Matching Behavior and the Effect of Other People's Action Related Reward on Decision Making

•••

A variation of the study by

Sugrue, Corrado and Newsome (2004)

Matching behavior:

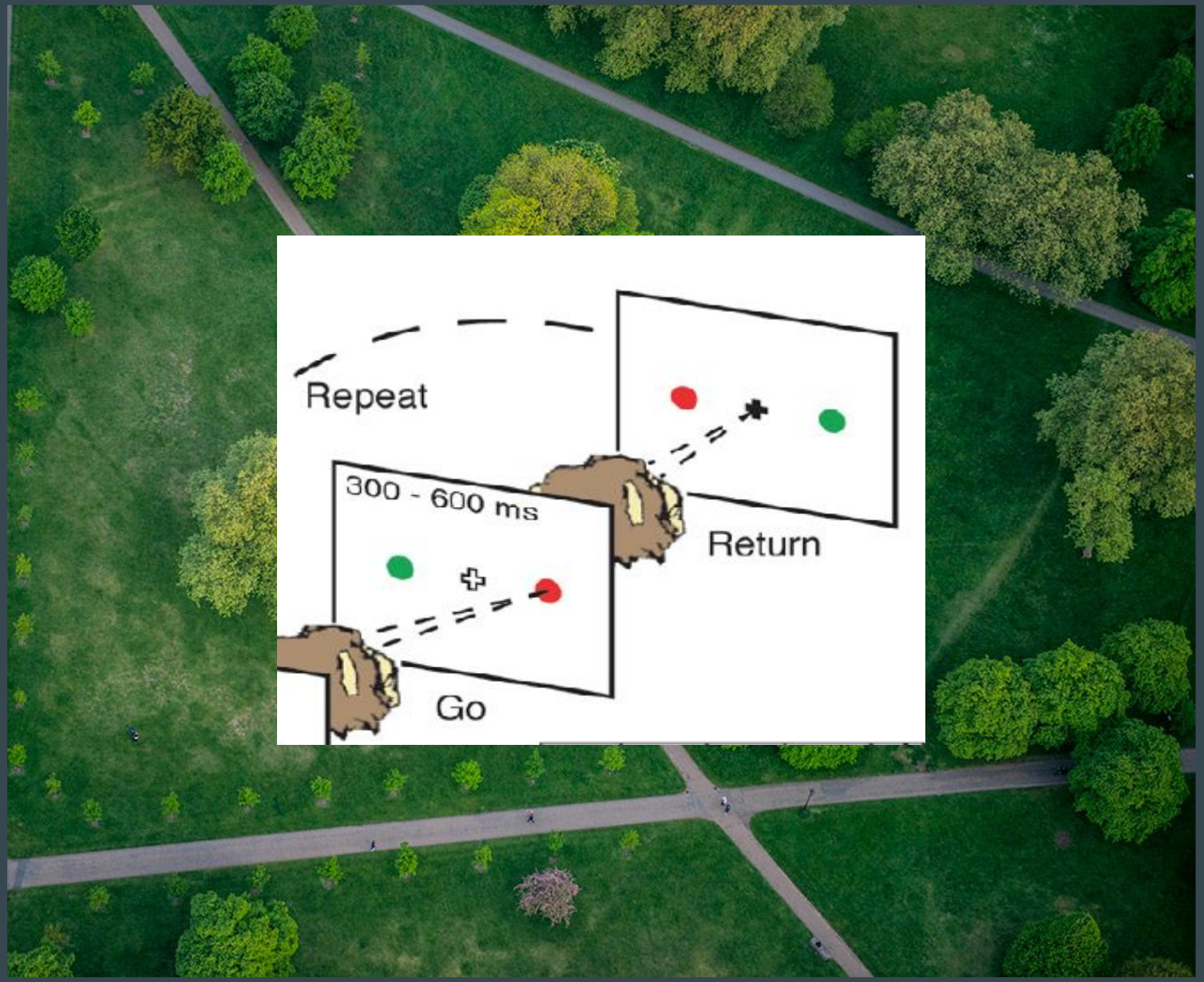
Time spent at a foraging site matches its relative abundance of resources

Matching behavior:

Time spent at a foraging site matches its relative abundance of resources

Matching law:

$$\frac{I_k}{\Sigma I} = \frac{C_k}{\Sigma C}$$



Matching behavior:

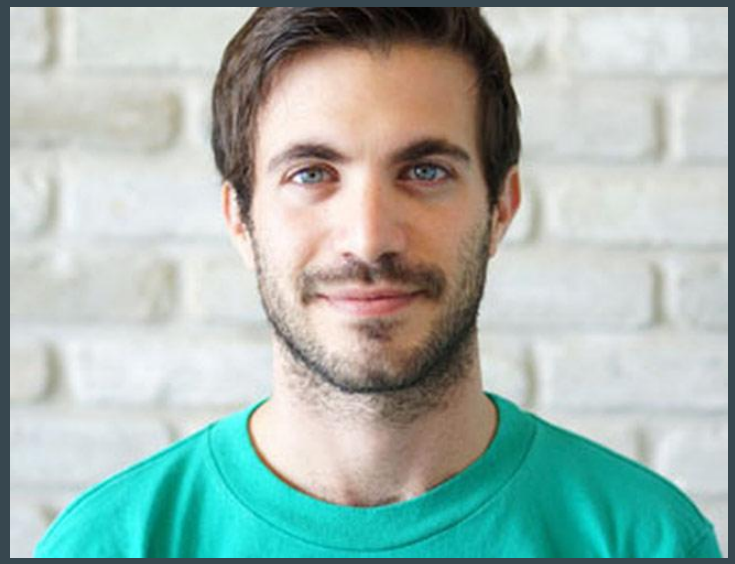
Time spent at a foraging site matches its relative abundance of resources

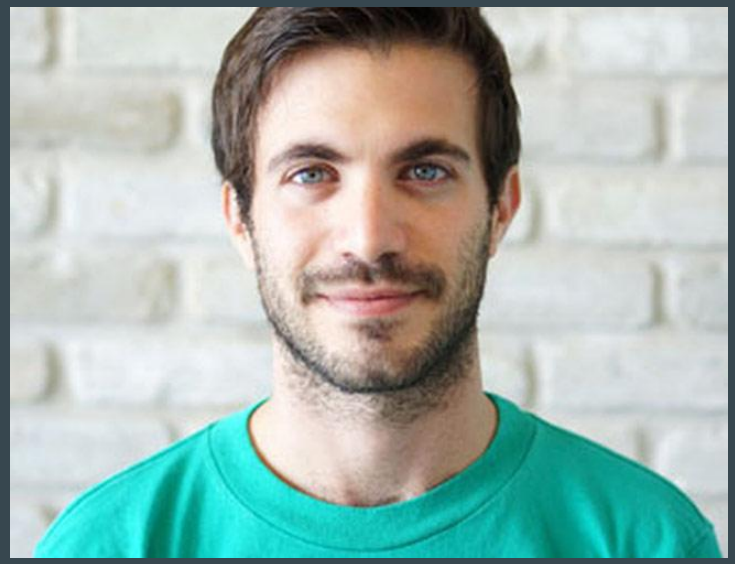
Matching law:

$$\frac{I_k}{\Sigma I} = \frac{C_k}{\Sigma C}$$

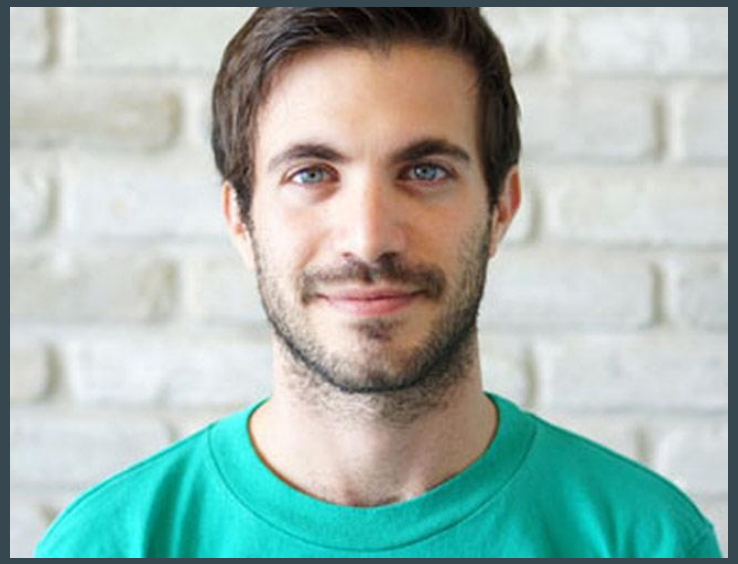
- 1 player
- 2 targets
- discretely changing reward rates

Do humans exhibit matching behavior?

