Evolutionary Dynamics of the Trust Game with Observation

Daniele Kordas
15183394

MSc Economics, Behavioural Economics and Game Theory

12th of August 2024

Supervisor: Matthijs van Veelen
Second reader: Theo Offerman

Statement of Originality

This document is written by Student Daniele Kordas who declares to take full responsibility for the contents of this document.

I declare that the text and the work presented in this document are original and that no sources other than those mentioned in the text and its references have been used in creating it. I have not used generative AI (such as ChatGPT) to generate or rewrite text.

UvA Economics and Business is responsible solely for the supervision of completion of the work and submission, not for the contents.

Abstract

This paper researches the evolutionary dynamics of the trust game with observation. It does so through an initial theoretical overview of the game's characteristics and how it is played in standard game theory and laboratory experiments. This is followed by a description of the model used in the analysis that consists of a series of standard trust games with different observability parameters *p*. Based on the value of *p*, a certain ratio of trustors is allowed to observe the trustees' strategy and play accordingly. A series of agent-based simulations has been run to test the model. The result confirmed the hypothesis that observability increases cooperative behavior in both roles. The data is presented first in an aggregate form to better observe the effect of the changes in *p*, and then it is separated to analyze it at the single variable level. With the help of closeup graphs and heatmaps, we have been able to isolate the mechanism behind the rise in cooperation rates.

Index

1. Introduction

It has always been a quest worth undertaking, the one for the reason of cooperative behavior in humans. Finding the very first reason we engage in prosocial behavior has countless applications in several fields, from economics to biology to industrial organization. It is also an additional tile in the puzzle of what makes us human. There is not a single omni-comprehensive explanation of why Homo Sapiens or its ancestors evolved cooperation in the way we can see it now[1]. There are many hypotheses, some receive more support and regard than others, but, as of now, any of them can still be proven true.

One of the instruments used to research pro-social behavior is game theory, more specifically by looking at games that present the players with a decision between cooperative and non-cooperative behavior. One of these games, and the main focus of this paper, is the trust game. The trust game is played between two players. The first one gets an endowment and can decide how much of it to send to the second player. This amount of money is doubled, then it is the turn of the second player that, if any amount of money was sent towards him, has to decide a proportion of it to send back. Let's assume, for the sake of simplification, that the size of the trustor's endowment is 1. If the trustor decides to keep the endowment he will walk out with a payoff of 1, if he sends the money he will earn an amount equal to the amount sent back from the trustee. The trustee will earn money only if the trustor sends, and the amount he will earn is equal to the total of money sent times the multiplier, after taking out the part he is going to send back to the trustor[2]. We can already see that both the choice of sending the money for player 1 and returning a positive amount for player 2 are not optimal if we work with the usual assumptions of game theory: rationality, common knowledge of rationality, and the players' goal is to maximize their payoff. One reason could be that the players exhibit prosocial behaviors, as opposed to selfish behaviors. Many explanations have been researched to explain the evolution of prosocial behaviors, such as population structure and repeated games. The focus of this study will be the possibility of

[1] Langergraber, K., Schubert, G., Rowney, C., Wrangham, R., Zommers, Z., & Vigilant, L. (2011). Genetic differentiation and the evolution of cooperation in chimpanzees and humans. *Proceedings of the Royal Society B: Biological Sciences*, *278*(1717), 2546-2552.

[2] Thielmann, I., Böhm, R., Ott, M., & Hilbig, B. E. (2021). Economic games: An introduction and guide for research. *Collabra: Psychology*, *7*(1), 19004.

observation, in particular, we will focus on the possibility for the trustor to observe the return rate of the trustee.

The main question behind this paper is: does the introduction of observation generate cooperative behavior?

The study can be a starting point to open discussions on different topics. Did observation play a role in evolving cooperative behavior in humans and other animals? Does observation help to generate trust between businesses? Does it open the door for other trust-inducing mechanisms like commitment? The analysis will consist of an agent-based simulation, with 100 agents playing each other for extended periods. We will observe the dynamic of the main strategies employed by trustors and trustees, the cooperation rates, and how these elements affect each other. Most importantly we will observe how different levels of observability will affect those relationships and if they will, in fact, raise the cooperation rate. We will see how this is the case and how the cooperation rate rises proportionally to the parameter of observability used. The results will show that not only does the cooperation rate jump up when the trustors can observe the return rate of the trustees, but also the sending rate of the non-observers will be higher as an effect of the trustees raising the return rate.

In the next chapter, we will dive into the existing literature and highlight the difference between the game theoretical dynamic of the trust game and the experimental results, with a reflection on the evolutionary reasons behind that. Afterward, we will explore the methodology used to obtain new insights into the dynamic of the game and its limitations. Then we will show and discuss the results obtained from the simulations and how they relate to the existing literature. Lastly, we will draw conclusions from the analysis, its limitations, and further research possibilities.

2. Theory and Literature

In this chapter, we will discuss the theory underlying the simulation and what the existing literature has to say about the trust game and its dynamics. We will highlight the results obtained in lab experiments, where they show a consistent level of cooperation rate.

2.1 Observability

The simulation introduces the new parameter observability, represented by the letter p. More precisely we are talking about probability of observation, the ratio of chances that the trustor can observe the return rate of the trustee before deciding to send or not to send the money. This parameter will be the main focus of the simulation, to try to understand how it influences the cooperation rates and which are the mechanisms behind it. The idea is that, by observing the return rate of the trustees, the trustors can develop a more complex strategy that takes into account the new information. The blind send/not send paradigm will be replaced by a decision that depends on the strategy of player 2, in the simulation this strategy will take the form of a threshold. After observing the trustor will send the money if the return rate is higher than the threshold and not send the money if it is shown to be lower. It will not only give a more complex strategy space to the trustor, which now will have both a strategy in case of not observing and one in case of an observation happening, but it will also cause the trustee to change its strategy based on the new behavior of player 1. This change is intended in an evolutionary sense; the strategies in the simulation are not caused by intentionality but they are the result of random evolutionary processes.

2.2 Trust Game Dynamics

With the introduction of an observability parameter, things get more complicated. Now the payoff of player 2 is influenced by the actions of observers and non-observers, a useful way to write it down as a weighted average of the two cases:

$$Payoff(P2) = 2p(1 - r)y + 2(1 - p)(1 - r)x$$

where *p* is the observation rate, *r* is the return rate of player 2, *x* is the strategy of player 1 when non-observing, and *y* is a parameter that tells if the threshold of player 1 when observing is met by the return rate. At first glance, it could seem a complicated calculation, but the variables *x* and *y* are binary values that are there capture the strategies of player 1, we can rewrite the formula by dividing it into 4 cases, corresponding to the actions of the trustor when not observing and when observing. We will call th the threshold of player 1 and use π to indicate the payoff.

If the trustor doesn't send with no observation and $th > r$:

- $\pi(P2) = 0$

If the trustor sends with no observation and $th > r$

- $\pi(P2) = 2(1-p)(1-r)$

If the trustor doesn't send with no observation and $th \leq r$

- $\pi(P2) = 2(p)(1-r)$

If the trustor sends with no observation and $th \leq r$

- $\pi(P2) = 2(p)(1-r) + 2(1-p)(1-r)$

If we were to do the same for the payoff of player 1 we would get:

- $\pi(P1) = 1$ for $x = 0$ and $y = 0$
- $\pi(P1) = 2(1-p)r$ for $x = 1$ and $y = 0$
- $\pi(P1) = 2(p)r$ for $x = 0$ and $y = 1$
- $\pi(P1) = 2(1-p)r + 2(p)r$ for $x = 1$ and $y = 1$

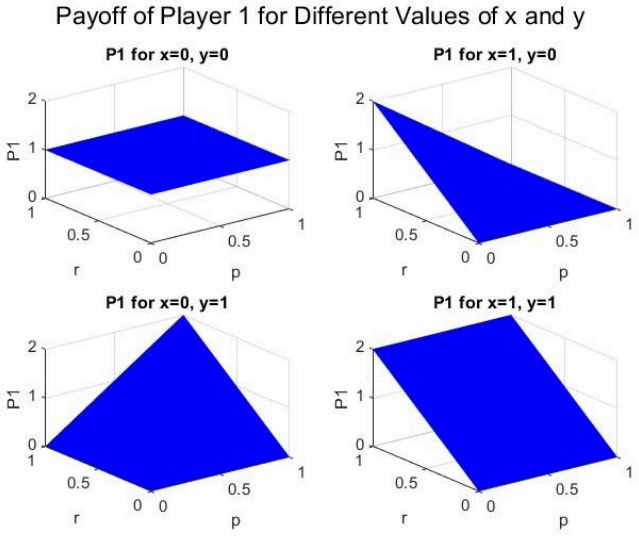

Figure 1

Based on the calculation of the payoffs, we can proceed to look at which strategies can be directly and indirectly invaded[3].

We can start by observing that when *x* and *y* are equal to 0, the payoff for both player 1 and player 2 doesn't depend on the return rate r. As a consequence, the strategy of player 2 is free to change to any amount of return; as long as all the trustors avoid sending in both cases, all those strategies are neutral to each other. If *r* is free to float, it is not unlikely that it will end up surpassing the th threshold or the value of 0.5, both significant for the choices of strategy in player 1 as we will see. Considering the strategies of the trustee, switching from *x=0* to *x=1* yields a higher payoff only when $r > 0.5$, is neutral when $r = 0.5$, and is not convenient when $r < 0.5$.

Not sending is robust against indirect invasion for all values of r under 0.5.

In the case of observers, having a threshold higher than the return rate is convenient for all $r < 0.5$, and it is indifferent which value higher than r it assumes. Every value tr higher than r is robust against indirect invasions when $r < 0.5$.

For values of *r* higher than 0.5, the sending strategy *x=1* is robust against indirect invasions. The strategy *tr* is robust against indirect invasion only at the value of 0.5, given the fact that it is a dominant strategy for $r > 0.5$.

