ACT-R BASED MEMORY MODELS OF ITERATED PRISONER'S DILEMMA

Thesis submitted to the

Institute for Graduate Studies in the Social Sciences

in partial fulfillment of the requirements for the degree of

Master of Arts

in Cognitive Science

by

Ayşegül Çetinkaya

Boğaziçi University

2009

ABSTRACT

Ayşegül Çetinkaya, "ACT-R Based Memory Models of Iterated Prisoner's Dilemma"

Iterated Prisoner's Dilemma game is an important tool for studying cooperation in social, biological and artificial environments. Various behavioral and neuroscientific experiments point to complex decision making and memory processes for human subjects. This thesis proposes four distinct memory models of Iterated Prisoner's Dilemma game that are built upon ACT-R cognitive architecture.

This work aims to overcome the shortcomings of a previous ACT-R based memory model by Lebiere et al. (2000), by providing extensive exploration of the parameter space and analysis of simulation results for all data points. Moreover, in contrast to previus work, this study introduces distinct declarative memory modules for each player. Third, model behavior is analyzed for the cases where it plays the game not only against itself, but against basic condional and unconditional strategies as well. Finally, by implementation of three new memory models for Iterated Prisoner's Dilemma, this study intends to attain cooperation against teaching strategies.

In decision making process, all memory models evaluate expected payoffs of possible moves according to the most likely outcome making that move. First model records game history in terms of frequency and recency of possible outcomes. Second memory model records outcome patterns that are experienced in the course of the game. Third model has a two step decision process where expected payoff is calculated according to both types of information about game history. Forth model employs an association mechanism between goal and declarative modules which enable the model to record outcome history in relation to contextual information that is kept in goal module.

After parameter setting, simulations are conducted for the cases where each model plays iterated game with itself and with basic game strategies. According to simulation results, all models were successful in exploiting and defending against unconditional strategies. Against teaching strategies, although they presented learning behavior, all models except third model have failed to attain cooperative equilibrium. First, second and forth models have adapted their behavior to exploit learning Pavlovian strategy and forgiving teaching strategies. All models exhibited learning behavior against basic strategies.

For the cases where each model plays the iterated game against itself, all models have successfully attained cooperation in a significant portion of the games. Apart from second model, all models exhibited a learning pattern consistent with human subjects. Moreover, similar to human subjects, simulated agents can be classified into teaching and learning groups according to their behavioral patterns.

TEZ ÖZETİ

Ayşegül Çetinkaya, "ACT-R Based Memory Models of Iterated Prisoner's Dilemma"

Tekrarlı Mahkum İkilemi oyunu sosyal, biyolojik ve yapay ortamlarda İşbirliği'nin araştırılması için önemli bir araçtır. Çeşitli davranış ve sinirbilim deneyleri insanların karmaşık karar verme ve bellek süreçleri olduğuna işaret etmektedir. Bu tez Tekrarlı Mahkum İkilemi oyunu için, ACT-R bilişsel mimarisi üzerinde geliştirilmiş dört farklı bellek modeli önermektedir.

Bu çalışma, paramete uzayının detaylı bir incelemesini ve tüm veri noktaları için simulasyon sonuçlarını sunarak, Lebiere et al. (2000) tarafından geliştirilen önceki ACT-R tabanlı bellek modelinin eksiklerinin giderilmesini hedeflemektedir. Bunun yanında, önceki çalışmadan farklı olarak, bu çalışma her oyuncu için ayrı tanımlanabilir (deklaratif) bellek modülleri sağlamıştır. Üçüncü olarak, model davranışı modelin yalnızca kendisiyle değil, temel koşullu ve koşulsuz stratejilerle karşılaştığı durumlar için de incelenmiştir. Bu çalışma üç yeni Tekrarlı Mahkum İkilemi bellek modelini geliştirerek, öğretme stratejilerine karşı da işbirliğine erişilmesini amaçlamaktadır.

Karar verme süreçlerinde, tüm bellek modelleri olası hamlelerin beklenen kazançlarını, o hamleyi yapmanın en muhtemel sonucuna göre değerlendirir. Birinci model, oyun geçmişini muhtemel sonuçların sıklığı ve zamansal yakınlığına göre kaydeder. İkinci model oyun sırasında deneyimlenen sonuç kalıplarını kaydetmektedir. Üçüncü modelin ise oyun geçmişi ile ilgili iki farklı bilginin de kullanıldığı iki adımdan oluşan bir karar verme süreci bulunmaktadır. Dördünce model amaç ve bellek modülleri arasında bağlantısal bir mekanizma kurarak sonuç geçmişini amaç modülünde tutulan bağlamsal bilgi ile birlikte kaydeder.