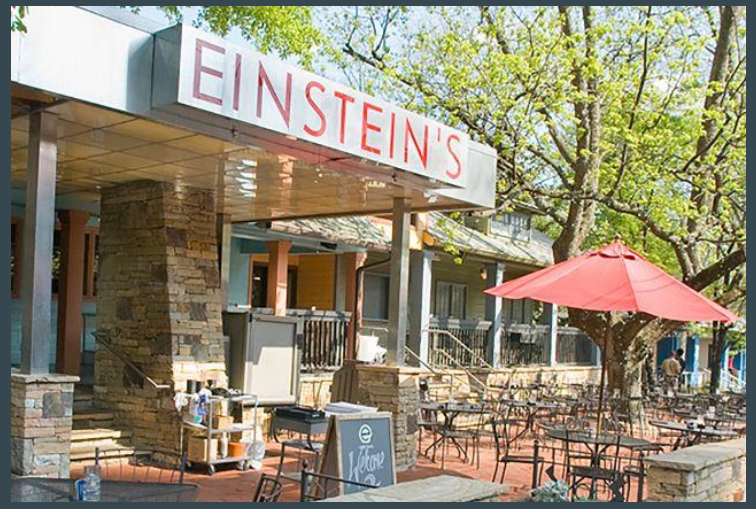


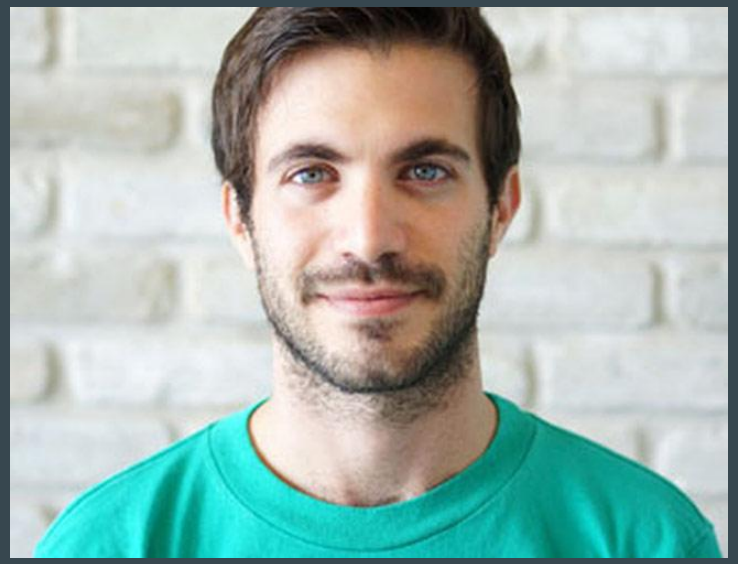


Steven

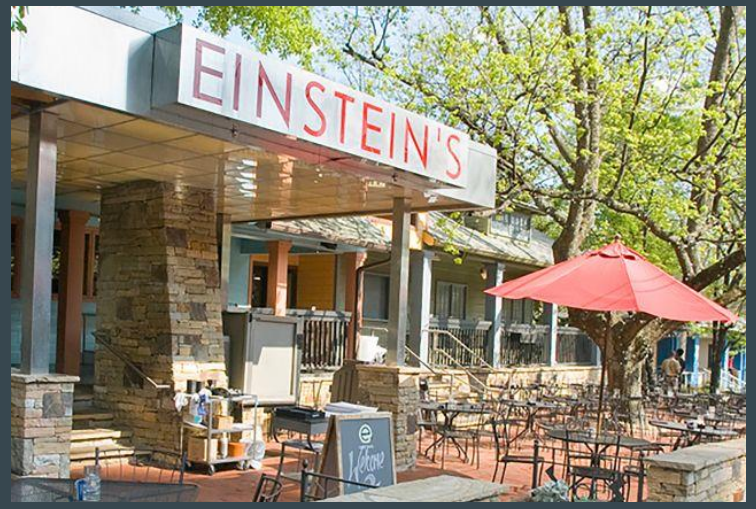


Steven



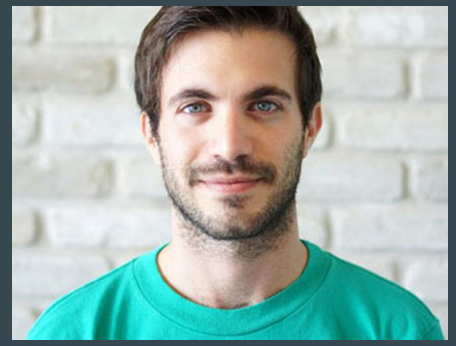


Steven



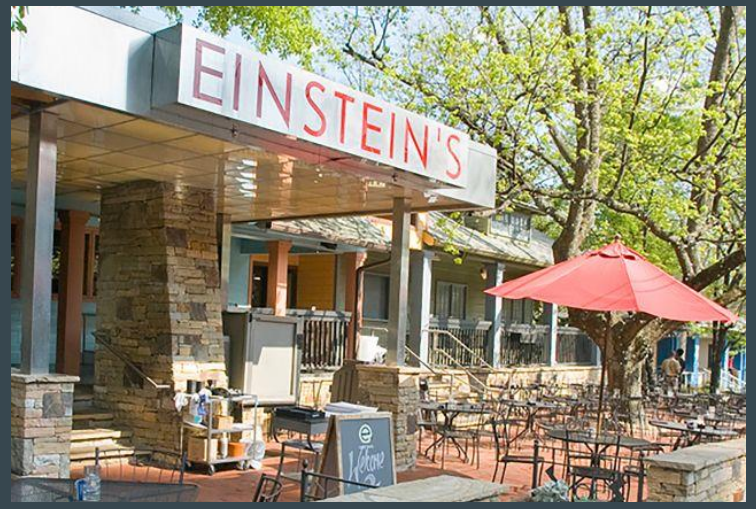
0.5

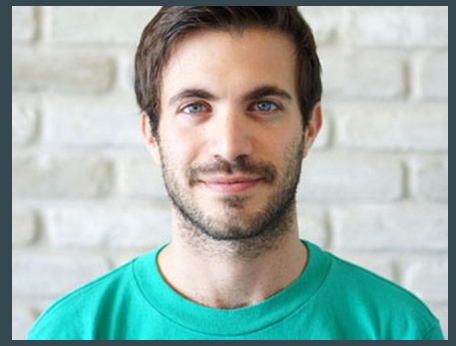
0.5



Steven

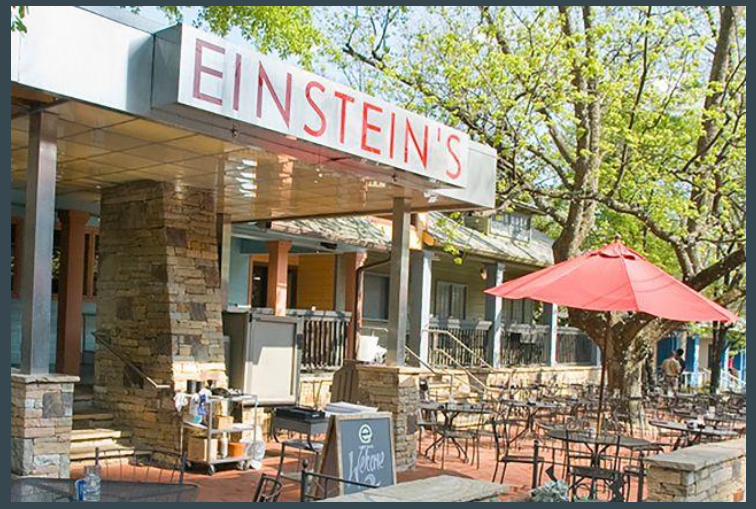
James





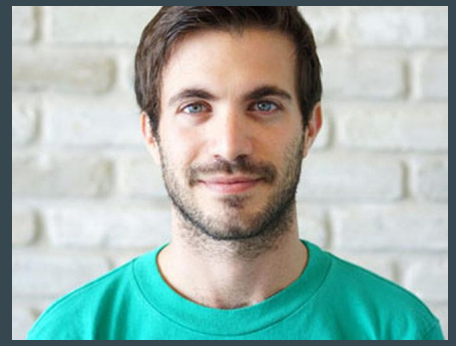
Steven

James



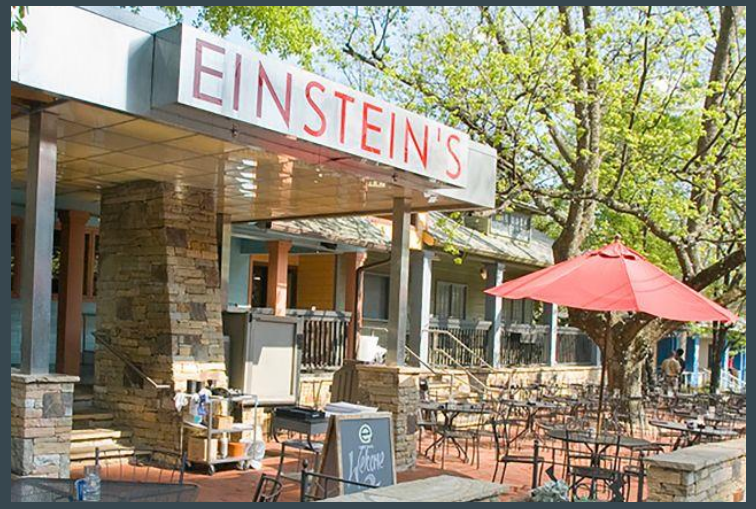
0.2

0.8



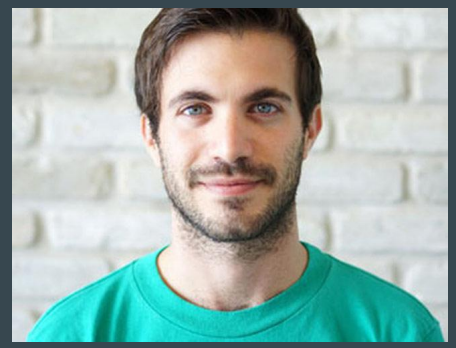
Steven

James



0.2

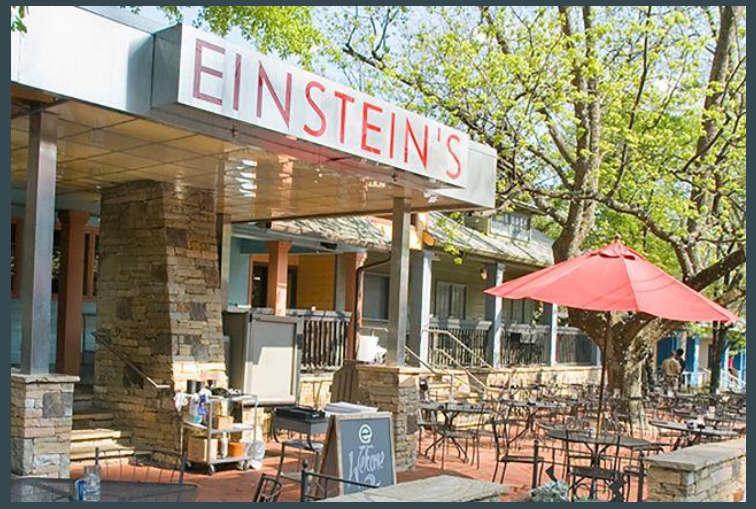
0.8



Steven

?