Describing the scenario for player 2 is more complex, for that reason, we reduce the space of strategies to only 2 possible ones: $r = 0$ or $r = tr$. We are allowed to that simplification because every value of r between 0 and tr is dominated by $r = 0$ and every value higher than tr is dominated by $r = tr$. Addressing the first case when player 1 never sends when not observing, the payoff for playing *r=0* is 0 while the one for playing *r=tr* is *2(p)(1-r)*, which we can see is higher for any value of *tr* different than 1. In the case where *x=1*, meaning that the non-observing trustor always sends, the payoff for playing r=0 is 2(1-p) while the one for playing r=tr is *2(1-p)(1-r)+2(p)(1-r)* or *2(1-r)*, we can conclude that player 2 is better off by playing *r=0* only if $2(1 - p) > 2(1 - r)$ or rewriting it if $r > p$ or better if $tr > p$. Vice versa

[3] Van Veelen, M. (2012). Robustness against indirect invasions. *Games and Economic Behavior*, *74*(1), 382-393.

for $tr < p$ *r=tr* is a dominating strategy and robust against indirect invasion. It also follows that the strategies are equivalent when $tr = p$.

2.3 Trust Game in the lab

A comprehensive meta-analysis shows how the trust game almost always results in a significant rate of cooperation, even in its more basic layout[4]. It is a substantial difference from the theory that places the Nash Equilibrium in non-sending and the Subgame Perfect Equilibrium in non-sending and returning 0.

This level of cooperation rate is a raw measure that can be interpreted in different levels, the first one that comes to mind, directly from the name of the game, is the level of trust. This is subject to debate[5], here we will refer to it as a measure of cooperation to not bring any confusion. However it is worth examining whether we can talk about different concepts, by drawing from the trust game scenario. For example, there is consistent research that shows how the trust game also involves risk preferences.[6]

3. Methodology

An agent-based simulation is used to produce the results that are being analyzed. Here we will explain in detail the role of parameters and variables of interest, and why we choose them. An explanation of the process of playing out the game and calculating the payoffs will follow. Then we will explain some choices that have been made during the creation of the simulation, and the section will conclude with the hypothesis on the results.

3.1 The Model

3.1.1 Parameters

[4] Johnson, N. D., & Mislin, A. A. (2011). Trust games: A meta-analysis. *Journal of economic psychology*, *32*(5), 865-889.
[5] Brülhart, M., & Usunier, J. C. (2012). Does the trust game measure trust?. *Economics Letters*, *115*(1), 20-23.
[6] Chetty, R., Hofmeyr, A., Kincaid, H., & Monroe, B. (2021). The trust game does not (only) measure trust: The risk-trust confound revisited. *Journal of Behavioral and Experimental Economics*, *90*, 101520.

The first step is to set all the parameters, this step allows us to play with them in the future and to obtain simulations with different variables. For this analysis, we will fix four of them: number of agents, mutation probability, scale of mutation, and intensity of selection leaving us to play with observation probability and time span. The former is the main focus of the research and the latter is needed to show both the long-term effects and the mechanisms behind them.

In the simulation, an endowment of size 1 has been used and the first player was presented only with the choice of sending the whole amount or not sending any. This simplification is harmless given the fact that, if a player benefits from sending a positive amount, they would also benefit from sending the whole amount. Even if it does not invalidate the model, this choice can be an object of discussion and it will be addressed in the limitations section.

In the vast majority of lab experiments, the multiplier of choice is 3, which means that for every amount that gets sent by the trustor, the trustee will receive a triple amount. Here it has been decided to reduce this number to two, with the mere goal of making 0.5 a meaningful threshold instead of numbers like ⅓, which would have resulted in harder to visualize. This change consequently reduces even more the chances of cooperation in the base model and makes it harder for new mechanisms like commitment to have an impact. This doesn't invalidate the results of the analysis but instead demonstrates that they are obtainable even in stricter conditions than the one in the lab.

The time span determines how many iterations happen in the simulation, the values used are $2\text{^}11 * 10\text{^}4$ and $2\text{^}11 * 10\text{^}2$, and the variable $t *$ represents the value already multiplied by 2^11 to make it easier to write repeatedly. Longer amounts of time are used when talking about average levels over the whole simulation, to get results as accurate as possible. Shorter values of *t* are used when analyzing graphically specific sections of the simulation to achieve more visual clarity.

The parameter we are most interested in is of course observability. It determines the probability for the trustor to observe the return rate of the trustee before choosing to send or not the money. We will refer to non-observing trustors as dumb trustors and to observing trustors as smart trustors.

3.1.2 Variables

The variables we are looking at are the main strategies of the players and a fourth indirect outcome. The first one is the sending rate when not observing, it can only assume the values of 0 or 1, and it describes the simple strategy of the non-observing trustor. The second it's the return amount; it indicates the fraction of the pie that, when interpreting the role of trustees, the players will send back. It is a continuous variable and in this case, it is allowed to send back any amount between 0 and 2 in intervals of 0.0002. For clarity, we will refer to the value as a fraction of the amount. The trustee can't differentiate between the trustors, once chosen the strategy he will play it against all the other players at that time *t*. The third variable is also a strategy of the trustor and comes into play only in the case of observation. It is a value that works as a threshold, if the return rate of the trustee is below it, there is no sending, if it is above he sends the money. The existence of this threshold also calls for a fourth value that records the amount of sending that happens with observation.

There are also some values, which are not independent variables in the model but only outcomes obtained following the behavior of the variables and parameters just listed. One of them is the total sending rate, a mere weighted average of the sending rate of observers and non-observers.

3.1.3 The Simulation

After deciding all the settings, a simulation is run, where all the players face each other, once playing the role of the trustor and once as the trustee. Subsequently, all the payoffs are averaged and ready to be used in the selection process which will change the population structure. This process is iterated *t* times, and the snapshots are taken to be analyzed in the form of graphs, heatmaps, and averages.

3.2 Limitations

3.2.1 Snapshots

To measure all the parameters that we see in the output a snapshot method is used: every 10^4 times, the values of the variables of interest are recorded and stored into

the variables called snapshots. This method needs a lighter code, compared to the recording of all the values, that allows running the code for longer time spans like (2^11)*(10^4), which would otherwise require excessive computational power. This choice will also not harm the measure, given the fact that we are working mainly with average values, it also helps to eliminate the noise during the creation of graphs and charts.

3.2.2 Local Mutations

In the evolutionary process, we decided to use local mutations instead of global mutations to reduce the mutation bias[7]. For what concerns the sending rate, we cannot talk about global or local mutations, the variable can only assume the values of 0 and 1, therefore every time it changes it is a change of size 1. It is possible that the higher volatility of the sending rate, compared to the lower one of the return rate, influences the results of the simulation. For example, it is possible that the no-observation sending rate has a faster response to changes in payoffs and it adapts way faster than the return rate to the new equilibrium. A way to check this would be to run a different simulation with a continuous variable instead.

3.2.3 Random Initialization

The process continues by having a random initialization of the population, meaning that all the strategies at time 0 are randomly generated. This can cause the emergence of differences in the simulations that are only caused by chance. An example follows.

We can see that when the simulation starts with a threshold value (yellow) way higher than the return amount (orange), the number of senders when observing (violet) stays low for a long time (*Figure 2*). Instead, when the threshold is met from the beginning,

[7] Akdeniz, A., & van Veelen, M. (2023). Evolution and the ultimatum game. *Games and Economic Behavior*, *142*, 570-612.

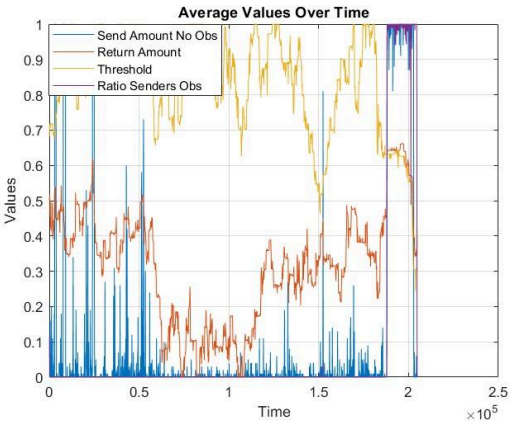

Figure 2

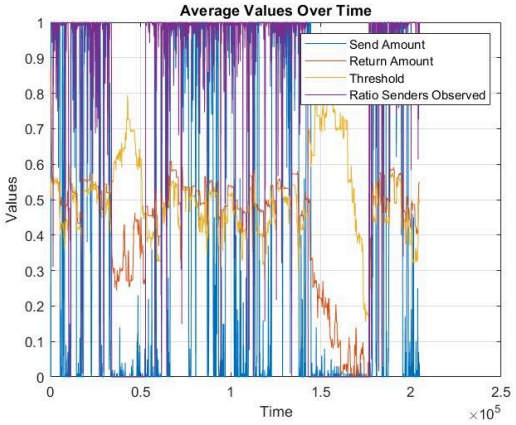

Figure 3

the variable in violet grows from the beginning, starting a sort of cycle (*Figure 3*). All the parameters in the two figures are identical except for the seeding, it is 40 in the first picture and 30 in the second. We will dive more deeply into the dynamics of the cycles in the results section.

We can also see that this problem gets progressively eliminated by extending the running time of the simulation; we will observe in the next section that at 10^4 the influence of the first initialization on the final results is negligible.

To ensure the reproducibility of the results, the seeds used for all the data here shown are mentioned in the document and in the document containing the code.

3.3 Hypothesis

3.3.1 No Observability

A special case is *p = 0.0*, where the model acts like any other trust game simulation with no observation happening. Applying backward induction we can see that, without observation, the payoffs would be *(1; 0)* in case of the trustor not sending, where the first number represents the payoff of player one and the second the one of player two. In the case of player one sending, the payoff would be *(2r; 2-r)* where *r* would represent the return rate. We can clearly see that the best move for player two is always to not send back anything, or more precisely *r = 0*. Any positive value of r is strictly dominated by *r = 0* and, when faced with enough selection pressure, it is safe to assume that over time the value would evolve towards 0. As a consequence of the

return rate moving to low values, sending money also becomes a losing strategy, in fact, it happens whenever $2r < 1$, or $r < 0.5$. That means that the strategy of player one will evolve to not sending. Because we assume rational players and the shared knowledge that all players are rational, in standard game theory we know that player one expects from the trustee $r = 0$, and he will not send the money, hence the subgame perfect equilibrium is (not send, 0).