Parametre ayarlarından sonra, her modelin tekrarlı oyunu kendisi ve temel oyun stratejileri ile oynadığı simulasyonlar yapıldı. Simülasyon sonuçlarına göre, tüm modeller koşulsuz stratejilerden faydalanmada ve onlara karşı kendilerini savunmada başarılı oldular. Öğretme stratejilerine karşı, her ne kadar öğrenme davranışı sergileseler de, üçüncü model hariç hiçbir model işbirliği dengesine ulaşamadı. Birinci, ikinci ve dördüncü modeller, öğrenen Pavlovian stratejisiden ve affedici öğretme stratejilerinden faydalanmayı öğrendiler. Tüm modeller temel stratejilere karşı öğrenme davranışı sergiledi.

Her modelin tekrarlı oyunu kendine karşı oynadığı durumlarda, tüm modeller oyunların önemli bir bölümünde işbirliğine ulaşmayı başardılar. İkinci model dışında tüm modeller, insan davranışıyla tutarlı bir öğrenme davranışı sergilemektedir. Bunun yanında, yine insanlar gibi, simule edilmiş oyuncular da davranış kalıplarına göre öğrenen ve öğreten gruplar olarak sınıflandırılabilirler.

ACKNOWLEDGMENTS

First of all, I have been extremely fortunate to have Assist. Prof. Esra Mungan and Prof. Dr. Taner Bilgiç as my co-advisors, this thesis is the result of their intellectual contributions and insightful feedback.

I am deeply grateful to Assist. Prof. Esra Mungan for being so generous, patient and supportive at every step of the process. I would like to thank her for reading early versions of this text numerous times and providing detailed comments. She has turned the entire process into a delightful experience. Without her efforts, encouragement and generous support, I would have never completed my thesis. She is a true inspiration and role model for me.

I would like to thank many times to my co-advisor Prof. Dr. Taner Bilgiç for allowing me to take his Decision Analysis course and providing his valuable time to answer my numerous questions and to discuss my early ideas. He played a crucial role in shaping of my thesis with his tremendous support and encouragement.

I would like to thank members of thesis committee; Assoc. Prof. Ayşe Mumcu, Prof. Dr. Cem Say and especially Assist. Prof. Haluk Bingöl, their comments and insightful questions have improved my work significantly. I would also like to thank Prof Dr. Reşit Canbeyli and Prof. Dr. Güven Güzeldere for reading my thesis proposal and providing thoughtful criticism. I am grateful to Prof Dr. Sumru Özsoy for her efforts in admission process for Erasmus program. I would like to thank my instructors from Osnabrück University Cognitive Science Department, Angela Schwering and Ulf Krumnack, I have extremely benefited from their course on Cognitive Architectures.

Assoc. Prof. Yağmur Denizhan has always been a true mentor over years. The intellectual environment she has created has deeply influenced my thinking, I would like to thank her for being so generous with her time and knowledge.

Many thanks to my friends; Uğur Güney and İlker Yıldırım for giving me their valuable time for the discussion of my work. I have also deeply benefited from encouragement of my friend, Cem Erol. I would like to thank my close friends, Ilgın Erdem, Dilan Yıldırım and Ozan Yüksel, for their tireless support, love and encouragement whenever it is needed. Aslı Selim has always been a trusted source of advice throughout the thesis process. I would have never completed my thesis, without her support and encouragement.

None of it would have been possible without my mother and father, I am thankful for the love and support they have provided at every step of my life. Finally, greatest thanks of all go to my brother Ahmet Çetinkaya, he has supported me in every possible way. I do not know how to thank him enough for his invaluable time, patience and efforts. Without his belief in me, I would have never achieved so far in my life. This work is dedicated to him with my deepest love and gratitude.

CONTENTS

FIGURES

TABLES

1 INTRODUCTION

Iterated Prisoner's Dilemma Game is extensively studied by social, biological and computer sciences in order to investigate emergence and evolution of cooperative strategies in natural and artificial settings. Behavioral experiments and neuroscientific studies are conducted in order to explain complex behavioral patterns of human subjects. Human players are successful in reaching and maintaining cooperative equilibrium in different settings of the game, contrary to the game theoretical predictions where defective outcome is expected to become dominant. Different decision making strategies are proposed as good approximates of human behavior in Iterated Prisoner's Dilemma. Several experimental studies examine human behavior when playing iterated and single-shot Prisoner's Dilemma games in different settings. Recently, Neuroeconomics experiments have focused on the game in order to reveal brain structures which are involved in decision making processes.

