**DATA SIMULATION – README**

FOLDER:

GitHub/RA-Spring2020/2020\_2 Fall/Data and Analysis/Simulations/Code\_Simulation/2. Data Simulation

OVERVIEW:

The Optimal Strategy (OS) in task 2 requires several assumptions, in particular:

* Calculate the exact value of every possible action (investigate R/B, accuse R/B)
* Update probabilities correctly, using Bayes rule
* Accuse only when the value of investigating < value of accusing
* Always pick the action with the highest value (no mistakes, even when values are close)

The scripts in this folder [e.g. **data simulation distribution.py**] generate a fictitious dataset that mimics the behavior of an agent with some characteristics. This dataset is similar to the one generated by humans playing the task, but we know what is the exact “model” that created this output file.

The script operates in four steps:

STEP 1: input these three parameters for the model – optimal policy [see **value function.py**]

* **alpha\_1 = 1 # how much the agent updates own beliefs, after observing a signal that confirms own prior (update in favor of the most likely state) [1=unbiased]**
* **alpha\_2 = 1 # how much the agent updates own beliefs, after observing a signal that contradicts own prior (update in favor of the least likely state) [1=unbiased]**
* **stop\_cost = 0 # stopping cost, how much is the psychological cost of stopping without reaching certainty about the correct state [0 = unbiased]**

In the same part of the script we also include the parameters of the general task (e.g. probability of receiving a revealing signal 25%, prize for being correct 1000 points, etc)

By using values that are different from 1-1-0, we can replace the OS model with other behavioral models, for example

* 0.5 – 0.5 – 0: conservatism (biased updating), always update only half of how much the Bayesian agent would [no stopping cost]
* 1 – 1 – 50: uncertainty cost, the agent suffers a psychological cost (e.g. 50 points) every time that they stop without knowing exactly the correct answer [no update bias]
* 1 – 0.5 – 0: confirmatory updating (unbiased when updating in favor of the prior, but update less when updating against the prior) [no stopping cost]
* And so on…

STEP 2: the dynamic programming algorithm [still in **value function.py**] computes the best policy for this model, and saves a “value function.csv” file with the value of each of the 4 actions for each beliefs state (probability of red being guilty)

STEP 3: input the fourth parameter for the model – behavior [see **data simulation distribution.py**]

* **k = 0.04 # parameter (lambda) in randomness, captures how much the agent will make mistakes in the implementation of the value function**

This parameter is used in the selection of the action (among the four possible ones), as you can see in line 134: choice = randomness(k,V\_stop[p\_index],V\_1[p\_index],V\_2[p\_index])

The chosen action is selected using a **randomness** function [lines 33-46] that uses a logistic function to add some “noise” and allows to make many small mistakes but few large mistakes. The parameter k (lambda) is capturing exactly “how much” of these mistakes we observe, and in particular ranges from 0 (completely random) to infinity (completely optimal). See for example the logit function used to calculate the probability of stopping:  
 p\_stop = 1/(1+math.exp(k\*(v\_red-v\_stop))+math.exp(k\*(v\_blue-v\_stop)))

STEP 4: use the value function (from step 2) and the lambda parameter (from step 3) to generate a full dataset with the behavior of this “stochastic computer” in a large number of trials as the ones that appear in the online experiment [this part is still in **data simulation distribution.py**]