3.3.2 Full Observability

Another peculiar occurrence is encountered when the observability parameter is 1, in this case, the game is almost indistinguishable from an ultimatum game, where the trustee acts like a proposer. In the ultimatum game, the first mover can decide how to split a certain amount. Subsequently, the second player is confronted with a binary choice: to accept the split or to refuse, in case of a refusal both players get nothing. The payoffs can be summarized as $(1-x; x)$ in case of an accepted offer and $(0; 0)$ in case of a refusal. The variable x represents how much the first player offered to the second player. The subgame perfect equilibrium is for the proposer to choose a $x = 0$ if we assume that the second player would accept when indifferent, or the smallest x possible otherwise. If we look closely at the mechanics of the game, Proposing a split is equal in outcome to choosing how much to send back, and the trustor acts like the second player of the ultimatum game, where he decides to send and benefit from the multiplier, or not send. The main difference is that, in the non-cooperative case of the ultimatum game, the outcome is (0; 0), while in the trust game it is (1; 0). It means that while in the ultimatum game, it is always either neutral or beneficial to accept the split, in the trust game the proposer will not be damaged by sending the money only if the return rate is 0.5 or higher. The subgame perfect equilibrium for the trust game is: for the trustee to return 0.5 or the smallest rate higher than 0.5 and for the trustor to send the money. Just as in the ultimatum game, the total earnings are one unit higher in case of cooperation, and the player that benefits the most from it is the one that can decide the split, or how much to send back in case of the trustee.

3.3.3 Partial Observability

For the values of p in between 0 and 1, we expect a mixed behavior. A small amount of observability could raise the cooperation rate, but it is not expected to be prominent before the observability gets to higher values. If the observability is

enough to cause the return amount to rise, then we will expect the sending with and without observability to follow. In general, in this simulation we expect the findings to follow the dynamics analyzed in the theory section.

4. Results

The simulations show trends for each of the variables observed.

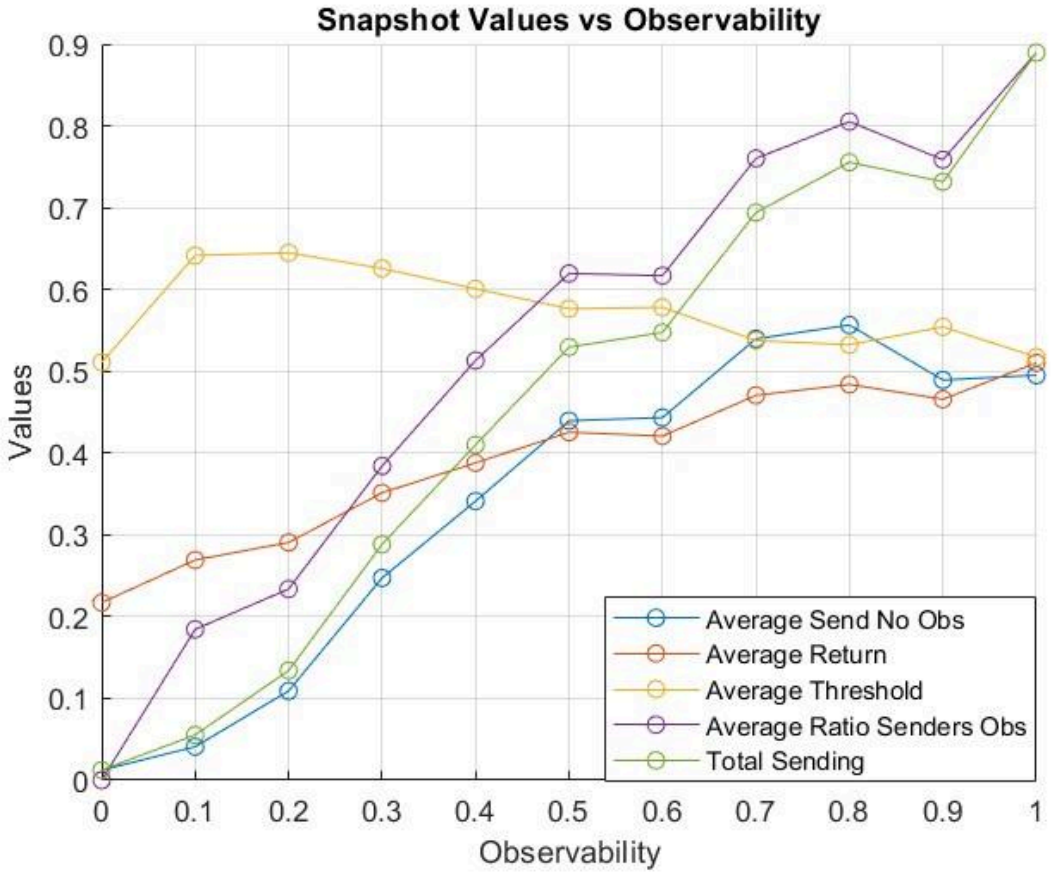

Figure 4

Observability	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
Average Send No Obs	0,012354	0,041011	0,10915	0,24729	0,34119	0,43943	0,44339	0,53967	0,55651	0,48931	0,49518
Average Return	0,21722	0,26919	0,29073	0,35138	0,38799	0,42543	0,42075	0,47063	0,4839	0,46579	0,51029
Average Threshold	0,51082	0,64145	0,6448	0,62574	0,60068	0,57645	0,57813	0,53757	0,53237	0,5542	0,51734
Average Ratio Senders Obs	0	0,18445	0,23378	0,38414	0,51316	0,61952	0,61685	0,76019	0,80525	0,75859	0,8895
Total sending	0,012354	0,055355	0,134076	0,288345	0,409978	0,529475	0,547466	0,694034	0,755502	0,731662	0,8895

Figure 5

In the graph of *Figure 4*, we see the value of our main variables of interest on the Y-axis and the level of observability on the X-axis. The simulation is run for a time of *t* = 10^4* and shows the average value of the variables over the whole time.

4.1 No Observability

First of all, it is good practice to focus on the extreme cases, the ones with *p = 0.0* and *p = 1*, that we already identified as peculiar.

For probability of observing is equal to 0.0 we find ourselves in the basic trust game, where, by looking at the variables in *Figure 5*, we can see it follows the theoretical predictions.

More precisely we observe an almost zero rate of senders, which is caused by a low return rate, once again, whenever the return rate is lower than 0.5, there is selection against sending. We can also observe that the average return is low, but not as low as the sending rate. This happens because there is selection against positive return rates only when there is a significant amount of sending, otherwise, the return rate is free to go up and down without much selection, almost following a random walk. That is a consequence of not having to play, and getting the same payoff as every other trustee.

It can be helpful to look at the data visualization for a simulation with *t = (2^11)*(10^2)* and *p = 0*, where the simulation is run for a shorter amount of time to allow us to better see the dynamics of the variables in detail. We see from *Figure 6* that there are 3 instances where the sending rate goes to extremely high values. From the closeup in *Figure 7*,

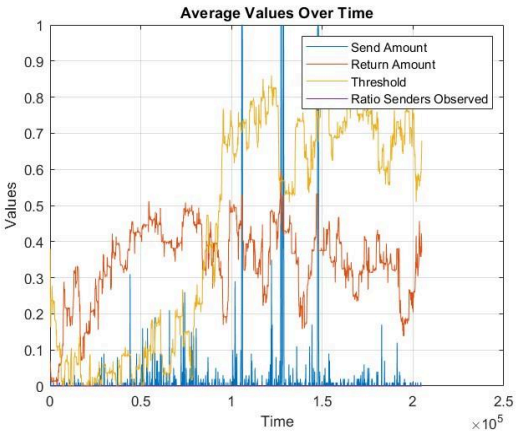

Figure 6

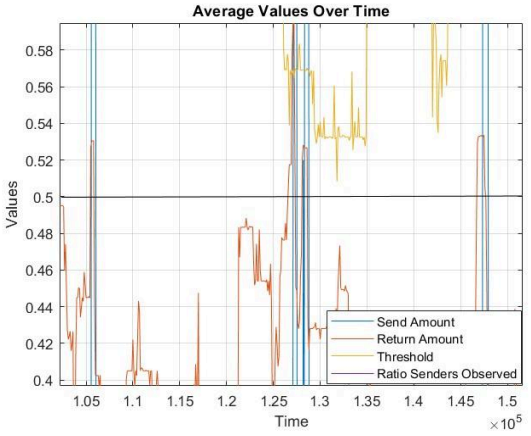

Figure 7

we can see that it always happens as a consequence of the return rate becoming higher than 0.5, making the payoff of *2r* higher than the payoff of not sending which is 1. It is also evident from the image that, once everyone starts sending, the penalty for having a positive return rate gets immediately higher, pushing the return rate to

lower values. This triggers the opposite mechanism, where the return rate gets again under 0.5, and sending is no longer a viable strategy for the trustors, who drop once again to a very low sending rate. This is why, at low levels of observability, the send amount remains extremely low and the return amount never passes the 0.5 level for extended periods. It doesn't go unnoticed that the return amount has a much more volatile behavior than the return amount, this is, once again, a consequence of the binary nature of the choice of the trustor. It affects the dynamics because every mutation carries a much higher change in value than the ones that affect the return rate and the observer threshold, those two being subject to local mutations.

Not very insightful is the average threshold that follows a random walk because it does not face any selection, given that the strategy never comes into play, but the value is coherent with the theory prediction. The ratio of senders, when observed, is clearly not measurable when there is no observation, therefore it was decided to give it the value of 0. The total sending corresponds exactly to the sending when not observed.