Adaptive Control of Thought-Rational (ACT-R) is a cognitive architecture inspired from ongoing research in Neuroscience and Cognitive Psychology fields. ACT-R has been extensively used to build models of human cognitive processes in memory, visual, problem solving and decision making tasks. Recently, Lebiere et al. (2000) developed an ACT-R based memory model of Iterated Prisoner's Dilemma. This model tries to simulate human learning behavior and distribution of outcomes that are observed in experimental settings of Iterated Prisoner's Dilemma. Following Lebiere et al. (2000) model, four basic memory models based on ACT-R cognitive architecture are developed in this study.

First model is similar to Lebiere et al. (2000) model in terms of decision and memory processes. Model encodes game history in terms of frequency and recency of possible outcomes. However, the model tries to overcome two shortcomings of the previous model. In Lebiere et al. (2000) model, agents share a common declarative memory module which causes reinforcement of memory items for an agent due to internal memory processes of the other one. In first model of this study, an architectural change is introduced in order to provide two distinct memory modules for each player.

As a result only experienced game outcomes are reinforced in memory modules of both players, otherwise internal memory processes of a player are not available for the other one. Second, after setting the parameters, Lebiere et al. (2000) conducted several runs of the model. However, a sample of ten runs is selected due to its resemblance to empirical results as it consists of six cooperative runs, three defective runs and a mixed run. Mean frequencies of outcomes and behavioral learning plots are reported according to these ten runs. Performance and characteristics of the general population is not discussed by the researchers. In our study, performance and behavioral characteristics of memory models against themselves and basic Iterated Prisoner's Dilemma strategies are examined for all simulation data.

Behavioral experiments show that human subjects can successfully attain cooperative equilibrium against both teaching and learning Iterated Prisoner's Dilemma strategies. Simulation results reveal the inadequacy of first model in learning to cooperate against conditional teaching strategies. As a result, a second memory model is built to record game history in a different manner. Outcome of two consecutive rounds are recorded as a pattern in memory module and model collects information of frequency and recency of outcome patterns. Despite the modified memory processes, second model has failed to reach cooperative equilibrium against teaching strategies. Therefore a third memory model is implemented to overcome the inability to cooperate with teaching strategies. Third model uses a two-step decision process where present and future payoffs of possible moves are evaluated.

Following the success of third model in detecting conditional behavioral patterns of basic strategies, a forth memory model is proposed in order to detect the conditionality in opponent's behavior in a different manner. Forth model records game history similar to first model and keeps information about frequency and recency of game outcomes in memory module. In addition to that, forth model records an associations between goal items and memory items. Goal items provides a contextual information during the recording of game history. Model uses association information for easy retrieval of memory items in similar contexts.

In conclusion, this thesis proposes four ACT-R based memory models of Iterated Prisoner's Dilemma Game. Each model is evaluated for its performance against itself and against basic conditional and unconditional game strategies. Model success in attaining cooperation, model behavior and performance is extensively investigated for all cases.

In first chapter, payoff structure and properties of Iterated Prisoner's Dilemma game are discussed in detail. In order to evaluate performance of memory models in different conditions, several basic Iterated Prisoner's Dilemma strategies are built in ACT-R environment. Characteristics and rules of All-Cooperate, All-Defect, Random, Tit-for-Tat, Tit-for-Two-Tats, Forgiving-Tit-for-Tat and Pavlovian strategies are also presented. Behavioral and neuroscientific studies of Iterated Prisoner's Dilemma game are briefly discussed in first chapter. Basic characteristics and functioning of ACT-R cognitive architecture and ACT-R based memory model of Iterated Prisoner's Dilemma by Lebiere et al. (2000) are also presented. Lastly, chapter covers the basic characteristics and evaluation criteria for memory models.

Implementation of game environment, basic strategies and memory models are discussed in second chapter. Architecture and functioning of game environment in ACT-R are explained in detail. This chapter includes a detailed analysis of memory and decision making processes and parameter space of four memory models.

Third chapter presents the results for computer simulations. Each model has played the iterated game of 300 rounds against basic strategies and against itself for 1000 times. Distribution of outcomes, learning patterns for different models are discussed in the chapter. In addition to that, models are examined in terms of conditional cooperation probabilities. Finally, conclusion chapter provides a summary of the results and relevance to previous work. Ideas related to future work is also briefly presented in the last chapter.

1.1 Iterated Prisoner's Dilemma Game

Prisoner's Dilemma constitutes an important problem in Game Theory. It is a 2-player game where each player has two options, they can either choose to cooperate or defect. This game is a non zero-sum game, the sum of gains and losses of players is not equal to zero. When both of them choose to cooperate simultaneously, in [cooperate, cooperate] case, they were better off compared to the case where both players choose to defect as [defect, defect] outcome is observed. However, if a player defects the cooperating partner and game is resulted in [defect, cooperate] or [cooperate, defect] outcome, defecting player gets the best payoff compared to all the other outcomes and cooperating player ends up with the worst payoff. Payoff structure is illustrated in an example of Prisoner's Dilemma game in Figure 1.