James



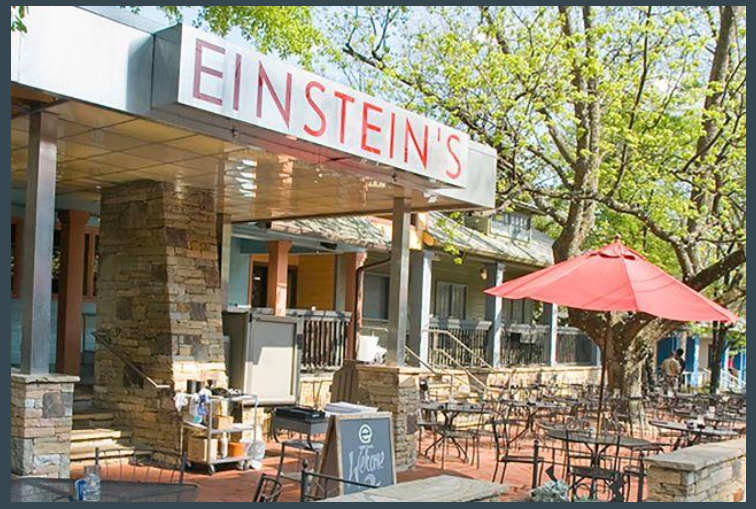
0.2

0.8

Steven

?

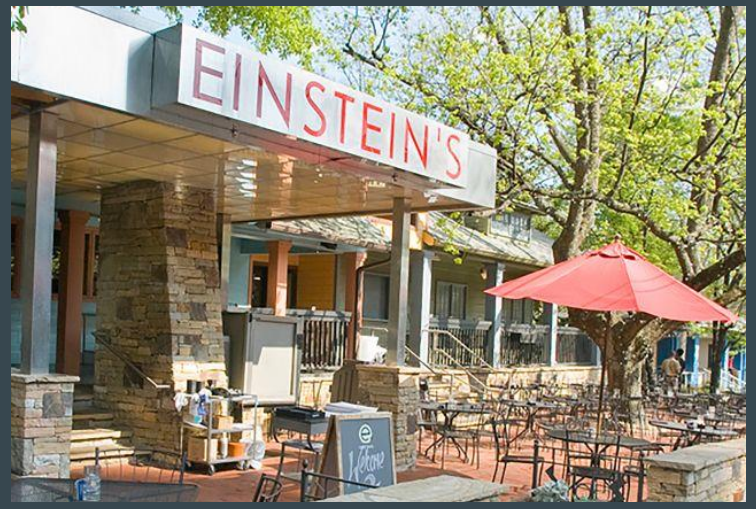
James



Steven

?

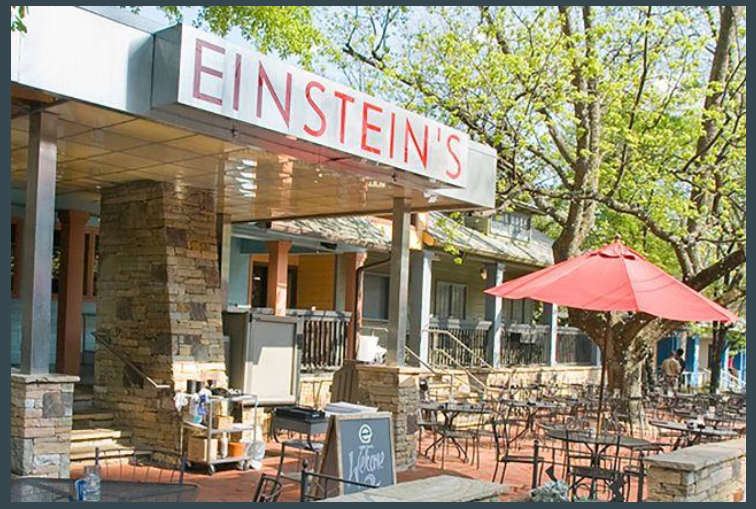
James



Does it make sense to take observations of others' action related reward into account when making decisions for ourselves?

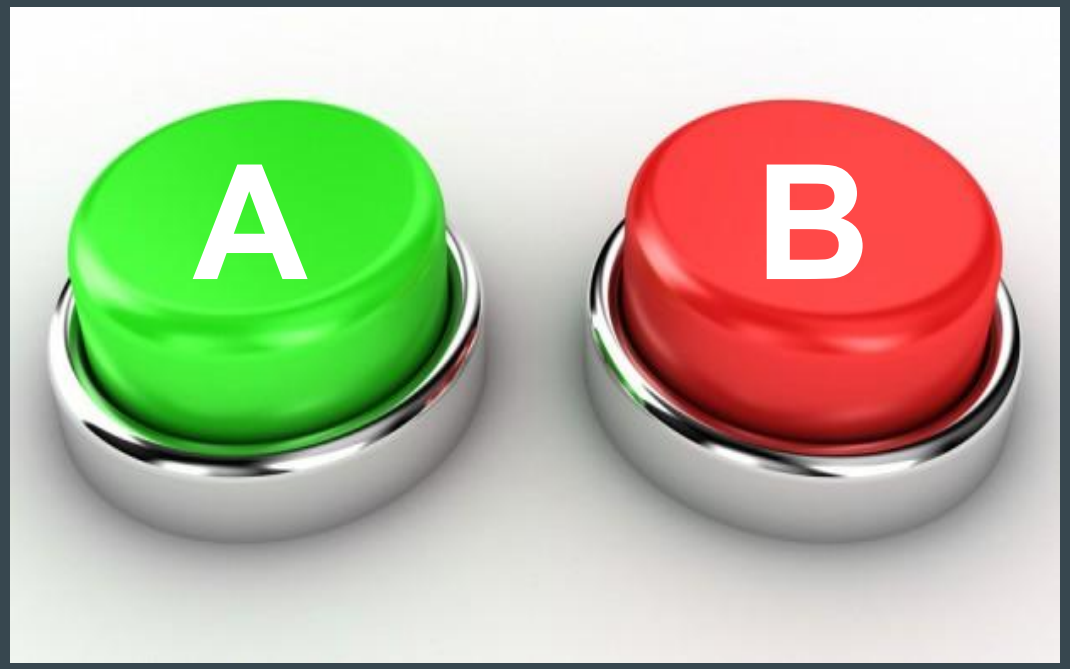
Player 2

Player 1



Player 1:

$P(A_1) = 0.6$ $P(B_1) = 0.6$



Player 2:

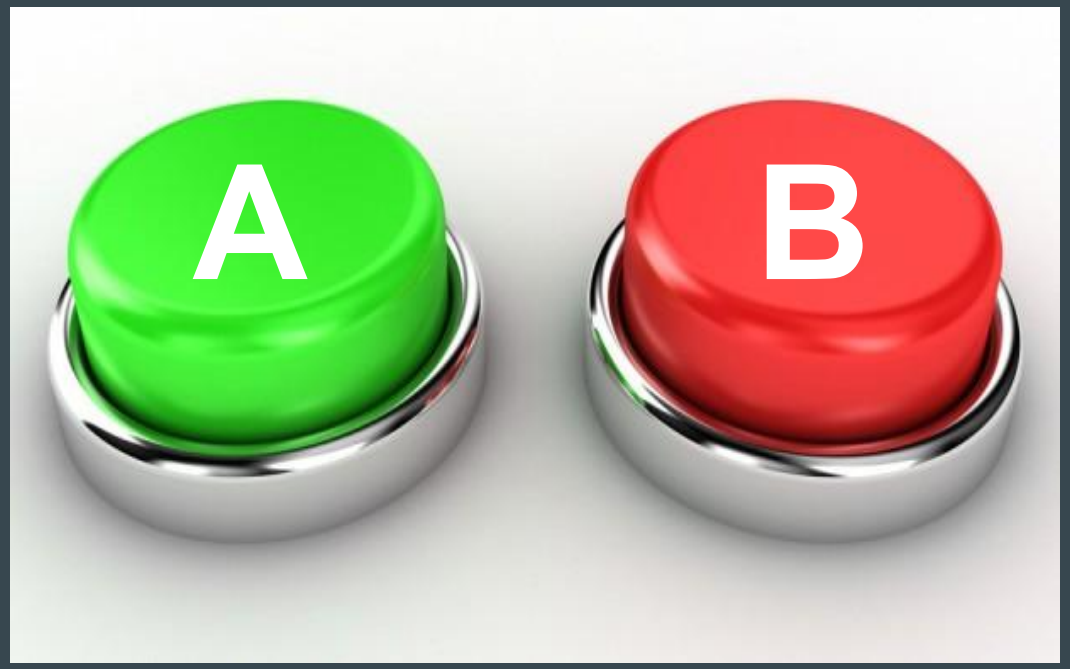
$P(A_2) = 0.2$ $P(B_2) = 0.8$

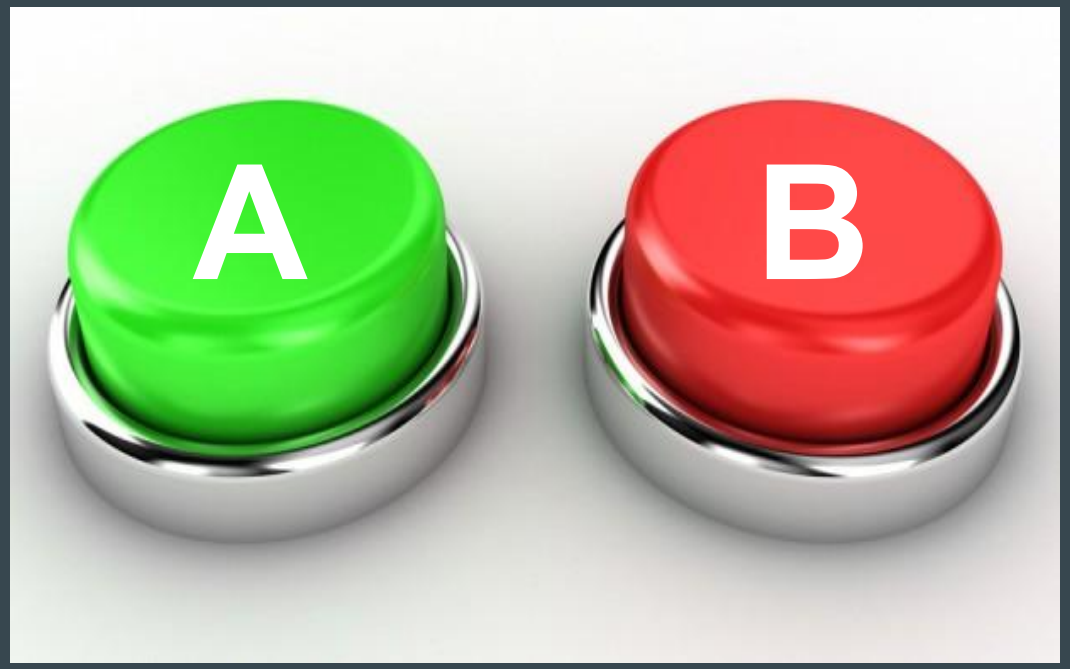
Player 1:

$P(A_1) = 0.6$ $P(B_1) = 0.6$

Player 1:

$P(A_1) = 0.2$ $P(B_1) = 0.8$



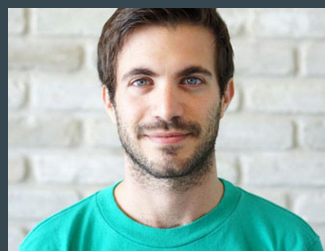


Player 2:

$P(A_2) = 0.2$ $P(B_2) = 0.8$

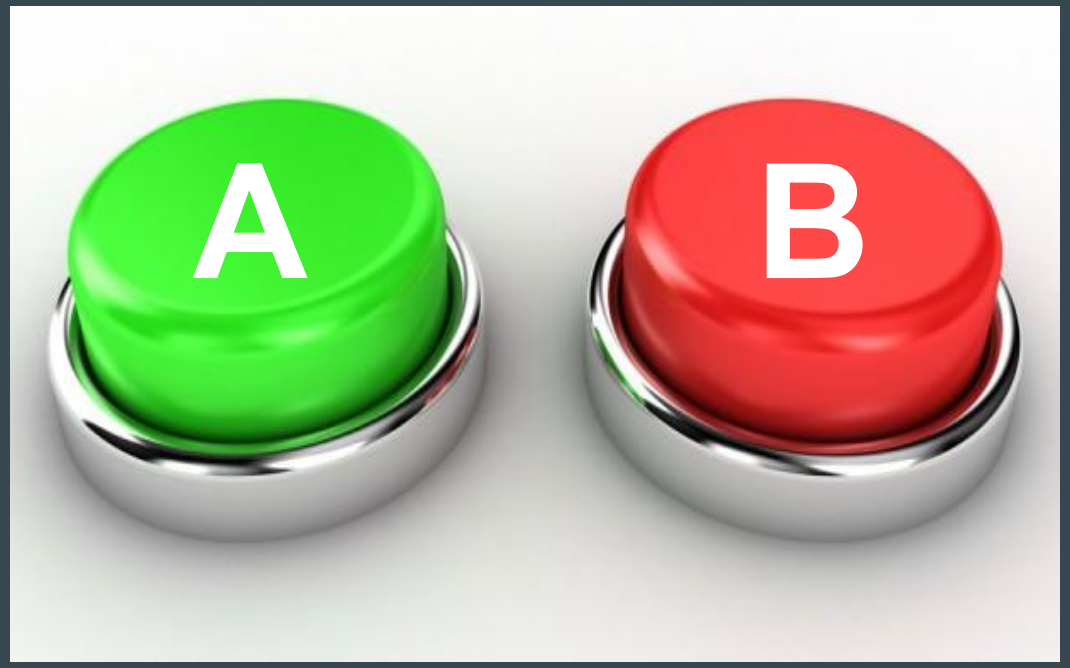
Player 2:

$P(A_2) = 0.2$ $P(B_2) = 0.8$



Player 1:

$P(A_1) = 0.6$ (0.2) $P(B_1) = 0.6$ (0.8)



Player 2:

$P(A_2) = 0.2$ $P(B_2) = 0.8$

Condition 1:
Player 2 **disregards action-related outcome of Player 1**, only factors in its own

Condition 2:
Player 2 takes Player 1's action-related outcome into account with a **weight** (0.25)

Condition 3:
Player 2 takes their partner's and their own action-related outcome equally into account

run

simulation.py | run.py | player.py

https://github.com/andythai/matchsim

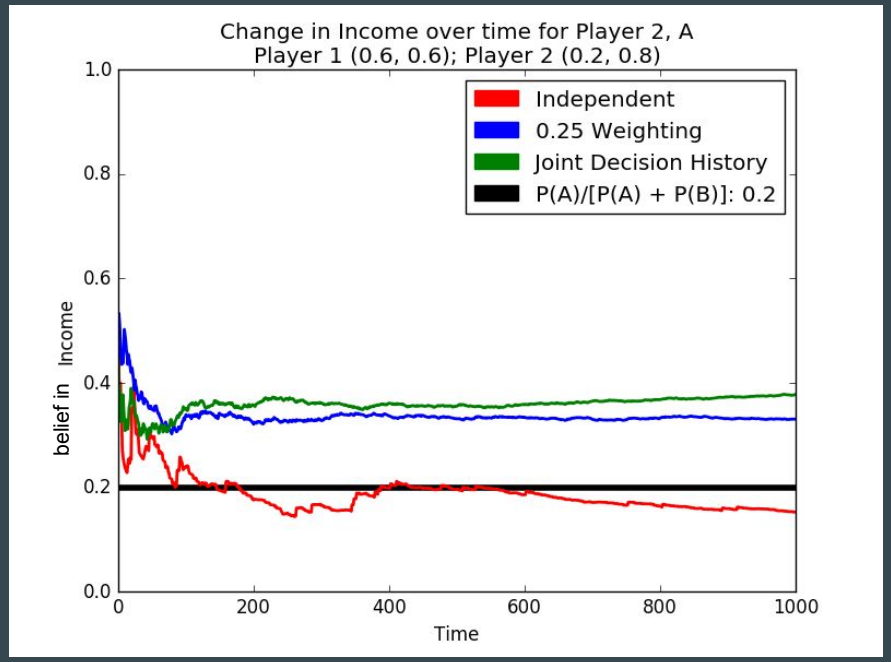
```
from simulation import show_info
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np


# SETTINGS (Only change options here)
# mode; 0: no weight, 1: unequal weight, 2: equal weight
def setup(mode):
    # CHANGE these

    # How much weight to place on partner's actions for mode 1
    # Weight is player1 * (1 - weight) + player2 * (weight)
    # Lower means player 2 plays less attention to player 1
    # Value should be between 0.0 and 1.0
    weight = 0.25

    # Probabilities of reward for each button (a, b) for each player (p1, p2)
    # Values should be between 0.0 and 1.0
    p1_prob_a = 0.2
    p1_prob_b = 0.8
    p2_prob_a = 0.6
    p2_prob_b = 0.6

    # Number of iterations for each simulation
    num_turns = 1000

    # Debug variable, show all income changes if true, otherwise only last income ratio
    # Leave false if you don't know what you're doing
    debug = False

    ##########################################################################################

    # Do NOT change these
    player1 = Player(1, 0, 0, p1_prob_a, p1_prob_b)
    player2 = Player(2, mode, weight, p2_prob_a, p2_prob_b)
    player1.other_player = player2
    player2.other_player = player1
    return player1, player2, num_turns, debug


# main method
# In each trial, player 1 plays without paying attention to the partner
# Only player 2's behavior changes
```

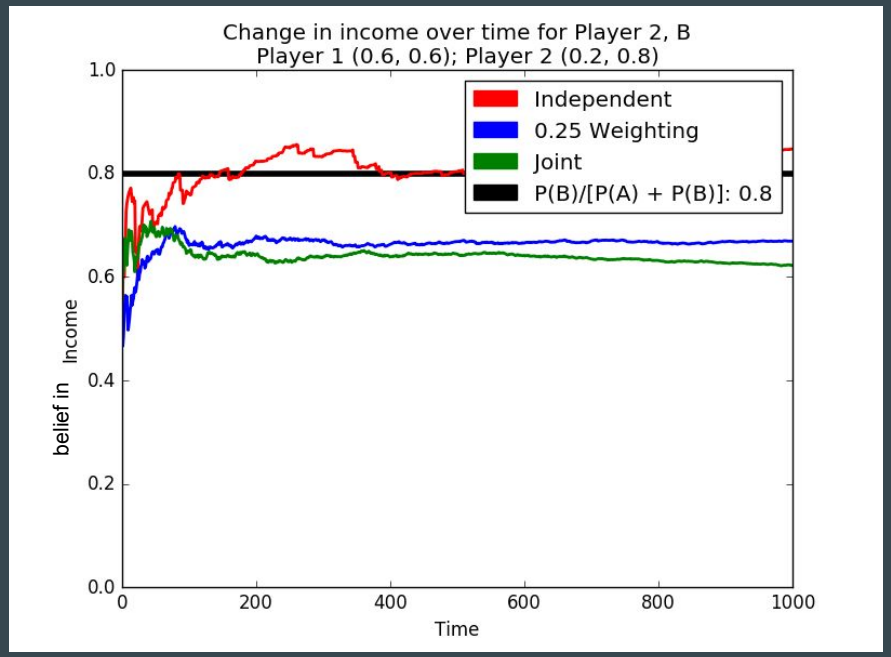
Platform and Plugin Updates
PyCharm Community Edition is ready to update.

