends, end (5) and start (7), respectively. In conjunction with the maze there is an expectation map with values 0, -1, and 1 representing no expectation, negative expectation at a dead end, and positive expectation at the goal location, respectively.

The virtual mouse is put through the maze for a set number of trials. The goal is to assess learning as a result of expectation cues in the matrix. The mouse is assumed to be very deliberative, and will chose the adjacent square with the higher expected value (Bullock, Lab 5). Over the course of the trials the decision process can change if there is a learning process that changes the EV's of "distinct squares in accord with whether those squares are on paths that lead to reward" (Bullock, Lab 5). Over time these squares will serve as a cue for a reward further along that path. When faced with an EV choice, midbrain Dopamine bursts occur which is known as adaptive coding of reward value. In fact, previous research reveals that "the size of such DA bursts scales with the expected value of the predicted outcome, that is, with the product of the conditional probability and the value of the outcome" (Bullock, Lecture 5). MatLab enables us to examine the number of steps needed to complete the maze across trials. From this data we can determine, among other things, how well the mice learned the maze.

2. MATLAB implementation (~ 3 pages).

The first MATLAB code, MazeRunner v3, begins by allowing a binary preference of whether trial plots are skipped or not. If you are running hundreds of trials, skipping the maze plot might be the better choice, in order to save time, but if you are more interested in examining the learning of the mouse throughout the maze, running a smaller number of trials and plotting each movement may be more beneficial. Another option for ‘fast’ or ‘slow’ speed is presented, for similar reasons as the previous option. After these are initialized, an array is creating to simulate a maze that the mouse would run through. Zeroes denote the path, ones denote walls and threes indicate dead ends. Expected values are then assigned to different areas of the map to help mimic the learning taking place within the mouse brain. Dead ends have an expected value of -1, but the goal location has an expected value of +1. All other areas of the path have no expected value (0). The idea is that expected values increase as a positive reward is given (finishing maze), but EVs are weakened by reaching dead ends. This strengthening of neural maps should eventually result in a much more defined trajectory from the beginning to the end of the maze.

The first input that is prompted to the user is the number of trials. Each trial is defined as the entire route taken by the mouse in order to reach the goal location. As the trials begin, two initial graphs are displayed: an original map of the maze and an original map of the expected values of certain locations. An array is created to keep track of the number of steps taken to reach the goal per trial. A for loop is used to track the current and previous location of the mouse that also tallies the number of steps taken. A while loop nested within the for loop helps to update expected values as each trial is run. This loop recursively repeats as each step is taken, and has a real time map of current location and expected values. Once all of the trials

have taken place two more figures are generated: a final maze, showing the routes that have decreased and increased expected values as well as an expectation map with numerical values is displayed. In addition to these figures is a line graph, that shows the step counts per trial. Generally, the earlier trials take significantly longer than subsequent trials as the updated EV map is strengthened over time. This figure is one of the simplest ways to visually demonstrate the phenomena taking place in this experiment.

There are many user-defined functions called during while-loop of the script. The first of these generates a list of possible moves that the mouse can move based on its current and previous locations. There are four possible directions that the mouse can move. Although this function helps to determine possible moves, it does not take into account expected values. Another function called expectedValues helps to accomplish this. expectedValues is essentially an array of whatever the current expected map is. As each point is passed, its EV is added to a list and changes its values on the expectation map. The function for updating the EV map assigns -1 to the current square if all surrounding EVs are -1. If the expected value is positive, then the EV is multiplied by .95. A function newLoc, then uses this updated expectation map to make movement decisions. The main rules that this function follows is that if a space has a positive EV, then move that way, and if all surrounding spaces have negative EV, then go backwards. If all moves have a neutral EV, then a random move is chosen. Lastly, a function called plotMaze graphically displays a current output of the maze. The mouse's location is symbolized by a green circle and positive and negative EV's are represented by white and red dots, respectively. The maze shows a real-time trajectory of the mouse's path as it navigates through each trial.

At the end of the trials, MATLAB outputs an array of the final EV's. In longer experiments, there is more time for learning, so many of the EV's are either 1, 0 or -1, or some

number approaching one of these numbers. There are few squares that have values close to .5 or -.5. This makes sense, because enough time has elapsed to allow for a point of the maze to have a pretty certain EV after passing that point on multiple previous trials. On shorter trials, these EV's may be further from integer numbers, due to the fact that the space was not traveled by enough to have a strengthened value on the expectation map.