4.2 Full Observability

We will still look at *Figure 3*, this time when $p = 1$. The average number of senders with no observation is not of importance given that there is no game played without observation, hence, it follows a random walk, it is still a good indicator that the simulation is coherent with the theory. At full observability, the return rate is close to 0.5, which is the value that maximizes the gain in a situation of perfect rationality, so it is no surprise that it gets selected by evolution.

We can see from *Figure 8* how when the return level is higher than 0.5 the threshold is free to move between any value smaller than the return level, and when it surpasses it, it is selected again to lower values.

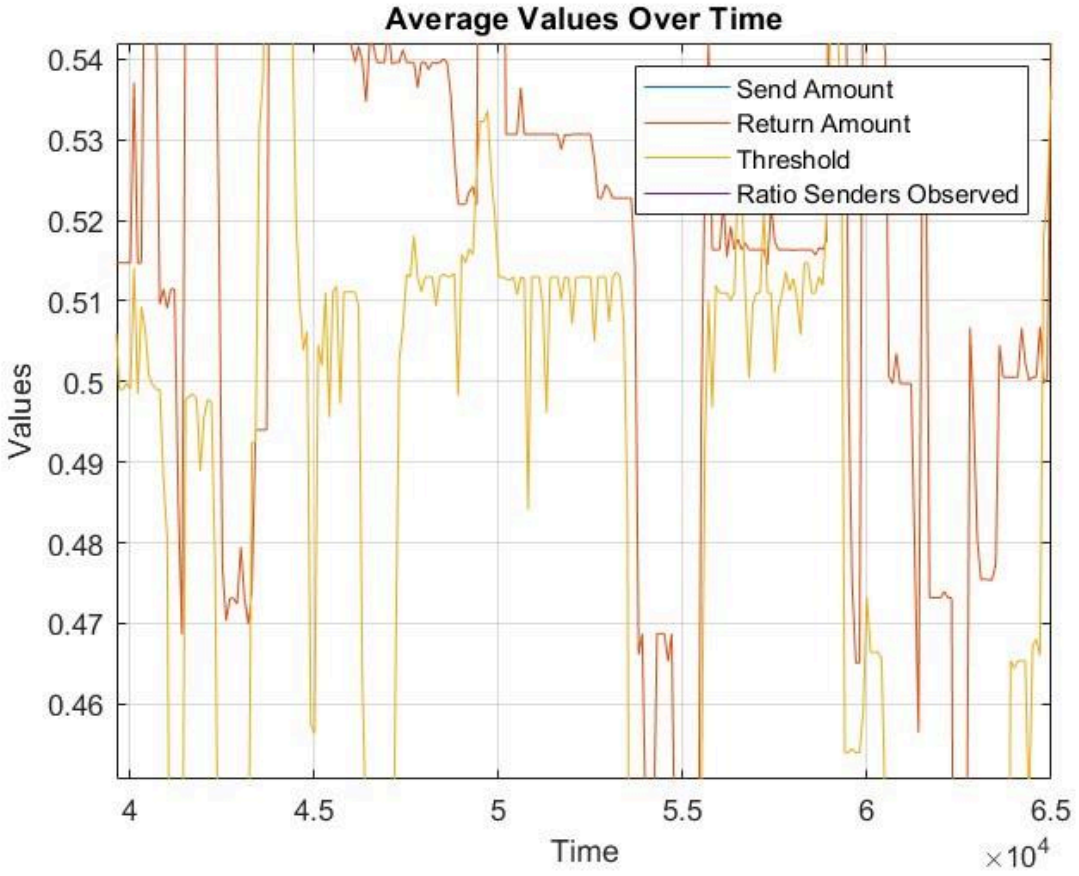

Figure 8 (Send Amount and Ratio of senders removed for visual clarity)

That happens because, as long as the threshold is lower than the return rate, it doesn't matter the value, the trustor will always send the money. On the other hand, the trustee has no advantage in holding a return rate higher than the threshold, therefore it will be selected to be as close to the threshold as possible. That mechanism causes the behavior shown in the image, where the threshold reaches lower values as an effect of neutral mutations, as a consequence the return rate adapts to lower values, almost matching the threshold. Once the return rate gets below 0.5 it is not convenient anymore for the trustee to send the money, as a consequence the threshold starts immediately to rise in value. The return rate, being better off by matching the threshold, starts consequently to rise, getting back to a more stable situation in the area below the 0.5 line. Now a question naturally emerges, why, if the threshold is selected so strongly to be just under the return rate, does its value end up to be slightly higher on average as we can see from *Figure 4*? The reason can be found by looking at the bigger picture: while it is true that the

return rate is most of the time above the value of the threshold, it is also true that there are instances where the two variables take substantially different journeys.

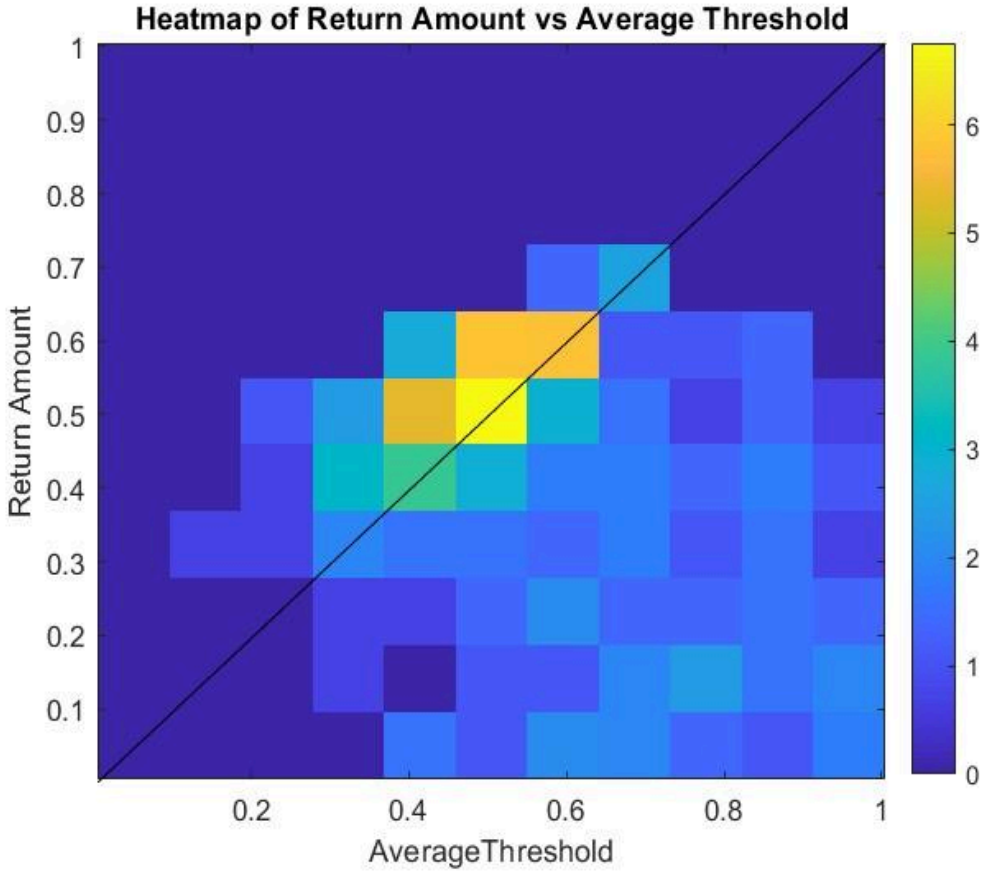

Figure 9

In *Figure 9* we have a heatmap, which is a graphic visualization of frequencies of the values falling in certain intervals for both variables. Following the color scale on the right we see that squares with warmer colors mean a higher number or pairing between the value of the threshold and the return amount. From it we can observe that indeed most of the points lay at the left of the line in the picture, meaning that in most cases the return amount is higher than the threshold. But it is also true that most of them carry only a small difference between the two values, while, in the less frequent cases where the threshold is higher, the difference in value is way higher. This causes the two averages in the end to be almost identical, with an even higher value of the threshold.

4.3 Partial Observability

Now we can focus on what happens at the intermediate levels of observation.
The average amount sent when no observation happens is shown to grow as a consequence of a higher observability. The payoff for the trustor is 1 if he doesn't send, and it is 2r when he sends. The complication emerges when we have two different scenarios, one with observability and one without it. The consequence is that very different sending rates can emerge between the players that observe and the ones that do not. As we see from *Figure 3*, there is an increasing gap in the sending rate between trustors who observe and trustors who don't, the reason why this happens is that the two mechanisms underneath the behavior of the variables are fundamentally different.

The no obs sent amount, as established before, depends only on the average return rate, whether it is higher or lower than 0.5. We need to focus on the term average, given that the trustor plays the game with every other player, his payoff will necessarily depend on the average level of return. If the average payoff is $\frac{1}{100}\sum_{i=1}^{100} 2 * r$ we can easily rewrite it as $2\sum_{i=1}^{100} r$ which is exactly $2 * \bar{r}$, two times the average of r. The same cannot be said for the sending rate with observation, given that a player can now distinguish between return rates. Let's take an example with the ideal threshold of 0.5 for the observing trustor, let's assume also a very unlikely but explicative situation where the return rates are 0.2 in half of the population and 0.6 in the other half. A dumb trustor will not benefit from sending in this scenario, given that his payoff would be $\frac{1}{100}2(50 * 0.2)(50 * 0.6) = 0.8$ which is lower than the 1 he would get by not sending. A smart trustor with a 0.5 threshold will instead only send money to the trustee with a higher return rate and get a total payoff of $\frac{1}{100}(50 * 1) + 2\frac{1}{100}(50 * 0.6) = 1.1$ which is a higher payoff than 1.