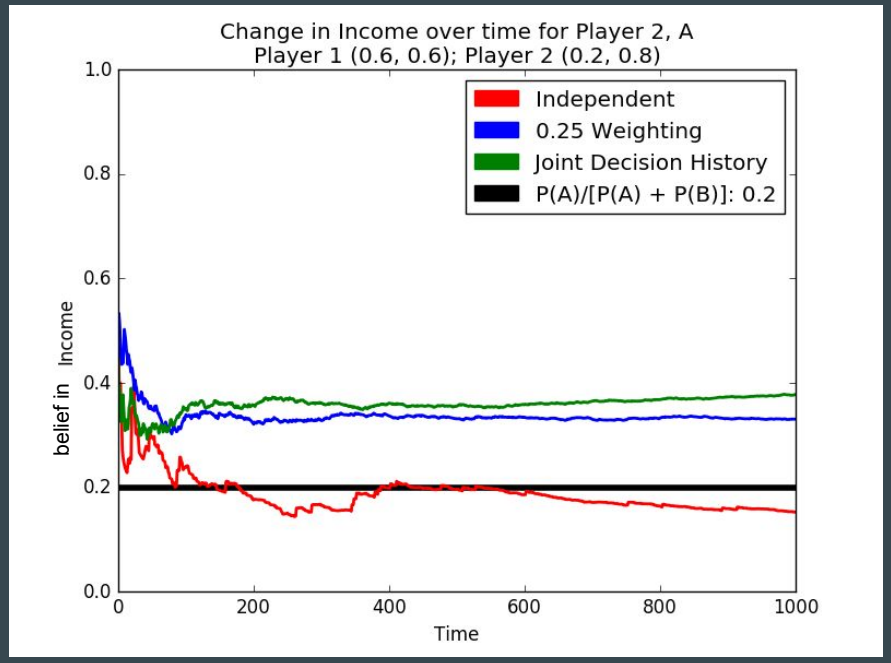
Simulation

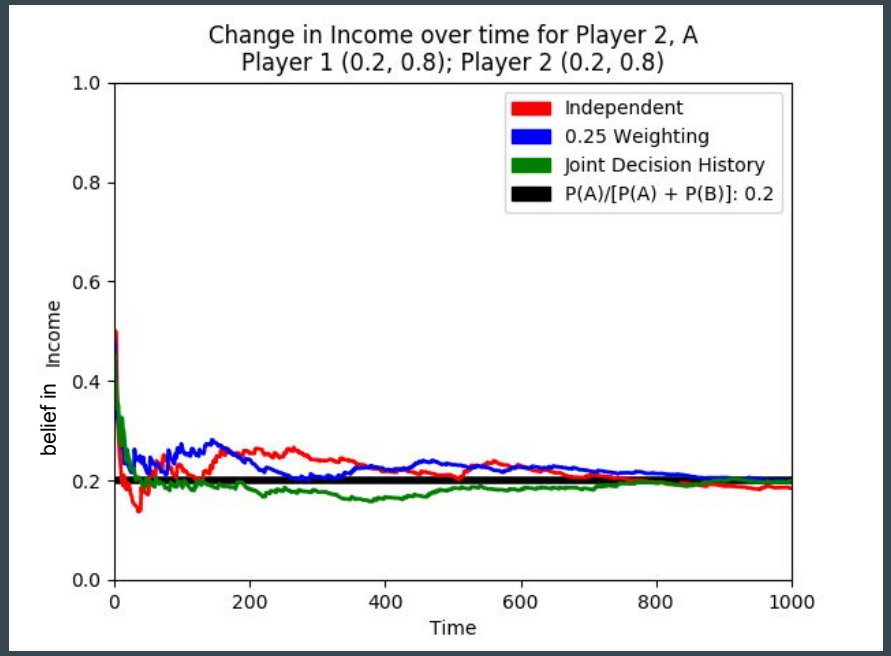
$$\frac{I_k}{\Sigma I} = \frac{C_k}{\Sigma C}$$

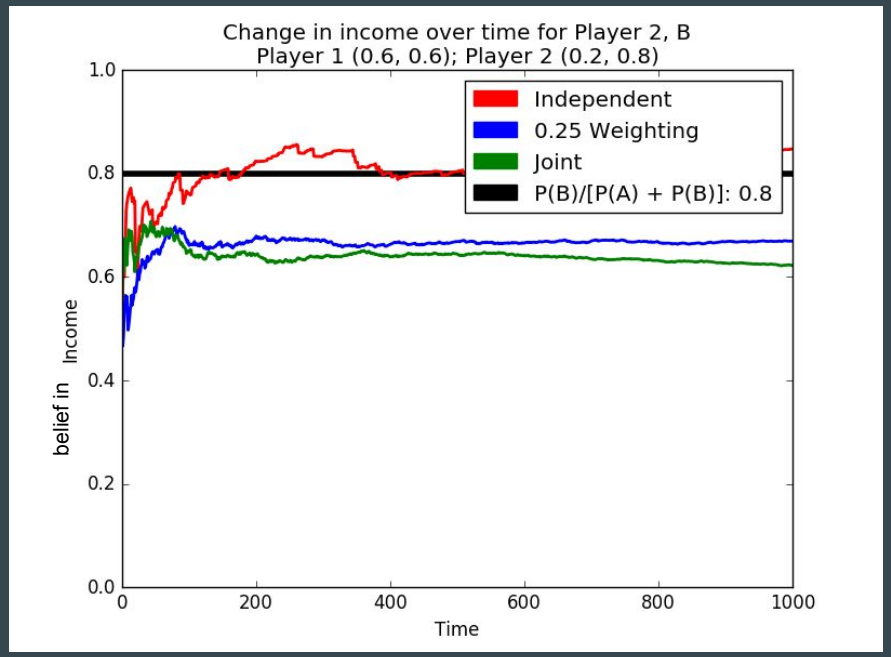
1. Initially start out with 0.5 income for both targets
2. Randomly pick target based on player's belief for income
   a. Example: 0.5 income = 50% chance of picking, 0.75 = 75%, and so on...
3. Randomly determine if player is rewarded based on true reward probability value of the target
4. Append results to player history
5. Repeat process for other player
6. Recalculate belief in income based on updated player history
7. Repeat 2-6 until simulation is over

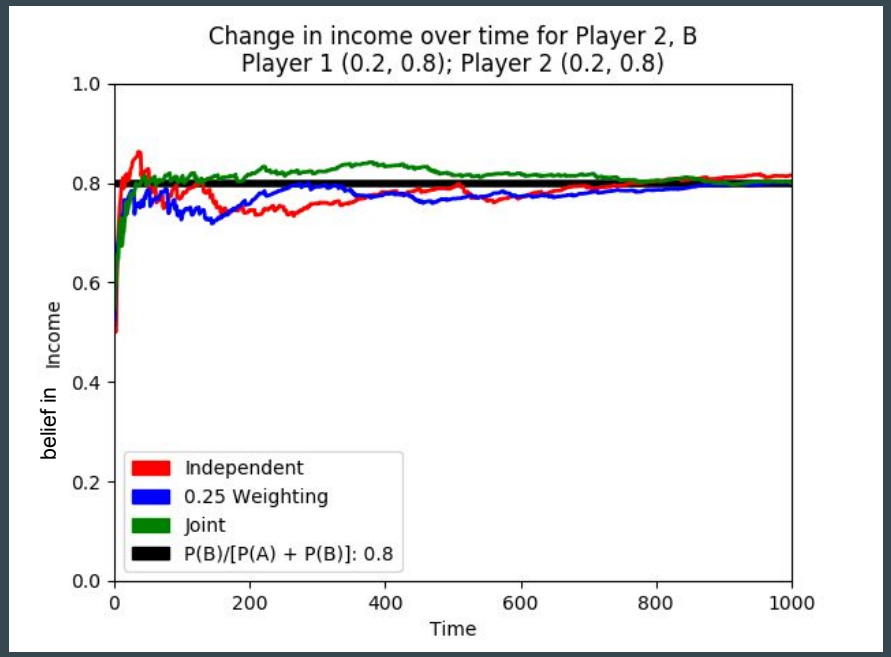

Change in Income over time for Player 2, A
Player 1 (0.6, 0.6); Player 2 (0.2, 0.8)
Independent
0.25 Weighting
Joint Decision History
P(A)/[P(A) + P(B)]: 0.2
belief in Income
Time
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000


Change in income over time for Player 2, B
Player 1 (0.6, 0.6); Player 2 (0.2, 0.8)
Independent
0.25 Weighting
Joint
P(B)/[P(A) + P(B)]: 0.8
belief in Income
Time
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000