The code for Mazerunner v4 is very similar to the code of Mazerunner v3, except for one key difference. The code for Mazerunner v4 uses incremental weight passback in order to strengthen the expectation map. This minor difference uses a different equation to calculate EV's, which ends up having a significant effect on the mouse's ability to navigate the path.

3. Results of computer experiments performed with the code MazeRunner_V3 (100 Trials):

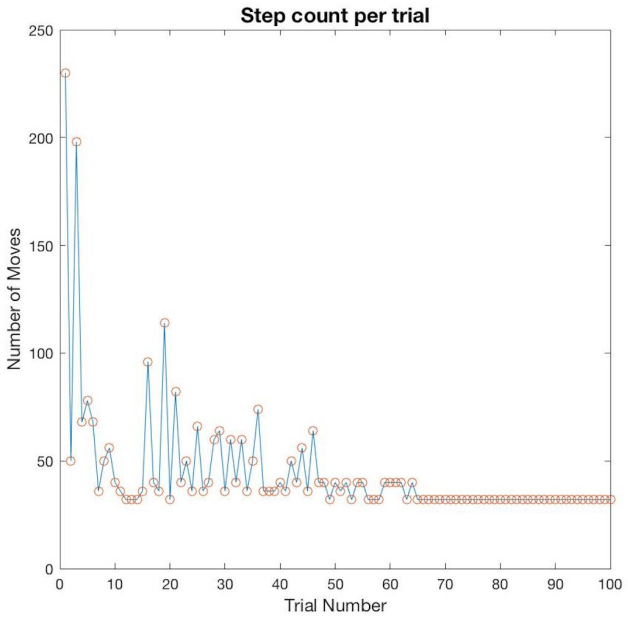

Figure 1: For our first experiment we ran MatLab script "MazeRunner_V3.m". The only parameter we inputted/changed was the trial count which we set to 100. Not surprisingly, the highest number of moves by the mouse was in trial 1. Over the course of the trials the step count decreased as the mouse "learned" to recognize that higher EV's indicated that the reward was along said pathway.

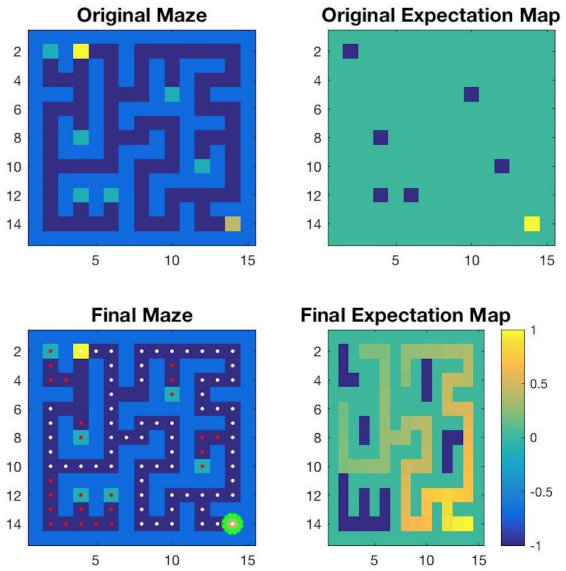

Figure 2: These 4 maze plots show the original maze, the original expectation map, the final maze, and the final expectation map. As expected the closer the mouse got to the end point (reward), the closer the EV was to 1.0.

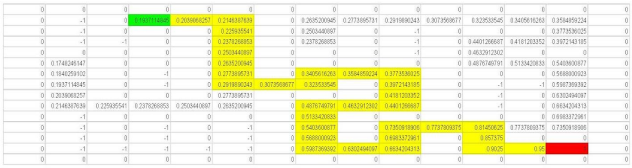

Figure 3: This figure shows the learned path. The green indicates the starting point and the red indicates the ending point. The highlighted boxes indicate the learned pathway.