		Player 2	
		Cooperate	Defect
Player 1	Cooperate	2 , 2	0 , 3
	Defect	3 , 0	1 , 1

Figure 1: Payoff matrix for an example of Prisoner's Dilemma game

Although there are n-player and asymmetric versions of the Prisoner's Dilemma, this study will focus on the classical version, 2-player game with the symmetric payoff structure. Generalized payoff structure for the 2-player, symmetric game is demonstrated in Figure 2.

		Player 2	
		Cooperate	Defect
Player 1	Cooperate	x , x	z , w
	Defect	w , z	y , y

Figure 2: Payoff matrix for general form of the Prisoner's Dilemma game, $w > x > y > z$

According to formal game theory, cooperation strategy is strictly dominated by

the defect strategy. Irrespective of the other player's decision, defect strategy yields a better outcome for either player. If we make the assumption that both players are rational which means that they build strategies according to the belief about the rationality of the other player and they try to maximize the outcome of the game, both players choose defect strategy and [defect, defect] is the Nash equilibrium outcome of this game. An important point is that Prisoner's Dilemma is different from similar games which Nash equilibrium exists in the sense that here, equilibrium solution is not Pareto-optimal. This means that there is another possible outcome where at least one of the players is better off. In the [cooperate, cooperate] case both players are better off compared to the Nash Equilibrium point.

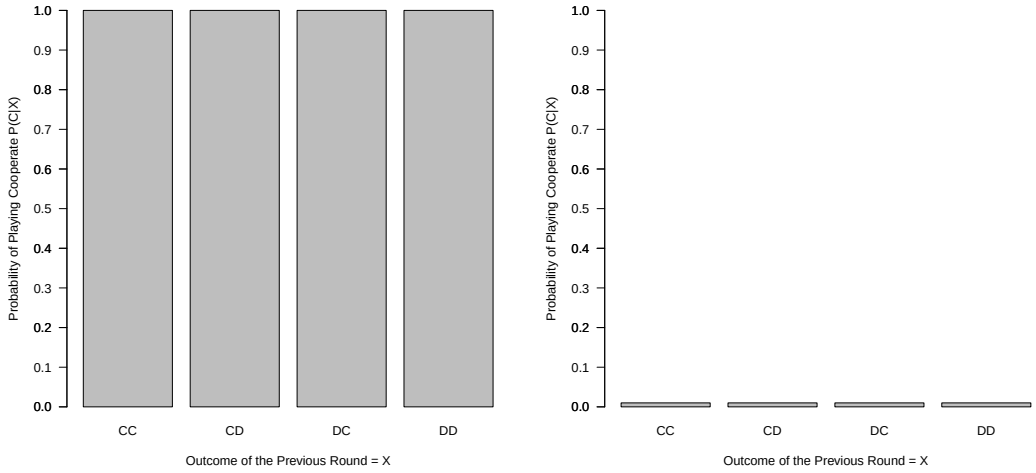
Iterated Prisoner's Dilemma is the iterated form of the game. Game is played several consecutive rounds by the same players. Game theoretical prediction for this variant of the game is similar to the one-shot version and equilibrium strategy is predicted as choosing defect by the theory.

1.2 Basic Iterated Prisoner's Dilemma Strategies

Iterated Prisoner's Dilemma is extensively studied in evolutionary game theory with agent-based computer simulations in order to investigate emergence of cooperation in environments where long-term and short term interests of the players usually conflict. Several basic strategies are used in similar computer simulations as building blocks for game strategies, these strategies are also implemented in our game environment. These basic conditional and unconditional strategies form a tool box for the assessment of model performance. Basic strategies are All-Cooperate, All-Defect, Tit-for-Tat, Forgiving Tit-for-Tat, Tit-for-Two-Tats, Pavlovian and Random strategies.

All-Cooperate or ALLC strategy chooses to cooperate unconditionally, irrespective of the outcome of the previous round. Probability to cooperate after observing each possible outcome is one, as depicted in Figure 3. This strategy is successful in cooperation with strategies that reward cooperative actions. However, it can not defend itself against defecting strategies and fails to punish defecting actions of the opponent.

All-Defect or ALLD, on the other hand chooses to defect in all rounds regardless of the past actions of both players (Figure 3). ALLD strategy is successful against unconditionally cooperative strategies. This strategy is disadvantageous against conditional strategies, since its defecting behavior is punished repetitively.