To explain the difference in value we need to go back to *Figure 2* and *Figure 3* where we have graphs with *t** equal to 10^2 and *p = 0.5*, with only a difference in random seeding. That difference causes a relative abundance of cooperative cycles in the second image. We can define a cooperative cycle as a series of times where the

average level of sending is extremely high, and the return rate is for most of the time above 0.5 and the threshold. This is not a precise definition and leaves room for interpretation, however, it can be useful to describe behaviors like the one we observe after *t = 1.5*10^5* in *Figure 10*. A negative cycle will then be an extended time where all those values lean toward zero, like what happens before *t = 1.5*10^5* in *Figure 10*.

It is no surprise that both the level of sending when observed and when not observed are higher in the second case, we can easily see it even from only looking at the picture. What is less obvious is that, in the first case, the sending rate is higher when not observed: 0.097549 no obs, 0.081148 with obs. It is the opposite case in the second scenario, where we have: 0.4193 no obs, 0.70285 with obs. This is a clue leading to the idea that in the absence of positive cycles, when the return is low, the sending rate of non-observers tends to have a more volatile behavior, while still remaining very low, compared to a return rate with observations that is almost always zero. On the other hand, when the return rate gets high enough, both sending rates, with their respective mechanisms, rise to very high levels. The difference in this case is that the observers have on average an even higher sending rate in positive cycles.

Here the graphs of *t* = 10^2 p= 0.2* and with p = 0.6:

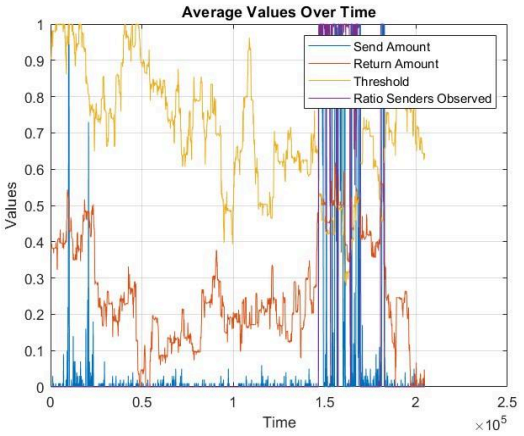

Figure 10

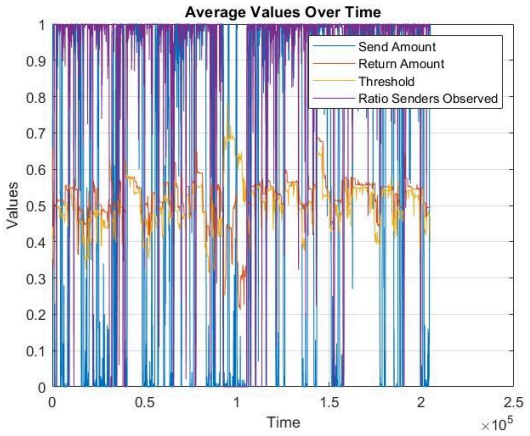

Figure 11

We can also see from *Figure 10* and *Figure 11*, respectively with *p = 0.2* and *p = 0.6*, that a higher observation rate also causes a higher amount of positive cycles, for both the non-observing trustors and the observing ones, even if in different proportions. It is important to remember that the levels in *Figure 4* are averages, that the story they tell, doesn't highlight the complex series of mutations that are shown by the other data, and that even when the cooperation rate is higher, there could be times where it drops significatively, before rising again.

The average return (orange) also grows with higher *p*. When *p = 0* the positive values are only due to the fact that, with a low level of the amount sent, the pressure that would bring it back to lower values is reduced. With the progressive growth of *p*, it gets more convenient to have a higher return, up to the value of 0.5, given the fact that it is never convenient to send back more than 0.5. In scenarios where we deal with both observers and non-observers, the payoff gets tricky. Every amount returned to non-observers is a loss, because they would have sent either way, on the other hand returning the amount for observers can trigger their threshold and make them send, so that is a way to formalize the payoff: $(1 - p) * 2(1 - r) + (p)(n)(1 - r)$ where n is the number of times the threshold is lower than the return rate, or more simply the ratio of senders in between observers. The difficult part is that the variable n also depends on the return rate, and for a certain level of observation, a higher n gets convenient even at the cost of a higher r. The convenience of a satisfactory return rate gets higher, and it is shown by the progressive rise in values of the variable at higher p.

The average threshold required by trustors to send their money starts as a random walk, as it comes into play only when p is positive. According to the theory, the best level of this value is 0.5 or right above. We can see in the graph that with a higher p, the pressure to get closer to the optimal value is higher and the actual value gets closer.

The last variable, the total sending, is no other than a weighted average of the sending of observers and non-observers, it is the variable that I am going to refer to as cooperation rate and which a lot of research addresses as trust level. The fact that it grows means that the strategy of sending the money has become advantageous (or less disadvantageous) for the trustors, and that can only happen when the return rate is growing as well, and getting close to 0.5. Of course one of the

main drivers of its growth is the fact that with higher observation rates we substitute dumb trustors with smart trustors, given that we weigh the value with the ratio of population that they cover, at p = 0.8 the non-observers will account only for the 20% of the measure.

Now a comparison between observation levels of 0.2 and 0.8 observing some heatmaps.

Heatmap for p = 0.2

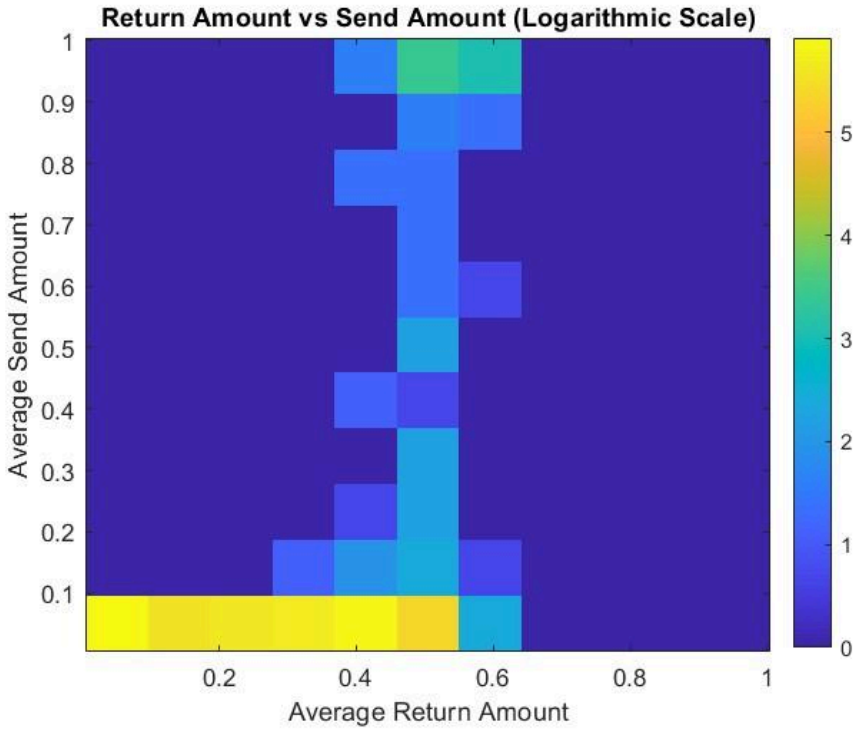

Figure 12

It is observable that the return lies in the interval from 0 to 0.5, it grows as a consequence of the freedom given by a low amount of senders and it gets pushed back by the fact that returning is not a viable strategy for such a low amount of observers (20%). Returning is beneficial only when it is observed and can induce trustors to send. The findings in the heatmap corroborate the dynamic explained in the previous analysis.

Heatmap for p = 0.8

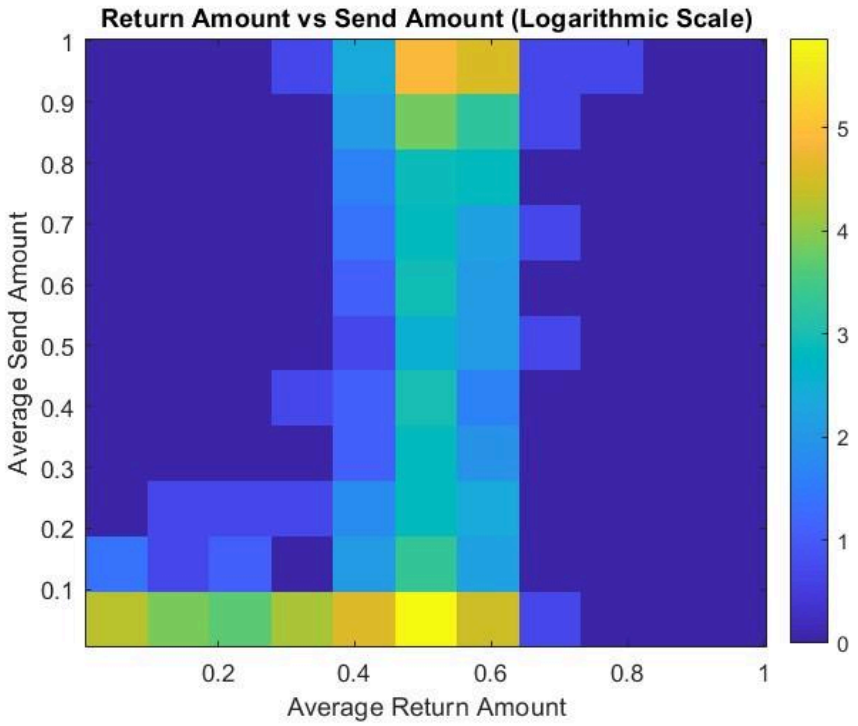

Figure 13

Here it is clear that the return has a positive shift and tends to cluster around 0.5, which is a consequence of more observers and the fact that it is now beneficial to return higher amounts. We can see that also the sent amount has a positive shift, even if it maintains the tendency to fixate at 0 and 1, due to the binary nature of the variable.

The most interesting result from both maps is that higher values of sending are correlated with higher values of return and they get to higher values only when the loss from sending is lower and there is less pressure for them to go back to zero. To look into the dynamic of this relationship we can observe again a graph of a simulation run for a lower amount of time (10^2), keeping $p = 0.8$.