MazeRunner_V4: 100 Trials

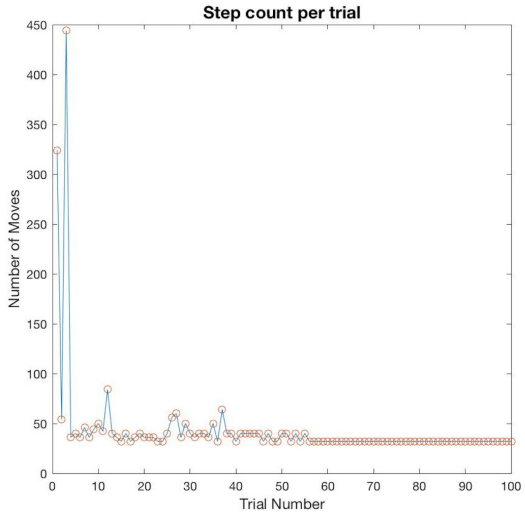

Figure 4: For this experiment we ran “MazeRunner_V4.m” for 100 trials. As expected, the highest step counts all occured before the 10th trial. The single highest step count was 445, the lowest was <50. This indicates learning occurred over time.

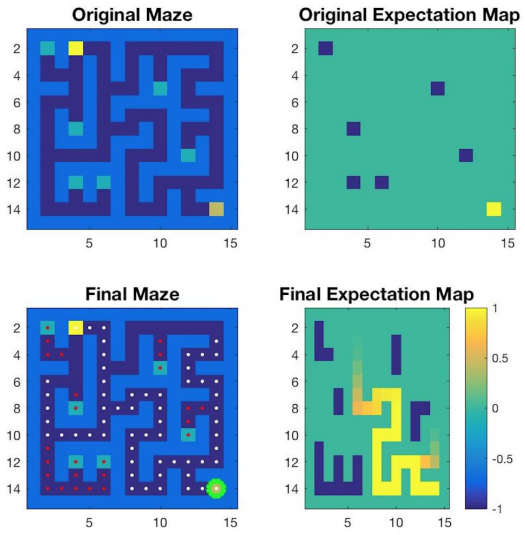

Figure 5: These 4 maze plots show the original maze, the original expectation map, the final maze, and the final expectation map. As expected the closer the mouse got to the end point (reward), the closer the EV was to 1.0.

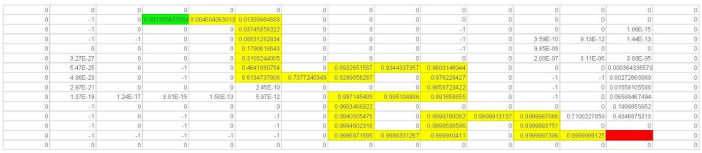

Figure 6: This figure shows the learned path. The green indicates the starting point and the red indicates the ending point. The highlighted boxes indicate the learned pathway.

MazeRunner_V3: 100 Trials + All EV's==0

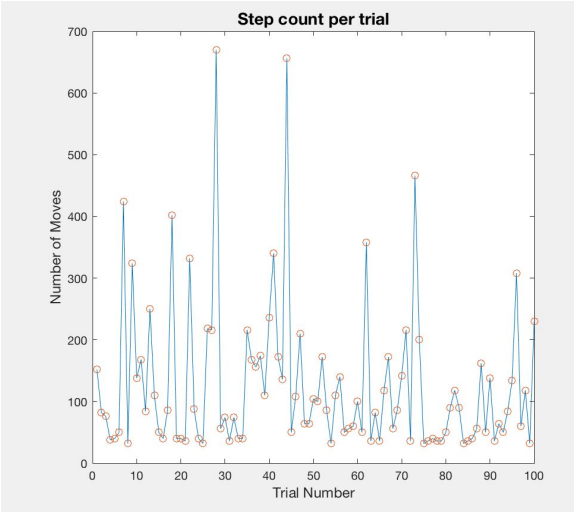

Figure 7: For this experiment we ran "MazeRunner_V3.m" for 100 trials with all squares set to an EV of 0. As expected, the step count did not decrease over time. Rather, it would fluctuate randomly. This indicates learning did not occur over time.