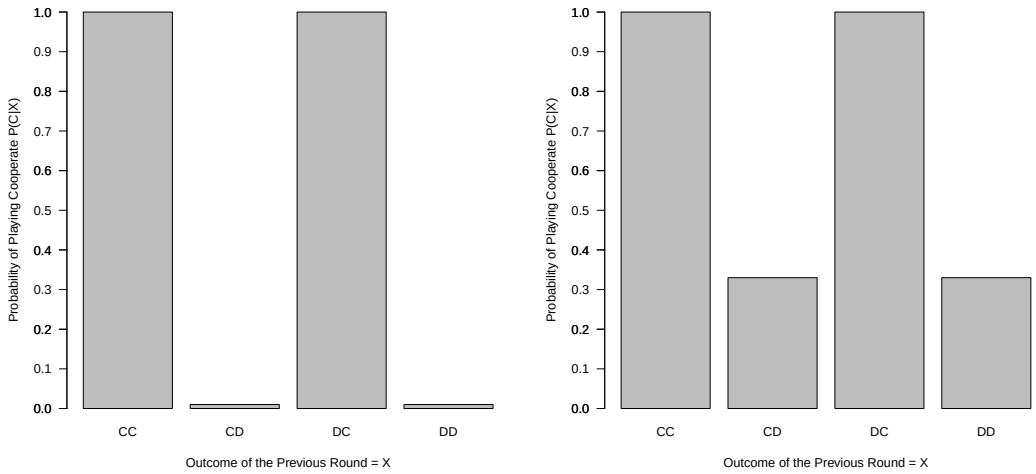


(a) ALLC Strategy (b) ALLD Strategy

Figure 3: Probability of playing cooperate for first player who employs (a) ALLC strategy or (b) ALLD strategy, after outcome of the last round [Move of First Player, Move of Second Player] is observed.

When compared to unconditional strategies like ALLC and ALLD, conditional strategies perform better as they can modify their behavior according to game history. Tit-for-Tat is a conditional strategy which reciprocates cooperative behavior by cooperation and punishes defect move of the other player by defecting itself. TFT player starts the game by cooperating and repeats the move of the other player in the previous round. Probability of cooperating is equal to one after cooperate action of the other player is observed in the previous round (Figure 4). TFT player is successful in cooperation when playing against cooperative players and defends itself against defecting players. Main disadvantage of TFT strategy is that it is unforgiving when defect move of other player is observed. Forgiving Tit for Tat strategy is forgiving by a certain probability, usually 1/3, after defect action is observed. Otherwise it is similar to Tit for Tat Strategy (Figure 4).

Tit-for-Two-Tats strategy is a variation of TFT strategy. Tit-for-Two-Tats strategy



(a) TFT Strategy (b) Forgiving TFT Strategy

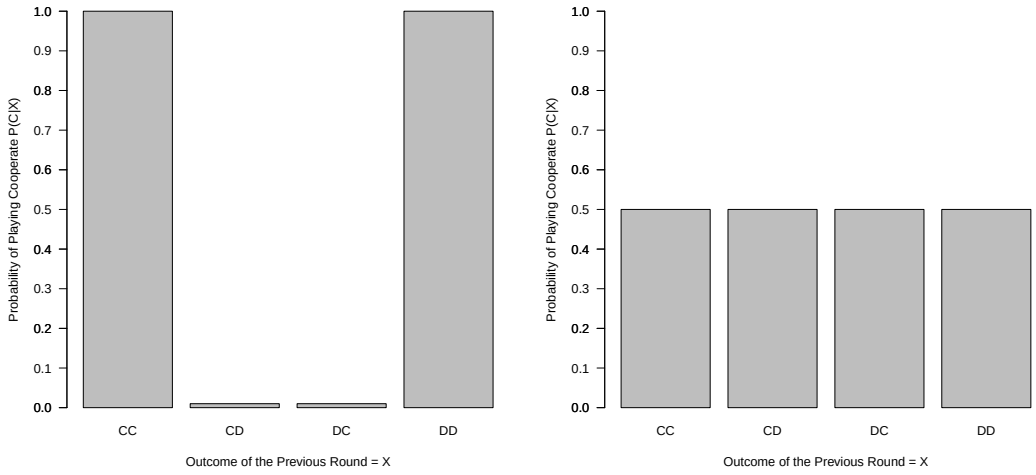
Figure 4: Probability of playing cooperate for first player who employs (a) TFT strategy or (b) Forgiving TFT strategy, after outcome of the last round [Move of First Player, Move of Second Player] is observed.

starts the game with cooperation and shifts to defect move when other player defects in previous two rounds consecutively. Punishment is less severe compared to TFT strategy since the strategy does not react to first defective move of the other player. Tit-for-Two-Tats player is equally forgiving as the TFT player, as it shifts to cooperate move when second player chooses to cooperate in the previous round.