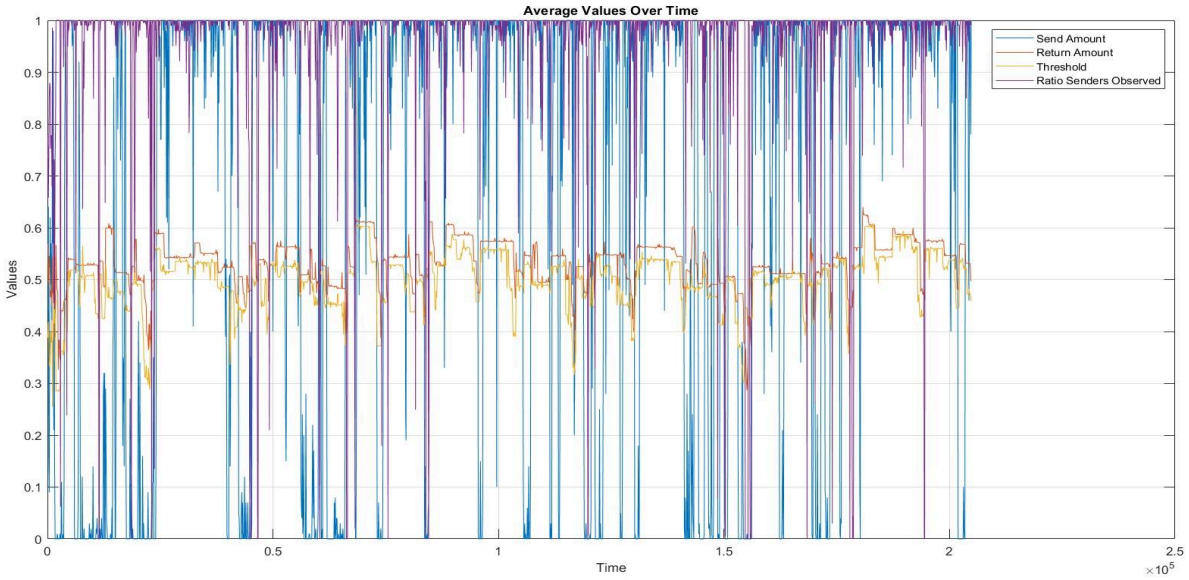

Figure 14

From this image, we can see that drops in send amount (blue) are a direct consequence of return rates (orange) dropping lower.

Now we can focus on a different couple of heatmaps, again for 0.2 and 0.8 observation rates.

Heatmap for p = 0.2

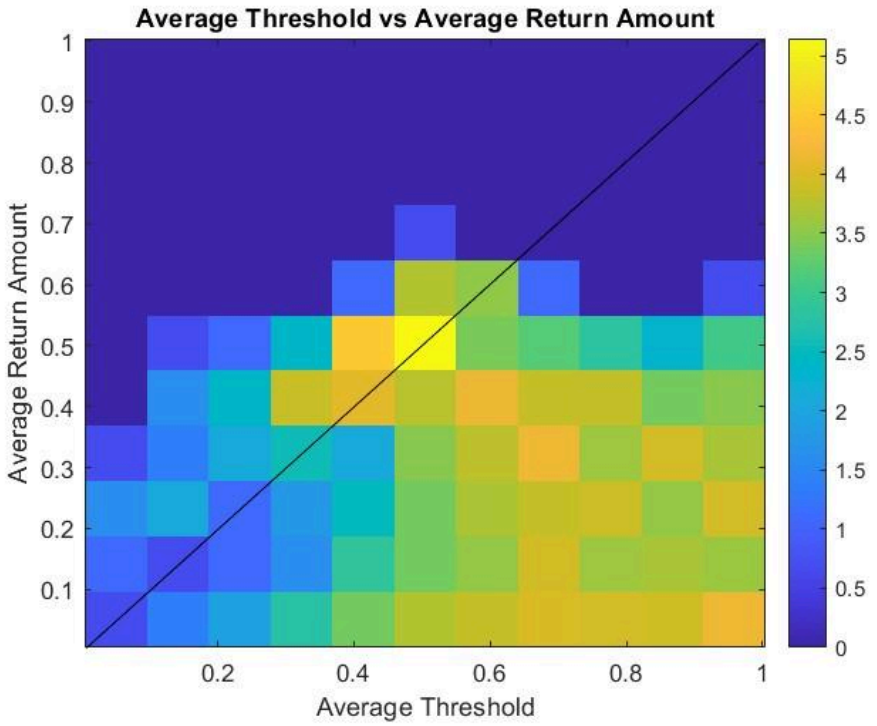

Figure 15

As we see also from Figure 2, the average threshold is still higher than the average return, as we can see that most points fall right of the line in the middle. We can also observe that when the two values meet it is around 0.5.

Heatmap for p = 0.8

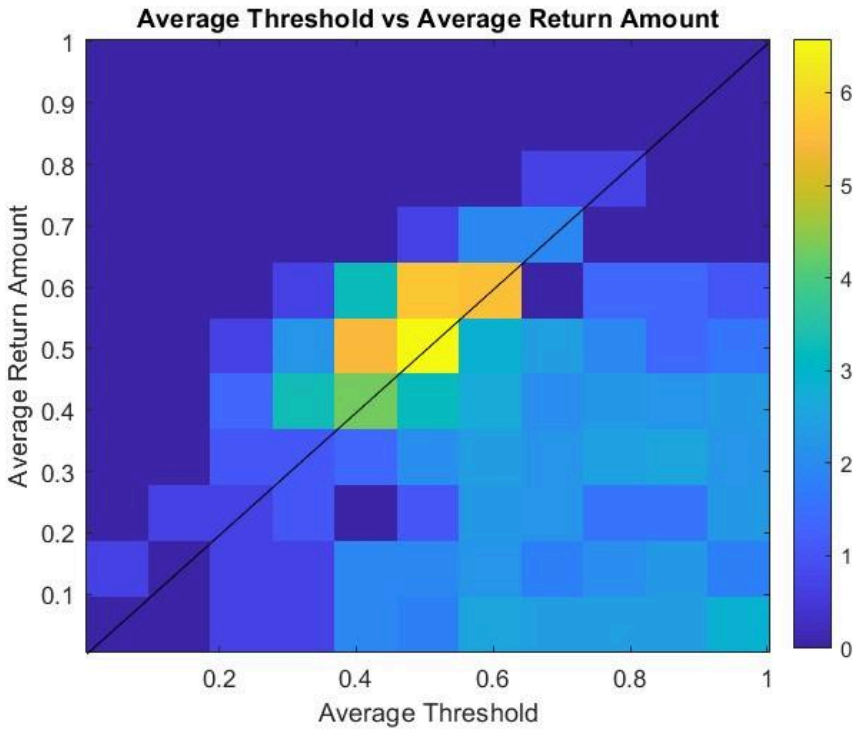

Figure 16

Instead, when we move the observability from 0.2 to 0.8, we can observe that most points fall at the left of the line, meaning that the average return is higher than the threshold, and the trustors that observe tend to send the money. We can still see that the points cluster around the optimal value of the two strategies, which lies around 0.5.

5. Conclusion and Further Research

5.1 Main Findings

We can conclude that observation has been revealed as a precious instrument for improving cooperative behavior in the model. Observation introduces the possibility for the trustor to know the return rate of the trustee and base its strategy on that information. As a consequence, having a positive return rate becomes a viable strategy because it causes the trustors to send. We have referred to those occurrences as positive cycles and they keep happening multiple times during the

long periods of the simulation. The frequency and length of those cycles increase with higher values of observability, as we can see from the closeups of the simulation in *Figures 9 and 10* and the aggregate data in *Figure 3*. The results match the theoretical predictions and confirm the expected evolution of the players' strategies. Further analysis helps to understand the relationships between single variables in the model with the help of heatmaps. This second step is fundamental to understanding the causality relation of the change in values inside the cycles and to have a more accurate picture compared to the one with aggregate values.

It is important to obtain this result to strengthen the findings of the theory with the ones of the simulations, to add one possible explanation to the observed behavior in the trust game, and to expand our knowledge about mechanisms that facilitate pro-social behavior in general.

5.2 Limitations

The main problem encountered during the designing of the simulation is the amount of data generated. After every period, the variables in the code store a new set of 3 strategies for each of the 100 agents. It shouldn't be a surprise that, given the long time spans we decided to deal with, the number grows enough to be a problem for both computation and data visualization. To solve this issue we used the snapshot method, which is explained more extensively in the methods section. Further analysis could drift away from the snapshot method and substitute it with something more preserving of the data as a whole, possibly with the help of a more performing machine and tools suitable for large amounts of data.

Another criticism of the research, this time on the results themselves, could be the idea that the paper doesn't demonstrate that the prosocial behavior in the trust game has evolved from this very own game, the hypothesis on which the simulation seems to be based. The answer to this is that the aim of the paper has never been to explain definitively the origin of such a behavior, but the one of exploring if observation could cause it. To understand in the first place if this behavior in humans has originated because of the trust game, the kind of analysis needed is anthropological and not game-theoretical.

5.3 Further Research

5.3.1 Commitment

The fact that observability helps to raise cooperative behavior opens the door to new mechanisms that make use of observations to evolve cooperative strategies, one of which is commitment. One of the advantages of this explanation is that it doesn't need population structure or repeated games to work[8], which can be a good fit for the trust game, as we showed with this research. Further analysis of how commitment can influence cooperative strategies and if it has a role in the evolution of pro-social behaviors is desirable and could lead to new unexplored areas of knowledge.

5.3.2 Reputation and Observation

A separate discussion is worth being faced by diving into repeated games. Hereby is not addressed any mechanism of reputation, which can also be an interesting way to model cooperative behavior. The main difference is that in the model with observation, there is no space for errors, if the observation happens it does so with 100% certainty and accuracy. In other models like reputation and face recognition, there is not such a binary situation and the signal can be unreliable.