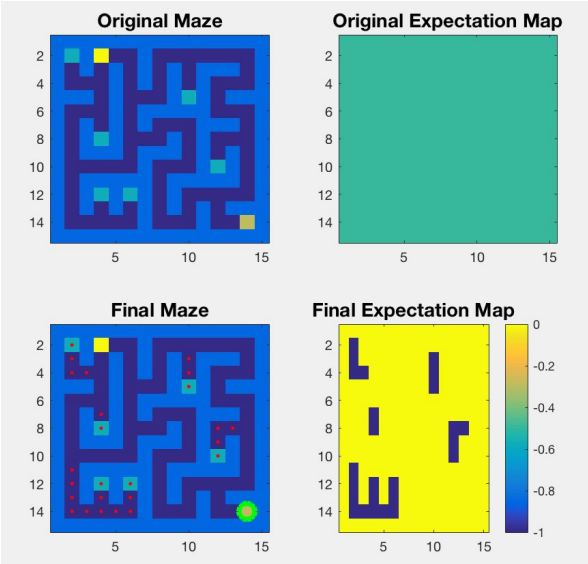

Figure 8: These 4 maze plots show the original maze, the original expectation map, the final maze, and the final expectation map. As expected, without EV's to guide the mouse, learning did not occur and the simulation was unable to construct a proper map of the environment.

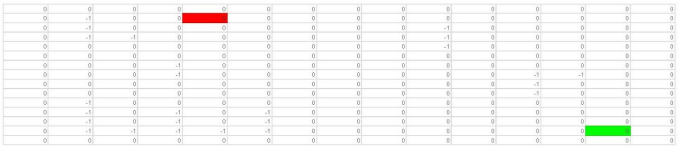

Figure 9: This figure shows the learned path. The red indicates the starting point and the green indicates the ending point. The highlighted boxes indicate the learned pathway, but without increasing EV's the mouse did not learn one path.

MazeRunner_V4: 100 Trials + All EV's==0

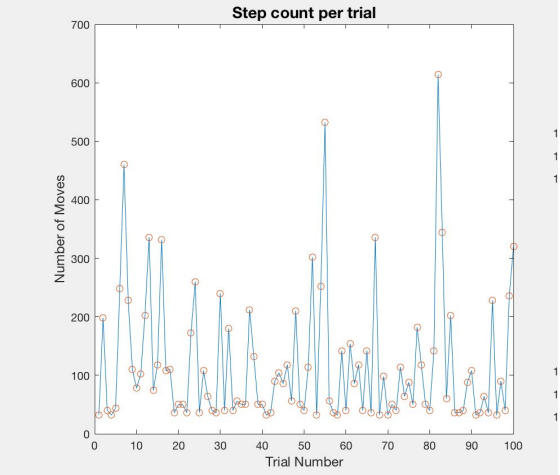

Figure 10: For this experiment we ran "MazeRunner_V4.m" for 100 trials with all squares set to an EV of 0. As expected, the step count did not decrease over time. Rather, it would fluctuate randomly. This indicates learning did not occur over time.

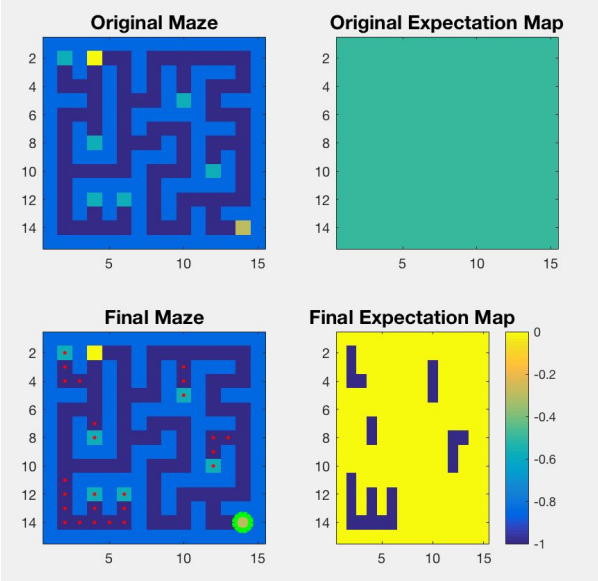

Figure 11: These 4 maze plots show the original maze, the original expectation map, the final maze, and the final expectation map. As expected, without EV's to guide the mouse, learning did not occur and the simulation was unable to construct a proper map of the environment.

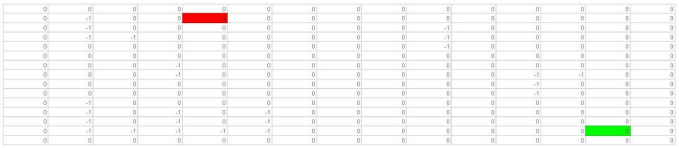

Figure 12: This figure shows the learned path. The red indicates the starting point and the green indicates the ending point. The highlighted boxes indicate the learned pathway, but without increasing EV's the mouse did not learn one path.