Pavlovian strategy employs "win stay, lose shift" idea. Player continues to defect after winning against a cooperative move, but shifts to cooperate move after its defect move is punished in the previous run. This strategy continues cooperating after mutual cooperation is observed. However, it punishes other player by defecting after its cooperate move is answered with a defect move. Cooperation probability with respect to the outcome of the previous round is illustrated in Figure 5. Main difference between TFT and Pavlovian strategies manifests itself in conditional probabilities of cooperation after [defect, cooperate] and [defect, defect] outcomes are observed. Pavlovian strategy shifts to cooperate after both players defected in the previous round, whereas TFT strategy chooses to punish the other player by defecting again. After [defect, cooperate] outcome is observed in the previous round, Pavlovian strategy continues to defect, TFT strategy on the other hand rewards cooperate move of the other player in

the previous round by shifting to cooperation in the next round.

Last basic strategy is Random strategy. Random player chooses to defect and cooperate with equal probability irrespective of the previous round of the game (Figure 5).



(a) Pavlovian Strategy (b) Random Strategy

Figure 5: Probability of playing cooperate for first player who employs (a) Pavlovian strategy or (b) Random strategy, after outcome of the last round [Move of First Player, Move of Second Player] is observed.

1.3 Decision Making and Learning in Iterated Prisoner's Dilemma

There were many behavioral experiments conducted since 1960s after the formulation of the game. Experiments conducted by Andreoni & Miller (1993) provide a good illustration of the behavioral results. Human subjects randomly matched with each other 200 rounds, each with a different human partner. Although the expected behavioral result is defect, a significant percentage of participants chose to cooperated in all rounds (Figure 6). This deviation from the game theoretical prediction even in one-shot games is attributed to the altruistic nature of human beings (Andreoni & Miller, 1993).

Iterated Prisoner's Dilemma is investigated by several computer simulation studies as part of evolutionary game theoretical research program, since 1980s. Several strategies are implemented as computer programs and they contested with each other in different evolutionary settings. It is generally accepted that evolutionary settings

can produce strategies which can successfully generate mutual cooperation outcomes (Axelrod, 1984).

It is important to note that, cooperative equilibrium based on reciprocal altruism is generally not observed in animal species (Axelrod & Hamilton, 1981). Iterated Prisoner's Dilemma is the main framework of cooperative behavior among animals. It is believed that strong preferences for immediate gains can explain instability of the cooperative equilibrium (Stephens et al., 2002). Therefore, it can be concluded that a mechanism that values long-term benefits and punishments over short-term immediate gains is needed for the evolution of cooperative equilibrium and strategies (Rilling et al., 2007).

Tit-for-tat strategy is proved to be very effective against a wide range of strategies by computer simulations. As Axelrod (1984) noted this strategy helps players to reach mutual cooperation which seems to be the best long term strategy.

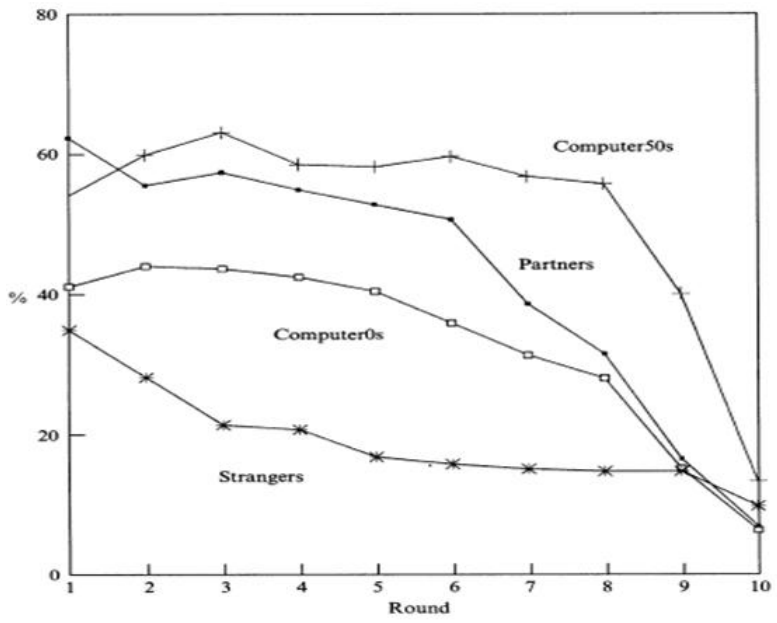


Figure 6: Percentage of cooperation in Iterated Prisoner's Dilemma experiments by Andreoni & Miller (1993).