An interpretation of p could be the quality of observation, we can think about a situation where the trustor can guess the return rate with probability p, and with probability $1-p$ he will be wrong. With this approach, we open to a more realistic interpretation, where for example a trustor can try to guess the reliability of a trustee by facial features, or any other signal. That is a limitation because this more realistic model is not incorporated into the research.

5.3.3 Non-binary Parameters

Another question that could be researched in the future is the one of how the simulation would unfold with non-binary parameters for the senders. In this model, the choice of a binary variable for the senders has both a computational advantage

[8] Akdeniz, A., & Van Veelen, M. (2021). The evolution of morality and the role of commitment. *Evolutionary Human Sciences*, *3*, e41.

and it was considered more realistic. Another researcher could find more accurate a depiction where the trustor can fraction the amount to send and develop a simulation that uses a continuous variable instead.

Bibliography

Akdeniz, A., & Van Veelen, M. (2021). The evolution of morality and the role of commitment. Evolutionary Human Sciences, 3, e41.

Akdeniz, A., & van Veelen, M. (2023). Evolution and the ultimatum game. Games and Economic Behavior, 142, 570-612.

Brülhart, M., & Usunier, J. C. (2012). Does the trust game measure trust?. Economics Letters, 115(1), 20-23.

Chetty, R., Hofmeyr, A., Kincaid, H., & Monroe, B. (2021). The trust game does not (only) measure trust: The risk-trust confound revisited. Journal of Behavioral and Experimental Economics, 90, 101520.

Imhof, L. A., & Nowak, M. A. (2006). Evolutionary game dynamics in a Wright-Fisher process. Journal of mathematical biology, 52, 667-681.

Johnson, N. D., & Mislin, A. A. (2011). Trust games: A meta-analysis. Journal of economic psychology, 32(5), 865-889.

Langergraber, K., Schubert, G., Rowney, C., Wrangham, R., Zommers, Z., & Vigilant, L. (2011). Genetic differentiation and the evolution of cooperation in chimpanzees and humans. Proceedings of the Royal Society B: Biological Sciences, 278(1717), 2546-2552.

Roberts, G. (2008). Evolution of direct and indirect reciprocity. Proceedings of the Royal Society B: Biological Sciences, 275(1631), 173-179.

Thielmann, I., Böhm, R., Ott, M., & Hilbig, B. E. (2021). Economic games: An introduction and guide for research. Collabra: Psychology, 7(1), 19004.

Van Veelen, M., García, J., Rand, D. G., & Nowak, M. A. (2012). Direct reciprocity in structured populations. Proceedings of the National Academy of Sciences, 109(25), 9929-9934.%Van Veelen, M. (2012). Robustness against indirect invasions. Games and Economic Behavior, 74(1), 382-393.