Summary of Results:

We ran four experiments for our project. The first two were MazeRunner_v3 and MazeRunner_v4 both for 100 trials, respectively. The second two were MazeRunner_v3 and MazeRunner_v4, both for 100 trials, only this time we set all the expected values in the EV map to zero.

For MazeRunner_v3 and MazeRunner_v4 at 100 trials each, both simulations produced the expected results. The highest number of steps during any trial occurred within the first 5 trials (235 steps for v3 and 445 steps for v4). For MazeRunner_v3, by around the 65th trial the step count remained constant. This indicates learning occurred and the mouse knew which path

to take for reward. For MazeRunner_v4 the step count reached a constant by around the 55th trial. This demonstrates even faster learning occurred during this simulation, due to the incremental weight passback function included in v4.

Once expected values were set to zero the mice had difficulty learning the correct pathway. Without a expectation map to guide them to the endpoint, mice in both MazeRunner_v3 and MazeRunner_v4 were unable to assess the increasing EV's and learn the best path. Step counts would fluctuate randomly over the course of 100 trials, indicating learning did not occur. These results demonstrate the importance of expected value calculations in everything from learning a maze to our basic spatial memory.

4. Questions, observations, interpretations:

The figures output from this experiment are useful, because they assess the data accurately across time and give a simple visual output of what is happening throughout the simulation. This simulation is definitely representative of how EV's allow for a map of navigable space to be constructed and the decisions that are affected. Two figures that provide some of the strongest evidence for conclusions presented in this paper are the step count graph and the final expectation map.

First of all, the step count graph almost always has a predictable outcome that shows clear learning over time. The first trials always take much longer and, at around 65 trials for MazeRunner v3 and at about 55 trials for MazeRunner v4, the mouse's performance reaches the end of the learning curve, and nearly has the entirely trajectory mapped out. This constant decrease of step count is very reliable evidence that learning is taking place in order to shorten the mouse's route. Secondly, comparing the final expectation map to the original expectation map shows a near direct trajectory carved out of EV's that was not present at all in the inceptive runs. Each updated expectation map slightly alters the next map as it chips away a route completely by numerical calculations. This same concept can be applied to humans as aversive stimuli, like a stench, may make us avoid being in a certain room, but a positive reward could make you to stay in one room much longer than others.

In a real life version of this experiment, hippocampal place cells would allow for the mouse to store a constantly updated map of the maze. Because we know Hebbian learning and synaptic modification to be present in all mammals, the EV model of maze learning can be concluded to be an accurate and reliable representation of how mice learn mazes and how to navigate their environment.

5. Discussion of the degree of correspondence between the processes included in the simulation code and actual brain processes:

Computational neuroscience offers scientists a means of quantifying and modeling neural networks as well as their corresponding behavioral responses. There are limits in the scope of many simulations though, and while exciting, these capabilities must be viewed with a cautionary if not skeptic attitude. The maze runner MatLab scripts we used for our experiments was able to calculate the exact number of steps taken each trial as well as compare the paths

taken to the corresponding expected values. The power of computational programs like MatLab enable scientists to gather and analyze large quantities of data. This data is encouraging but the complexity of neural networks far surpasses any script's contents. We know that expected value calculations arise from midbrain dopaminergic projections to the striatum (Starkweather et al, 2017). These projections vary in strength and the MatLab scripts assume the mouse is very deliberative and will chose the adjacent square with the highest EV. In a real mouse however their path choice is the result of numerous projections aside from midbrain DA. In fact, by itself "the DA signal carries little or no information about the specific sensory qualities of a reward or expected reward" (Bullock, Lecture 5). Mice may have intrinsic fears or behaviors that interfere with or modulate a computational model based on very specific parameters.

Despite these complications computational neuroscience still serves a valuable pedagogical purpose. With MatLab, examining and utilizing code to test neural network models allows students to better understand the implications of modulation (i.e an increased signal strength) on behavioral responses, among other things. In regards to our experiments, the ability to run thousands of trials in a short period of time helps students learn the effects of learning in conjunction with expected value calculations on spatial location maps. Additionally, the graphical capabilities of programs like MatLab provide a method of organizing and visualizing data.