Andreoni & Miller (1993) conducted experiments where subjects played ten-round iterated Prisoner's Dilemma 20 times with different human partners .They exhibited a high level of cooperate behavior, around fifty percent and they delayed defect response. However, they tend to switch to defect strategy in the final rounds as predicted. Their

behavior is consistent with the reciprocal altruism hypothesis. The belief of there are "irrational " players which would play cooperate strategy may cause rational players to switch to cooperate strategy in order to obtain a higher outcome in the iterated version of the game. It was argued that in order to generate mutual cooperation, players have to believe a certain portion of the possible partners will play cooperative strategies. In order to test this hypothesis, subjects played Iterated Prisoner's Dilemma against computer partners employing Tit-for-Tat strategy with fifty percent probability. Results were similar to the case where subjects were matched against human partners (Figure 6). Therefore, we can conclude that people do believe a significant portion of possible partners would have altruistic motives, in fact behavioral results of the one-shot games imply that some portion of the population actually behave altruistically and people tend to employ strategies which build an altruistic reputation (Andreoni & Miller, 1993).

Prisoner's dilemma usually played in a simultaneous fashion where both players decide simultaneously and behavior of the other player is inferred from the outcome of the game. There are other variations of the game where game is played in a sequential way and second player decides after observing the first player's action. In alternating Prisoner's Dilemma, two players play iterated sequential Prisoner's Dilemma while changing the role of first player in subsequent rounds. Wedekind & Milinski (1996) conducted experiments where subjects played simultaneous and alternating versions of Iterated Prisoner's Dilemma. Experimental results revealed that subjects were consistent in their strategies and 30 percent of the subjects adopted a Forgiving Tit-for-Tat strategy, whereas 70 percent preferred a Pavlovian strategy. In behavioral experiments subjects employed an improved version of Pavlovian strategy, where they choose to cooperate in the next round after mutual cooperation or mutual defection with a probability less than one, this way player can make advantage over cooperative strategies and player can defend itself to unconditional defect strategies better than the classic Pavlovian strategy.

Several computer simulations conducted in order to investigate the success of strategies in evolutionary settings. These studies revealed that against unconditional strate-

gies, Pavlovian players were much more successful in simultaneous games and Forgiving Tit-for-Tat strategies were more successful in alternating Prisoner's Dilemma. Wedekind & Milinski (1996) study shows that both Pavlovian and Forgiving Tit-for-Tat players exist in human population. However, these players failed to adapt their behavior according to the type of the game they played.

In a later study, Milinski & Wedekind (1998) investigated memory constraints on strategy selection in iterated Prisoner's Dilemma. In unconstrained case, most of the subjects employed complex and improved forms of Pavlovian strategy and others used Forgiving Tit-for-Tat strategy. When constrained by a second memory task the proportion of subjects who used Forgiving Tit-for-Tat strategy increased, since the working memory requirement of this strategy is smaller than complex Pavlovian strategy employed in the previous case. Subjects who stick to the Pavlovian strategy on the other hand performed poorly in the memory task.

As Camerer (2003) noted, human beings are capable of not only learning to adapt their strategies according to other player, but also observing that their partners are learning. They use this fact for their advantage and adopt teaching strategies that will alter the beliefs of the other player about their reputation. Baker & Rachlin (2002) conducted a study in order to investigate adoption of teaching and learning strategies against human and computer partners. In first experiment, subjects were told that they were playing against computer partners, however they were not given any information about the strategy employed by the computer player. When computer was playing a teaching strategy (Tit-for-Tat), subjects learned to cooperate with it with a learning strategy (Pavlovian strategy). When computer was playing a learning strategy (Pavlovian strategy), subjects successfully adopted a teaching strategy (Tit-for-Tat). In the second set of experiments, when subjects were told that they were playing against a human partner who was using the same teaching strategy, they failed to employ a learning strategy, they constantly tried to use a teaching strategy against the human partner assuming that the player is using a learning strategy. In the second condition, they were told that they are playing against a computer strategy and the probability of

cooperation is signaled by a spinner. Actually computer was using a learning strategy, but subjects failed to recognize learning behavior and failed to develop a cooperative strategy in this game. These results highlight the contextual effects on learning and behavior in strategic games.