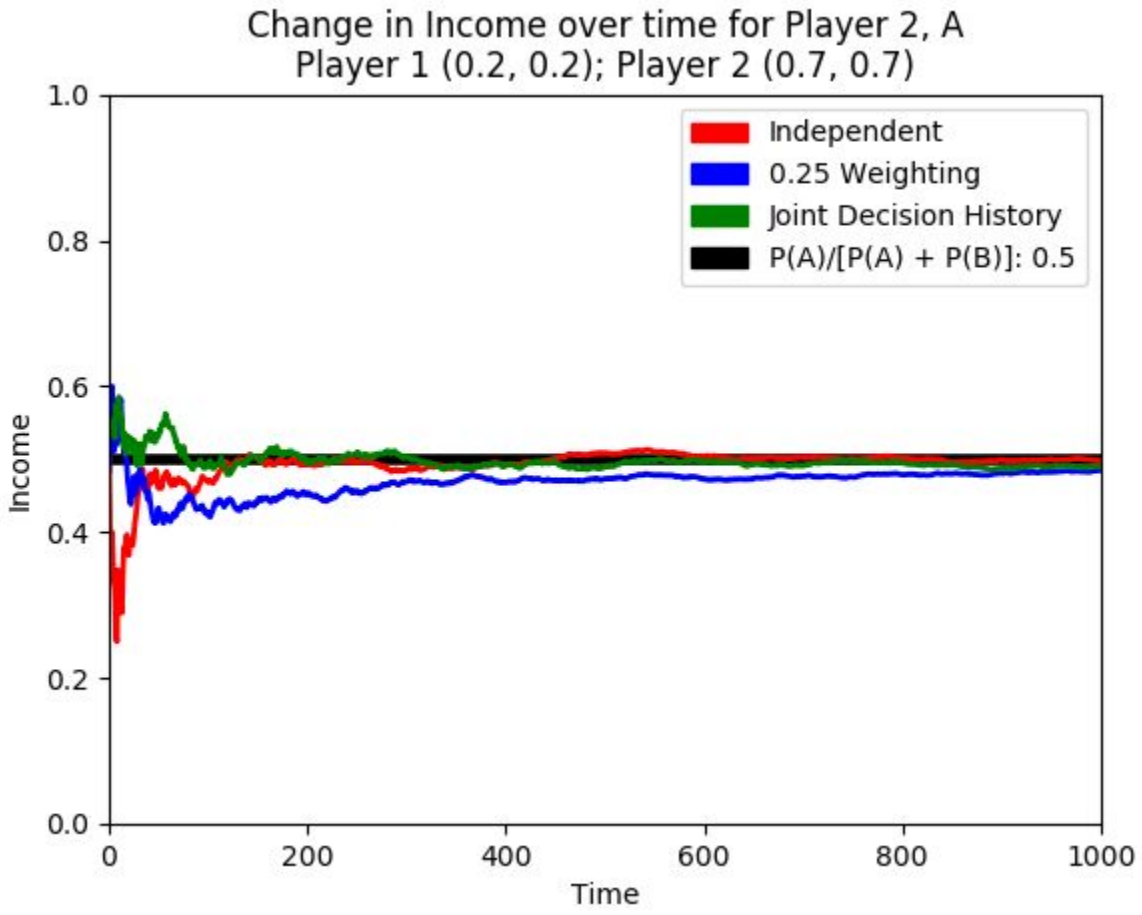

Change in Income over time for Player 2, A
Player 1 (0.6, 0.6); Player 2 (0.2, 0.8)
Independent
0.25 Weighting
Joint Decision History
P(A)/[P(A) + P(B)]: 0.2
belief in Income
Time
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000

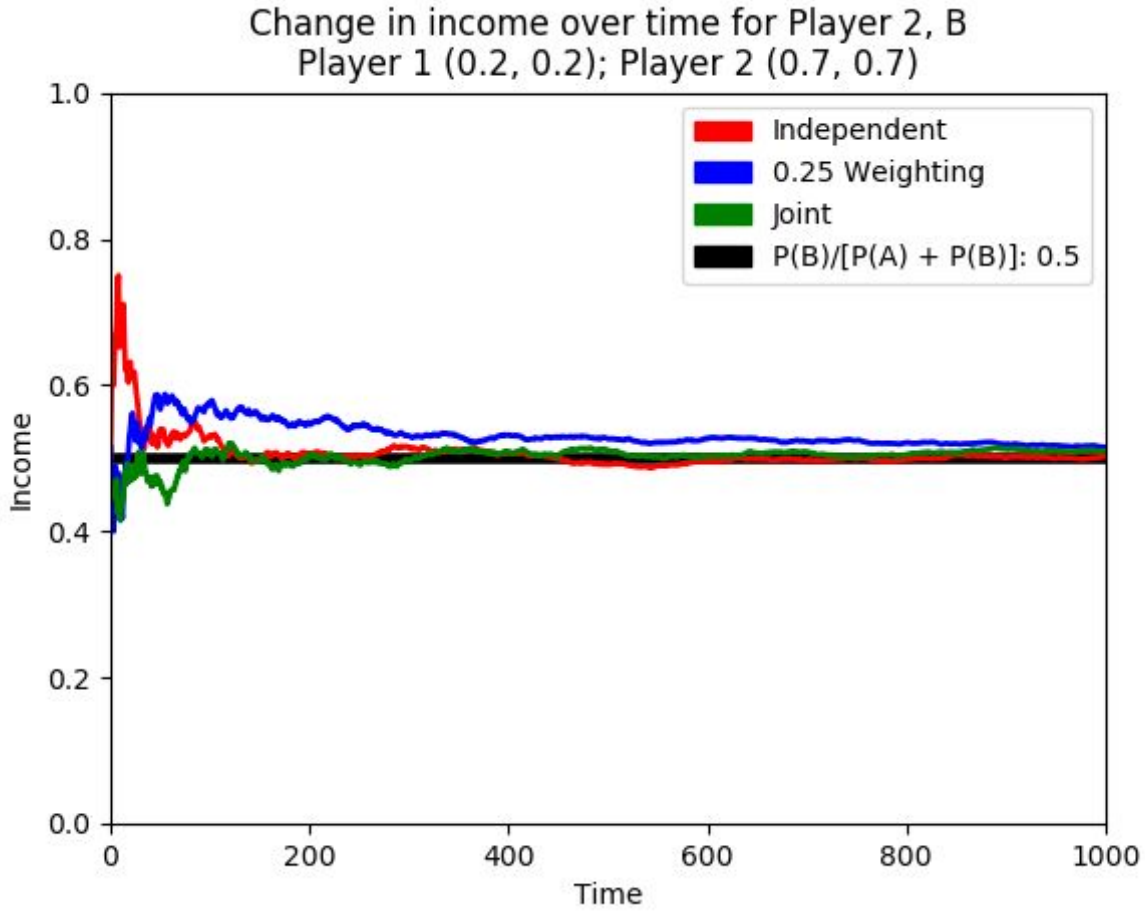

Change in Income over time for Player 2, A
Player 1 (0.2, 0.8); Player 2 (0.2, 0.8)
Independent
0.25 Weighting
Joint Decision History
P(A)/[P(A) + P(B)]: 0.2
belief in Income
Time
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000


Change in income over time for Player 2, B
Player 1 (0.6, 0.6); Player 2 (0.2, 0.8)
Independent
0.25 Weighting
Joint
P(B)/[P(A) + P(B)]: 0.8
belief in Income
Time
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000


Change in income over time for Player 2, B
Player 1 (0.2, 0.8); Player 2 (0.2, 0.8)
Independent
0.25 Weighting
Joint
P(B)/[P(A) + P(B)]: 0.8
belief in Income
Time
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000

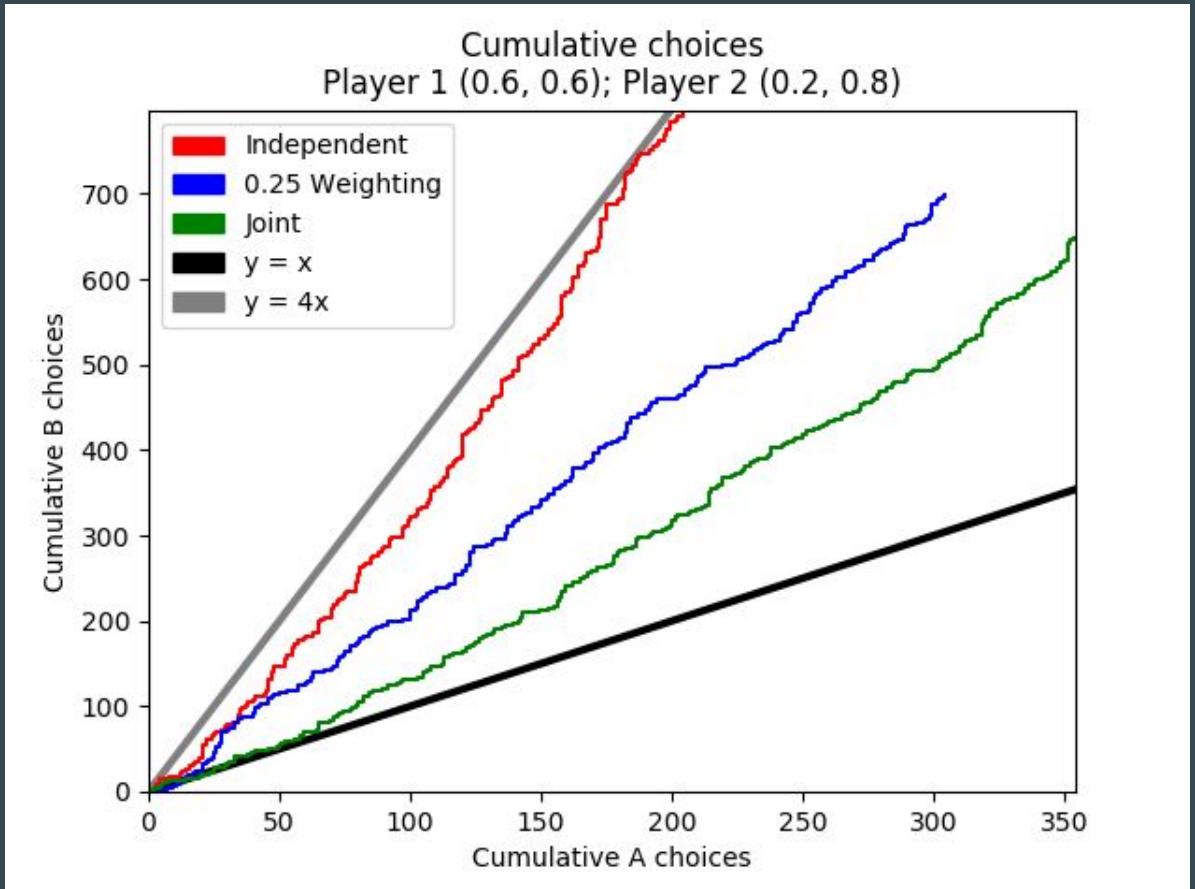
What if they have different reward rates, but the same income?