Appendix A

Here presented is the code used in all the simulations. The value of some of the parameters may vary depending on which simulation is taken into analysis. The code is obtained by performing modifications to the code used in Akdeniz, A., & van Veelen, M. (2023). Evolution and the ultimatum game. Games and Economic Behavior, 142, 570-612.

```
% Trust game simulations with local, independent mutations, and partial
observability
%% Population parameters
s = 100; % number of agents
w = 1; % intensity of selection
u = 0.001; % mutation probability
p = 0.5; % probability of observing other's behavior
scale = 5; % division scale of mutation
x = 1; % size of the pie to be split
t_max = (2^11)*(10^2); % maximum number of time periods
t_timesteps = 10^2; % determines how often a snapshot is taken
t_snapshots = t_max/t_timesteps; % number of times a snapshot will be taken
summary = zeros(18,1); % final average send amounts and return thresholds
average_send_over_time_not_observed = zeros(t_max,1);
average_send_over_time_observed = zeros(t_max,1);
average_threshold_over_time = zeros(t_max,1);
% New matrix to store the ratio of players sending money when there is
observability
ratio_senders_observed_snapshot = zeros(t_snapshots, 1);
% Stopping rule
epsilon = 0.005; % when to stop
m_final_send_not_observed = zeros(s,1);
m_final_send_observed = zeros(s,1);
m_final_threshold = zeros(s,1);
rng(30); % setting the seed
counter1 = 0; % number of times mutation occurred at reproduction
counter2 = 0; % number of times mutation occurred due to fixation
m_snapshot1 = zeros(s,t_snapshots); % matrix to record snapshots of send amounts
in time
m_snapshot2 = zeros(s,t_snapshots); % matrix to record snapshots of return
amounts in time
m_snapshot3 = zeros(s,t_snapshots); % matrix to record snapshots of thresholds
in time
stop = 0;
%% Random initialization of the population
m = rand(s,3); % random initial strategies: first is send amount, second is
return amount, third is threshold
m(:,1) = round(m(:,1)); % ensure initial send amounts are either 0 or 1
```

```
m_new = zeros(s,3);
m_total = m; % initialization of the total matrix
average_send_over_time_not_observed(1,1) = sum(m(:,1))/s;
average_send_over_time_observed(1,1) = sum(m(:,2))/s;
average_threshold_over_time(1,1) = sum(m(:,3))/s;
t = 2;
%% Game play
while t <= t_max % as long as maximum time step is not reached
   payoff = w*zeros(s,1); % payoff vector in actual payoffs
   probs = rand(s,s); % matrix of observation probabilities for each interaction
   num_senders_observed = 0; % counter for senders when observability is in
effect
   num_observations = 0; % counter for the number of observations
   for i = 1:s % game play for each player i
       for j = 1:s % for each opponent j
           if j ~= i
               % i as trustor
               if probs(i,j) > p % doesn't observe
                   sent_amount = m(i,1) * x;
               else % observes
                   num_observations = num_observations + 1; % count the number
of observations
                   if m(j,2) >= m(i,3) % send if trustee behavior is high enough
                       sent_amount = x;
                       num_senders_observed = num_senders_observed + 1; % count
the number of senders when observing
                   else
                       sent_amount = 0;
                   end
               end
               received_amount = 2 * sent_amount;
               returned_amount = m(j,2) * received_amount;
               payoff(i,1) = payoff(i,1) + w*(x - sent_amount +
returned_amount);
               payoff(j,1) = payoff(j,1) + w*(received_amount -
returned_amount);
           end
       end
   end

   payoff = payoff./(2*(s-1)); % averaging the payoffs over all interactions
   payoff_exp = exp(payoff); % exponential transformation of the payoff vector
   payoff_proportional = payoff_exp/sum(payoff_exp); % payoff vector with
proportional payoffs
   %%%% Wright-Fisher version
   r = rand(s,1); % random vector to determine who are reproducing
   mut = rand(s,3); % random vector to determine if a mutation occurs at spot k
   for k = 1:s % for each k, to determine who is reproducing to the location
m(k,:)
       index = 0; % to determine the location of the reproducing agent
       while r(k,1) > 0 % while loop to pick an agent with the given
```

```
probabilities
            index = index + 1;
            r(k,1) = r(k,1) - payoff_proportional(index);
        end

        if mut(k,1) < u % mutation occurs at send amounts w/o observing
            z1 = round(rand); % mutation results in 0 or 1
            counter1 = counter1 + 1; % number of times mutation occurred
            m_new(k,1) = z1;
            if mut(k,2) < u % mutation occurs at return amounts w/ observing
                z2 = (rand - 0.5)/(scale); % random number to determine the new
mutation
                counter1 = counter1 + 1; % number of times mutation occurred
                m_new(k,2) = m(index,2) + z2;
                if mut(k,3) < u % mutation occurs at thresholds
                    z3 = (rand - 0.5)/(scale); % random number to determine the
new mutation
                    counter1 = counter1 + 1; % number of times mutation occurred
                    m_new(k,3) = m(index,3) + z3;
                else % no mutation occurs at thresholds
                    m_new(k,3) = m(index,3);
                end
            else % no mutation at return amounts w/ observing
                m_new(k,2) = m(index,2);
                if mut(k,3) < u % mutation occurs at thresholds
                    z3 = (rand - 0.5)/(scale); % random number to determine the
new mutation
                    counter1 = counter1 + 1; % number of times mutation occurred
                    m_new(k,3) = m(index,3) + z3;
                else % no mutation occurs at thresholds
                    m_new(k,3) = m(index,3);
                end
            end
        else % no mutation occurs at send amounts w/o observing
            m_new(k,1) = m(index,1);
            if mut(k,2) < u % mutation occurs at return amounts w/ observing
                z2 = (rand - 0.5)/(scale); % random number to determine the new
mutation
                counter1 = counter1 + 1; % number of times mutation occurred
                m_new(k,2) = m(index,2) + z2;
                if mut(k,3) < u % mutation occurs at thresholds
                    z3 = (rand - 0.5)/(scale); % random number to determine the
new mutation
                    counter1 = counter1 + 1; % number of times mutation occurred
                    m_new(k,3) = m(index,3) + z3;
                else % no mutation occurs at thresholds
                    m_new(k,3) = m(index,3);
                end
            else % no mutation at return amounts w/ observing
                m_new(k,2) = m(index,2);
                if mut(k,3) < u % mutation occurs at thresholds
```

```
                    z3 = (rand - 0.5)/(scale); % random number to determine the
new mutation
                    counter1 = counter1 + 1; % number of times mutation occurred
                    m_new(k,3) = m(index,3) + z3;
                else % no mutation occurs at thresholds
                    m_new(k,3) = m(index,3);
                end
            end
        end

        if m_new(k,1) > 1
            m_new(k,1) = 1;
        elseif m_new(k,1) < 0
            m_new(k,1) = 0;
        end
        if m_new(k,2) > 1
            m_new(k,2) = 1;
        elseif m_new(k,2) < 0
            m_new(k,2) = 0;
        end
        if m_new(k,3) > 1
            m_new(k,3) = 1;
        elseif m_new(k,3) < 0
            m_new(k,3) = 0;
        end
    end
    m(:,:) = m_new(:,:); % updating to the old population state to the new
population
    if rem(t,t_timesteps) == 0
        m_snapshot1(:,t/t_timesteps) = m(:,1); % recording send amounts present
at point t
        m_snapshot2(:,t/t_timesteps) = m(:,2); % recording return amounts present
at point t
        m_snapshot3(:,t/t_timesteps) = m(:,3); % recording thresholds present at
point t
        if num_observations > 0
            ratio_senders_observed_snapshot(t/t_timesteps) = num_senders_observed
/ num_observations; % recording the ratio of senders with observability at point
t
        else
            ratio_senders_observed_snapshot(t/t_timesteps) = 0; % no observations
occurred
        end
    end

    average_send_over_time_not_observed(t,1) = sum(m(:,1))/s;
    average_send_over_time_observed(t,1) = sum(m(:,2))/s;
    average_threshold_over_time(t,1) = sum(m(:,3))/s;
    m_total = m_total + m; % adding the current state of the population
    t_reached = t;
```

```
   for j = 4:11
       if t == t_timesteps*(2^j)
           if abs(sum(average_send_over_time_not_observed(1:(t/2), 1)) -
sum(average_send_over_time_not_observed((t/2+1):t, 1)))/(t/2) <= epsilon
               if abs(sum(average_threshold_over_time(1:(t/2), 1)) -
sum(average_threshold_over_time((t/2+1):t, 1)))/(t/2) <= epsilon
                   if abs(sum(average_send_over_time_observed(1:(t/2), 1)) -
sum(average_send_over_time_observed((t/2+1):t, 1)))/(t/2) <= epsilon
                       stop = j;
                   end
               end
           end
       end
   end
   if max(m(:,1)) == min(m(:,1)) && max(m(:,2)) == min(m(:,2)) && max(m(:,3)) ==
min(m(:,3))
       number_of_reproductions = geornd(u); % drawing a random number of
reproductions until next mutation
       number_of_time_periods = fix(number_of_reproductions/(3*s)); % number of
generations passed until next mutation
       mut_location = rem(number_of_reproductions,3*s); % location of the
mutation
       if t + number_of_time_periods <= t_max
           for b = 1:number_of_time_periods
               average_send_over_time_not_observed(t_reached+b,1) =
sum(m(:,1))/s;
               average_send_over_time_observed(t_reached+b,1) = sum(m(:,2))/s;
               average_threshold_over_time(t_reached+b,1) = sum(m(:,3))/s;
               if rem(t_reached+b,t_timesteps) == 0
                   m_snapshot1(:,(t_reached+b)/t_timesteps) = m(:,1); %
recording send amounts present at point t
                   m_snapshot2(:,(t_reached+b)/t_timesteps) = m(:,2); %
recording return amounts present at point t
                   m_snapshot3(:,(t_reached+b)/t_timesteps) = m(:,3); %
recording thresholds present at point t
                   if num_observations > 0

ratio_senders_observed_snapshot((t_reached+b)/t_timesteps) =
num_senders_observed / num_observations; % recording the ratio of senders with
observability at point t
                   else

ratio_senders_observed_snapshot((t_reached+b)/t_timesteps) = 0; % no
observations occurred
                   end
               end
               for j = 4:11
                   if t_reached+b == t_timesteps*(2^j)
                       if
abs(sum(average_send_over_time_not_observed(1:((t_reached+b)/2), 1)) -
sum(average_send_over_time_not_observed(((t_reached+b)/2+1):(t_reached+b),
```

```
1)))/((t_reached+b)/2) <= epsilon
                            if
abs(sum(average_threshold_over_time(1:((t_reached+b)/2), 1)) -
sum(average_threshold_over_time(((t_reached+b)/2+1):(t_reached+b),
1)))/((t_reached+b)/2) <= epsilon
                                if
abs(sum(average_send_over_time_observed(1:((t_reached+b)/2), 1)) -
sum(average_send_over_time_observed(((t_reached+b)/2+1):(t_reached+b),
1)))/((t_reached+b)/2) <= epsilon
                                    stop = j;
                                end
                            end
                        end
                    end
                end
            end
            counter2 = counter2 + 1; % number of times mutation occurred
            new_z = (rand - 0.5)/(scale); % random number to determine the new
mutation
            if mut_location == 0
                m_total = m_total + m*(number_of_time_periods-1); % accounting
for periods in between where nothing happens
                m(s,3) = m(s,3) + new_z; % the mutant replaces the threshold of
the last individual
                if m(s,3) > 1
                    m(s,3) = 1;
                elseif m(s,3) < 0
                    m(s,3) = 0;
                end
                t = t + number_of_time_periods; % fast-forwarding to the new time
period in which mutation occurred
                t_reached = t;
            else % mut_location > 0
                m_total = m_total + m*(number_of_time_periods); % accounting for
periods in between where nothing happens
                if rem(mut_location,3) == 1 % mutation in send amounts w/o
observing
                    mut_individual = fix(mut_location/3)+1;
                    m(mut_individual,1) = round(rand); % the mutant replaces the
strategy at location (k,1) with 0 or 1
                elseif rem(mut_location,3) == 2 % mutation in return amounts w/
observing
                    mut_individual = fix(mut_location/3)+1;
                    m(mut_individual,2) = m(mut_individual,2) + new_z; % the
mutant replaces the strategy at location (k,2)
                    if m(mut_individual,2) > 1
                        m(mut_individual,2) = 1;
                    elseif m(mut_individual,2) < 0
                        m(mut_individual,2) = 0;
                    end
                else % mutation in thresholds
```

```
                    mut_individual = fix(mut_location/3);
                    m(mut_individual,3) = m(mut_individual,3) + new_z; % the
mutant replaces the strategy at location (k,3)
                    if m(mut_individual,3) > 1
                        m(mut_individual,3) = 1;
                    elseif m(mut_individual,3) < 0
                        m(mut_individual,3) = 0;
                    end
                end
                mut_index = 0; % check for additional mutations within the same
generation
                while mut_index == 0
                    number_of_reproductions2 = geornd(u); % drawing a random
number of reproductions until next mutation
                    if mut_location + number_of_reproductions2 <= 3*s
                        counter2 = counter2 + 1; % number of times mutation
occurred
                        mut_location = mut_location + number_of_reproductions2;
                        new_z = (rand - 0.5)/(scale); % random number to
determine the new mutation
                        if rem(mut_location,3) == 1 % mutation in send amounts
w/o observing
                            mut_individual = fix(mut_location/3)+1;
                            m(mut_individual,1) = round(rand); % the mutant
replaces the strategy at location (k,1) with 0 or 1
                        elseif rem(mut_location,3) == 2 % mutation in return
amounts w/ observing
                            mut_individual = fix(mut_location/3)+1;
                            m(mut_individual,2) = m(mut_individual,2) + new_z; %
the mutant replaces the strategy at location (k,2)
                            if m(mut_individual,2) > 1
                                m(mut_individual,2) = 1;
                            elseif m(mut_individual,2) < 0
                                m(mut_individual,2) = 0;
                            end
                        else % mutation in thresholds
                            mut_individual = fix(mut_location/3);
                            m(mut_individual,3) = m(mut_individual,3) + new_z; %
the mutant replaces the strategy at location (k,3)
                            if m(mut_individual,3) > 1
                                m(mut_individual,3) = 1;
                            elseif m(mut_individual,3) < 0
                                m(mut_individual,3) = 0;
                            end
                        end
                    else
                        mut_index = 1; % next mutation goes out of the current
generation
                    end
                end
                t = t + number_of_time_periods + 1; % fast-forwarding to the new
```

```
time period in which mutation occurred
                t_reached = t;
            end
            m_total = m_total + m; % adding the current state of the population
            average_send_over_time_not_observed(t,1) = sum(m(:,1))/s;
            average_send_over_time_observed(t,1) = sum(m(:,2))/s;
            average_threshold_over_time(t,1) = sum(m(:,3))/s;
            if rem(t,t_timesteps) == 0
                m_snapshot1(:,t/t_timesteps) = m(:,1); % recording send amounts
present at point t
                m_snapshot2(:,t/t_timesteps) = m(:,2); % recording return amounts
present at point t
                m_snapshot3(:,t/t_timesteps) = m(:,3); % recording thresholds
present at point t
                if num_observations > 0
                    ratio_senders_observed_snapshot(t/t_timesteps) =
num_senders_observed / num_observations; % recording the ratio of senders with
observability at point t
                else
                    ratio_senders_observed_snapshot(t/t_timesteps) = 0; % no
observations occurred
                end
            end

        else % mutation happens after the max time period is reached
            for b = 1:(t_max-t_reached)
                average_send_over_time_not_observed(t_reached+b,1) =
sum(m(:,1))/s;
                average_send_over_time_observed(t_reached+b,1) = sum(m(:,2))/s;
                average_threshold_over_time(t_reached+b,1) = sum(m(:,3))/s;
                if rem(t_reached+b,t_timesteps) == 0
                    m_snapshot1(:,(t_reached+b)/t_timesteps) = m(:,1); %
recording send amounts present at point t
                    m_snapshot2(:,(t_reached+b)/t_timesteps) = m(:,2); %
recording return amounts present at point t
                    m_snapshot3(:,(t_reached+b)/t_timesteps) = m(:,3); %
recording thresholds present at point t
                    if num_observations > 0

ratio_senders_observed_snapshot((t_reached+b)/t_timesteps) =
num_senders_observed / num_observations; % recording the ratio of senders with
observability at point t
                    else

ratio_senders_observed_snapshot((t_reached+b)/t_timesteps) = 0; % no
observations occurred
                    end
                end
            end
            m_total = m_total + m*(t_max - t);
            t_reached = t_max;
```

```
            m_final_send_not_observed(:,1) = m(:,1); % send amounts at the last
point
            m_final_send_observed(:,1) = m(:,2); % return amounts at the last
point
            m_final_threshold(:,1) = m(:,3); % thresholds at the last point
            t = t_max + 1;
        end
    end
    t = t + 1; % moving to the next period
    if stop ~= 0
        t = t_max + 1;
    end
end
```