The completeness and applicability of a model also may vary species to species. Human subject experiments may require a completely different model than one applied to a mouse subject experiment. Humans are capable of much higher orders of thinking whereas mice rely more on a basic set of values such as food or water. Therefore, the expected value calculation methods may differ between the two. According to previous research, multiple "MRI studies indicate that expected value [in humans] is represented as a 'common currency' network

encompassing the medial part of the OFC/ventromedial prefrontal cortex (vmPFC) and ventral striatum" (Li et al, 2016). This supports the idea that "that human decision making often departs from what we would expect if the objective EVs alone determined decision-making" (Bullock, Lecture 5). Human expected value calculations rely on many subjective factors, such as loss aversion, that also influence decision-making. On the other hand, our maze runner simulations assumes that mouse EV calculations are based on dopamine cell responses to reward-predictive cues that directly reflect the EVs of those cues (Bullock, Lecture 5).

All in all the degree of correspondence between simulation codes and actual brain processes varies from model to model. Their applicability also depends on the species being tested. While current models do lack some completeness, the ability to visualize and quantify data with simulation codes like the ones used in MatLab offer a great pedagogical opportunity. As the computational neuroscience field grows and new discoveries are made there is no doubt that simulations will become more and more accurate and applicable.

6. Discussion of some foraging-relevant brain processes that are known to exist but for which there is no corresponding process in the simulation code.

Foraging-relevant brain processes remains a topic of interest for neuroscientists. Computational neuroscientists construct models of decision making in a foraging setting using simulation code. Unfortunately these models are limited in their scope when applied to a complex mammal (i.e a human). Previous studies indicate that the threshold for decision making is a "background variable that represents the marginal intake rate associated with the overall environment. In economic terms, it is the opportunity cost of pursuit" (Hayden et al, 2014). This presents some problems with constructing simulation codes for foraging: (1) they are modeled

after problems encountered by animals; (2) decisions are usually framed as a foreground (pursue) vs. background (ignore), rather than as two simultaneously presented alternatives as in standard economic tasks; (3) human subjects exhibit loss aversion which is difficult to model with MatLab or a similar simulation program. These issues often prevent simulation codes from achieving full applicability. Even more interesting, in foraging tasks, “animals discount only weakly or not at all (Hayden et al, 2014)”. Our MazeRunner simulation codes did not account for these variables and extraneous influences. In order to improve our current models we would need to add new functionalities. For one, the current simulation code is missing a function for loss aversion. This is important when examining human subjects because loss aversion directly impacts our expected value calculations (Bullock, Lecture 5). Secondly, the current simulation code lacks a temporal discounting function. Human decision making relies on a temporal context, and our simulation code does not take this into account. Finally, the expected values in the EV map are preset which presents an intrinsic problem. This assumes our mouse is very deliberative, and a live mouse may not exhibit such rational behavior. This prevents our simulation from achieving complete applicability.

Works Cited:

Wilson, J. J., Harding, E., Fortier, M., James, B., Donnett, M., Kerslake, A., . . . Jeffery, K. (2015, August 01). Spatial learning by mice in three dimensions. Retrieved March 30, 2019, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4451476/

Terry, A. V., & Jr. (1970, January 01). Spatial Navigation (Water Maze) Tasks. Retrieved March 30, 2019, from https://www.ncbi.nlm.nih.gov/books/NBK5217/

Starkweather, C. K., Babayan, B. M., Uchida, N., & Gershman, S. J. (2017, April). Dopamine reward prediction errors reflect hidden-state inference across time. Retrieved March 30, 2019, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5374025/

Li, Giovanna, Isnard, Jean, Dreher, & Jean-Claude. (2016, January 25). The Neural dynamics of reward value and risk coding in the human orbitofrontal cortex. Retrieved from https://academic.oup.com/brain/article/139/4/1295/2464198

Hayden, Y., B., Walton, & E., M. (2014, March 31). Neuroscience of foraging. Retrieved March 31, 2019, from https://www.frontiersin.org/articles/10.3389/fnins.2014.00081/full

Bullock, D. (n.d.). *NE340 Lecture 5* [PDF]. Boston: Boston University.