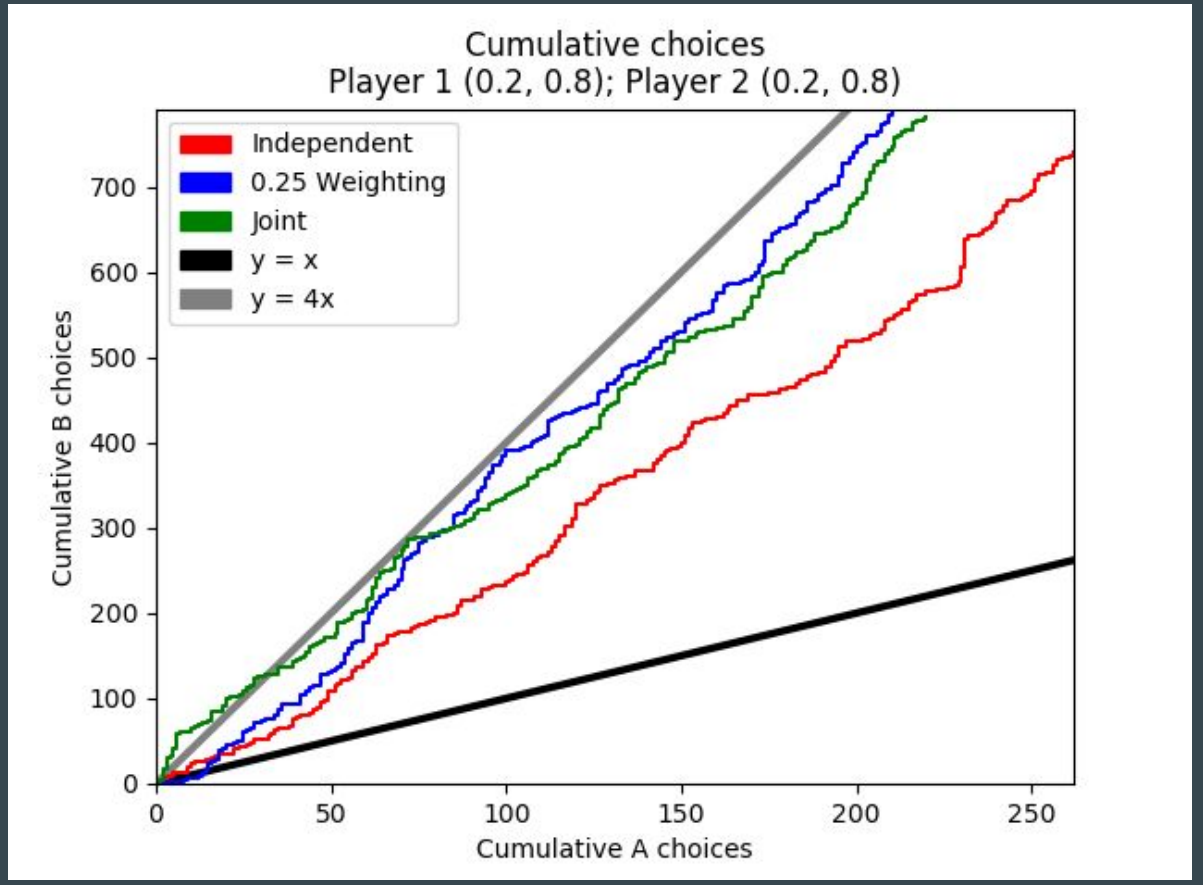


Cumulative Choices

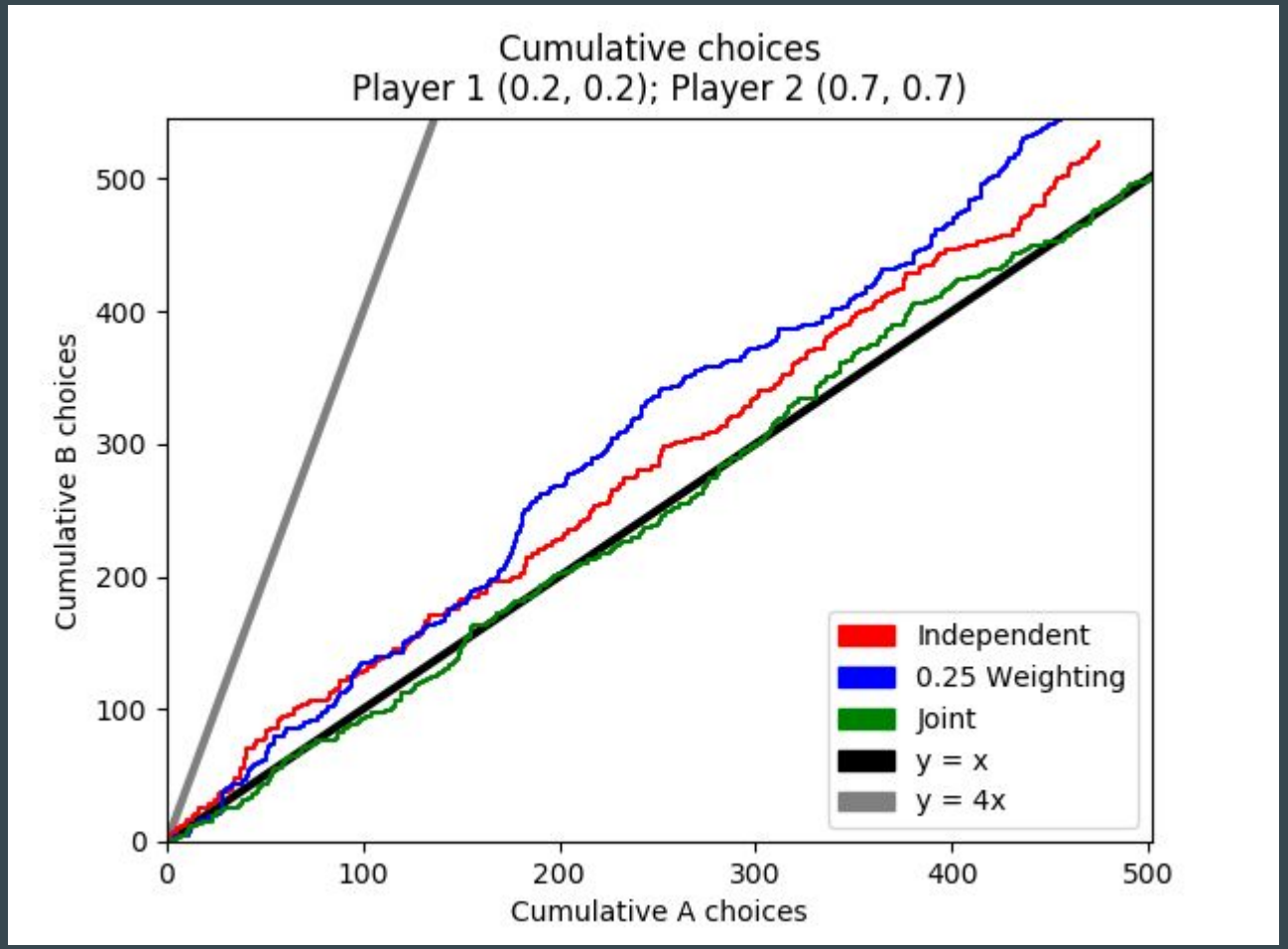
Different



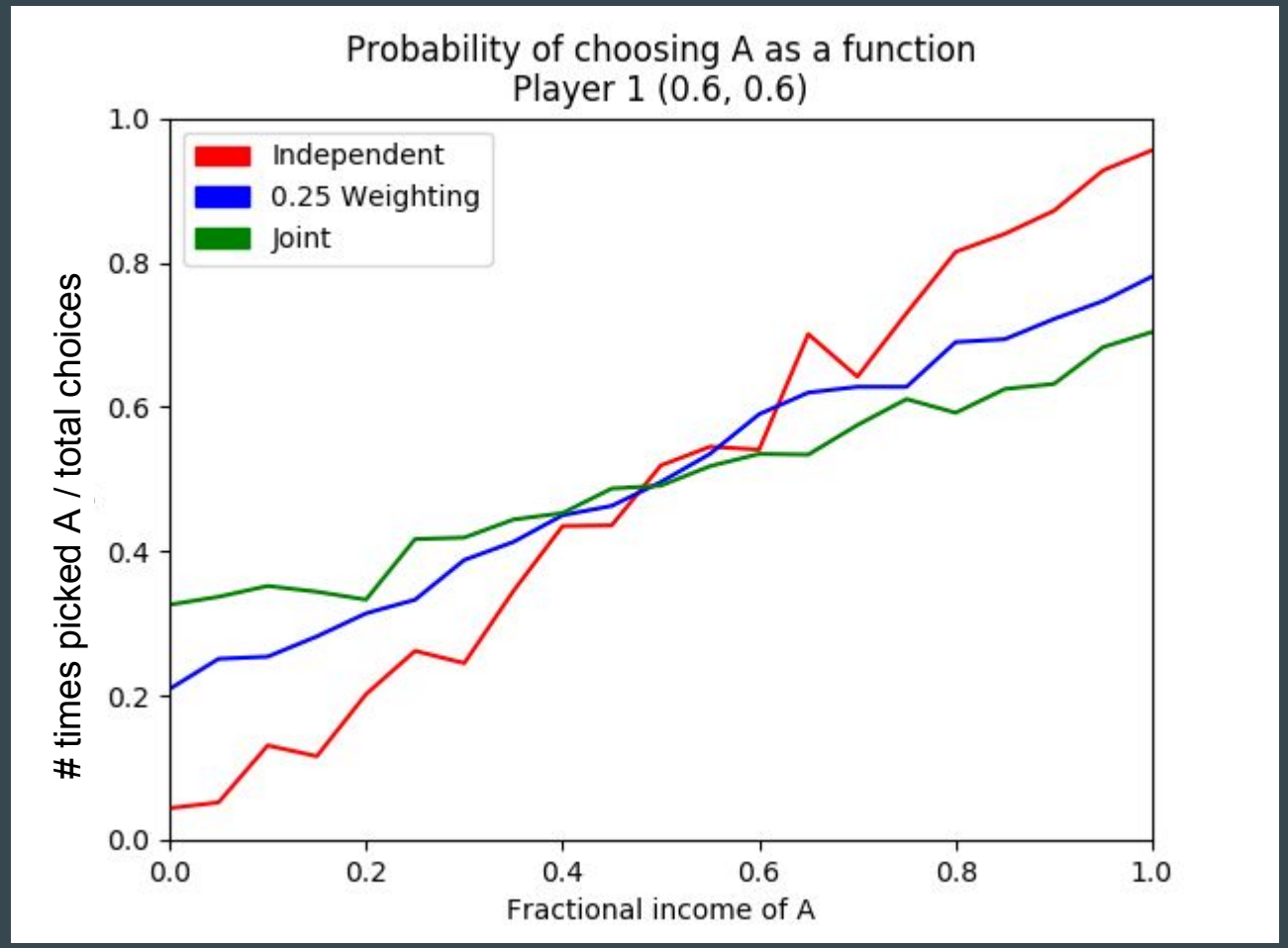
Same



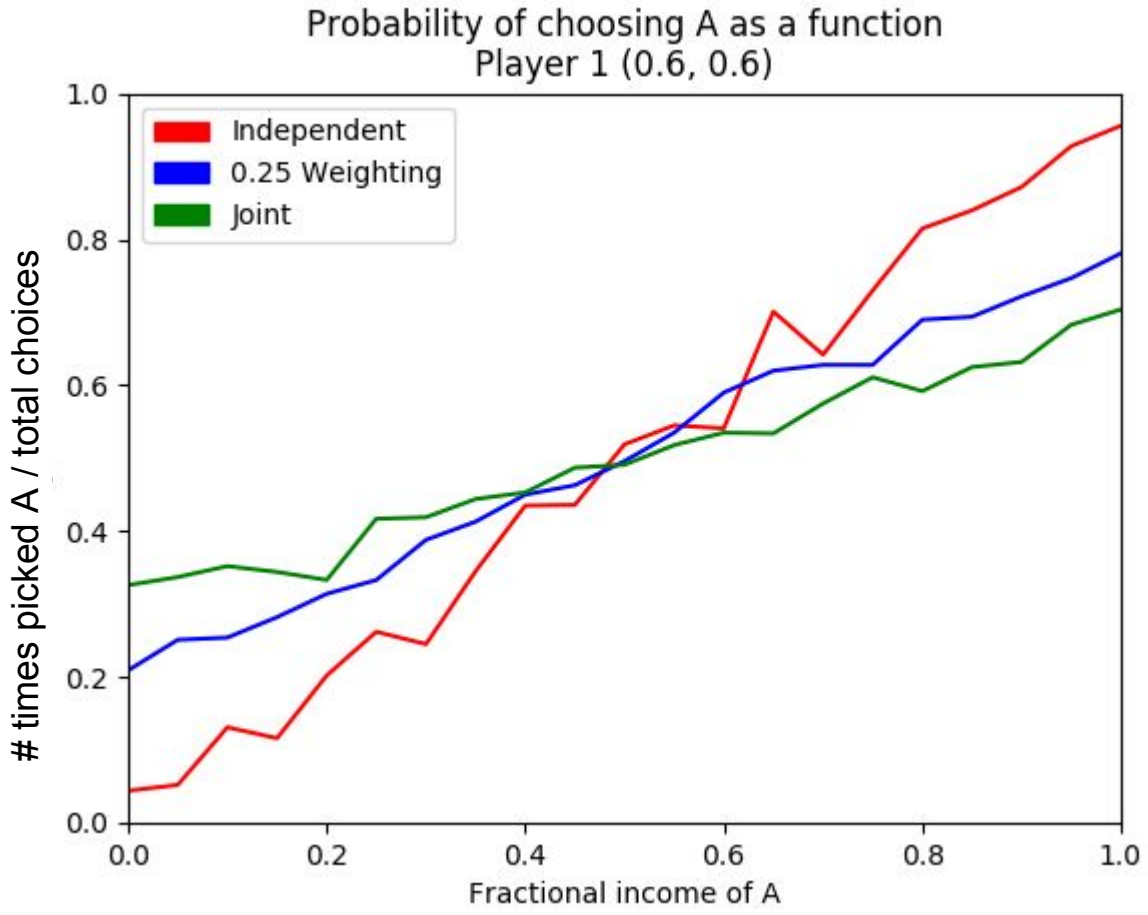
Player1: (0.2, 0.2) Player2: (0.7, 0.7)

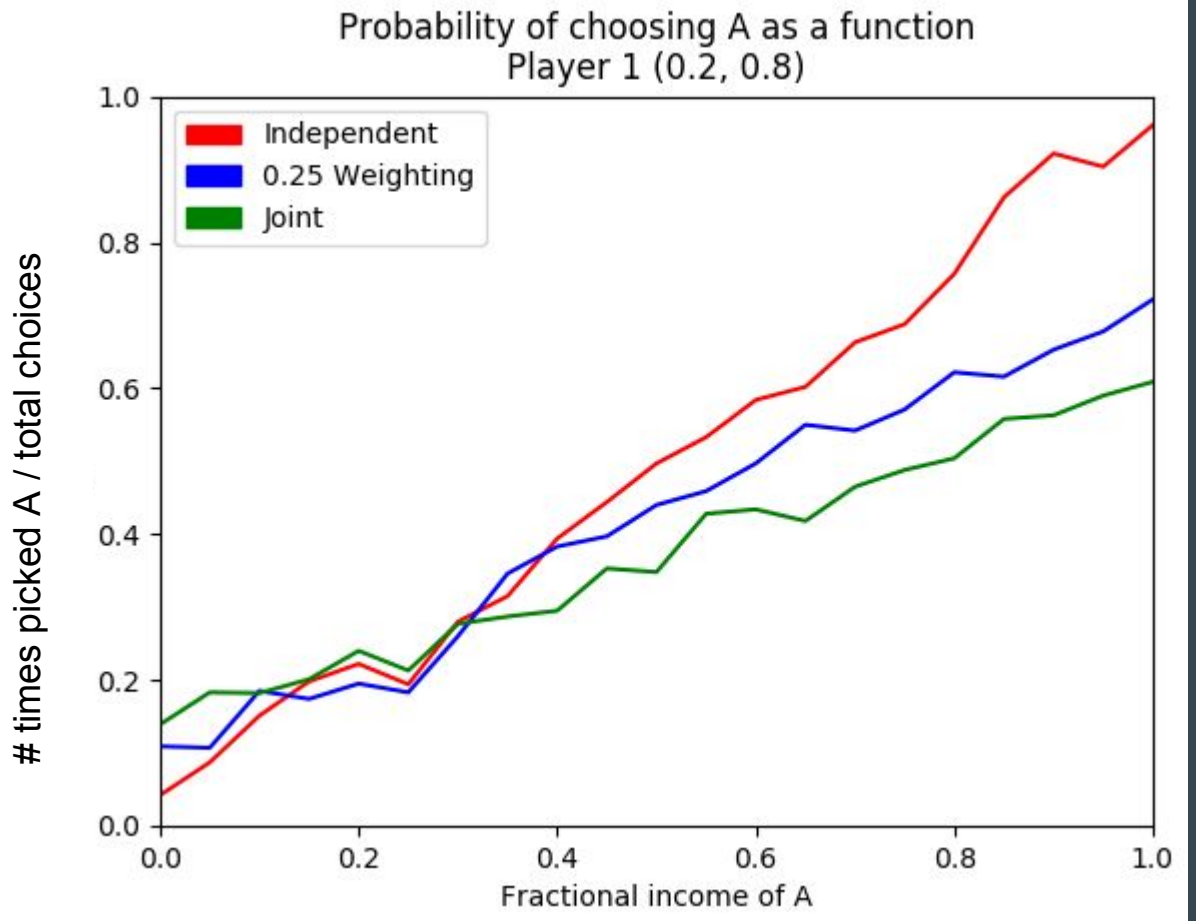


$$\frac{I_k}{\Sigma I} = \frac{C_k}{\Sigma C}$$



$$\frac{I_k}{\Sigma I} = \frac{C_k}{\Sigma C}$$





Take home message

- Matching behavior might not be the optimal strategy but is still widely used.
- Research on optimal strategies for reward-based decision making does not take into account influence by other parties.
- We see that taking into account observations of another individual's outcomes might be beneficial if the reward rates closely match your own.
  - Otherwise, it may be detrimental.
- This is just the first step in exploring other people's actions influence our own decision making processes.

Future steps

Extend the simulation to

- several targets
- several players to observe
- different methods how to integrate action-related outcome
- changes in reward rates over time
- cost of changing targets

Conduct experiments with different populations

- SONA’s (unacquainted players)
- Siblings
- ASD patients

https://github.com/andythai/matchsim

Thank you!

```
from simulation import show_info
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np


# SETTINGS (Only change options here)
# mode; 0: no weight, 1: unequal weight, 2: equal weight
def setup(mode):
    # CHANGE these

    # How much weight to place on partner's actions for mode 1
    # Weight is player1 * (1 - weight) + player2 * (weight)
    # Lower means player 2 plays less attention to player 1
    # Value should be between 0.0 and 1.0
    weight = 0.25

    # Probabilities of reward for each button (a, b) for each player (p1, p2)
    # Values should be between 0.0 and 1.0
    p1_prob_a = 0.2
    p1_prob_b = 0.8
    p2_prob_a = 0.6
    p2_prob_b = 0.6

    # Number of iterations for each simulation
    num_turns = 1000

    # Debug variable, show all income changes if true, otherwise only last income ratio
    # Leave false if you don't know what you're doing
    debug = False

    ##########################################################################################

    # Do NOT change these
    player1 = Player(1, 0, 0, p1_prob_a, p1_prob_b)
    player2 = Player(2, mode, weight, p2_prob_a, p2_prob_b)
    player1.other_player = player2
    player2.other_player = player1
    return player1, player2, num_turns, debug


# main method
# In each trial, player 1 plays without paying attention to the partner
# Only player 2's behavior changes
```