Rick & Weber (2008) investigated the role of meaningful learning in solving strategic games. They started with the assumption that human beings are capable of two different learning methods. First method is reinforcement learning, where human beings acquire an implicit or procedural knowledge. This knowledge is the result of positive and negative reinforcers signaling reward and punishment as the outcomes of actions. The second type of learning is conceptual, symbolic or meaningful learning. As a result, people will acquire declarative general knowledge which can be applied to similar problems regardless of the contextual differences between problems. They tested this hypothesis by making subjects play strategic games several rounds without observing the other players behavior, hence the outcome of the game. Each of these games has solutions that can be reached by applying iterated elimination of dominated strategies. A dominated strategy is a strategy which pays equal or less when compared to other strategies regardless of the other player's choice. When played by observing the behavior of others, people employed reinforcement learning methods and reached equilibrium solution eventually in later rounds of the game. However, they failed to transfer their strategy to similar type games. When subjects played these games without observing the outcome, they failed to employ reinforcement learning. However, they learned to implement iterated elimination of dominated strategies method and they were also successful in transferring this solution strategy to much more complex but similar games. They successfully gained general meaningful solution methods without reinforcement learning.

1.4 Neural Correlates of Decision Making in Prisoner's Dilemma

Rilling et al. (2002) conducted an fMRI study of subjects playing Iterated Prisoner's Dilemma Game. Subjects believed they were playing with human partners, they were

actually playing with a Forgiving Tit-for-Tat strategy. The results show that mutual cooperation is correlated with activation in brain areas which are closely related with reward processing such as Nucleus Accumbens, Caudate Nucleus, Ventromedial Frontal Cortex, Orbitofrontal Cortex and Rostral Anterior Cingulate Cortex. Researchers concluded that this neural network reinforces cooperative equilibrium and prevents subjects from making impulsive self regarding decisions.

Rilling et al. (2007) claimed that Prisoner's Dilemma is unique in the sense that this optimal strategy may be the result of either emotional or cognitive processes. A player may choose to cooperate or defect according to its emotional processing or he can employ the strategy as a result of its cognitive, strategical thinking in order to maximize the long term outcome.

Recently, Rilling et al. (2007) reported a study on the relation of psychopathy and cooperation. In this imaging study, subjects played Iterated Prisoner's Dilemma game against a computer playing the game according to a Forgiving Tit-for-Tat strategy, however they were told that they were playing against human partners. Subjects were required to complete two self-report psychopathy questionnaires. Patients suffering from psychopathy disorder present failure to experience emotions which enable the individual to exhibit appropriate social behavior. Similar to Orbitofrontal damage patients, they also suffer from egocentricity, impulsivity, irresponsibility, shallow emotions, lack of empathy and guilt which are crucial for social behavior and mentalizing. However they try to overcome their incapability by cognitive reasoning mechanisms (Rilling et al., 2007).

Results of the experiment indicate that, subjects which score higher on psychopathy tests chose defect strategy more often and they have a tendency of not cooperating after a mutual cooperation is established with the partner. When their cooperation is not reciprocated and [cooperate, defect] outcome is observed, they showed less Amygdala activation. In comparison to subjects with low psychopathy scale scores, they exhibit less activation in Orbitofrontal cortex when they are cooperating and less activation in Dorsolateral Prefrontal Cortex and Rostral Anterior Cingulate Cortex when they are

defecting. Subjects with low psychopathy scores have a tendency to cooperate whereas subjects with high psychopathy scores are biased towards defect strategy. However, both tendencies can be overcome by effortful cognitive control (Rilling et al., 2007).

1.5 Cognitive Architectures, Adaptive Control of Thought - Rational

Cognitive architectures provide the basic structures for computational cognitive modeling. While some of the cognitive architectures were developed in order to execute specific cognitive tasks, others such as ACT-R (Adaptive Control of Thought-Rational) try to simulate cognitive capabilities of human beings. Cognitive architectures have become important tools in order to explain cognitive mechanisms behind certain patterns of behavior. Researchers hope to integrate cognitive architectures with multi-agent methods in order to provide successful social simulation tools for social sciences (Sun, 2006).

Every cognitive architecture has different assumptions related to modularity and representation of rules and knowledge. Some cognitive architectures inspired from functional and anatomical research related to human brain. Others, on the other hand, focus only on implementing intelligent behavior and providing intelligent solutions for puzzles and problems in fields such as problem-solving, reasoning and decision-making. Cognitive Architectures such as SOAR, ACT-R, CLARION, ICARUS are widely accepted among researchers, Anderson et al. (2004) provides an extensive review of modules, memory structures and functioning of these architectures.

ACT-R has been developed since late nineteen seventies, focus of ACT-R is modeling human behavior, model and processing mechanisms are mainly inspired from recent research in Neuroscience. ACT-R is composed of four processing modules, each processes a different type of information (Figure 7). First module is sensory module and its main responsibility is to process visual or auditory sensory information. Second module is motor module and it executes actions, third module is the intentional module which is responsible for goal maintenance and processing. Last module is the declarative module which is basically a long-term memory structure and holds long-

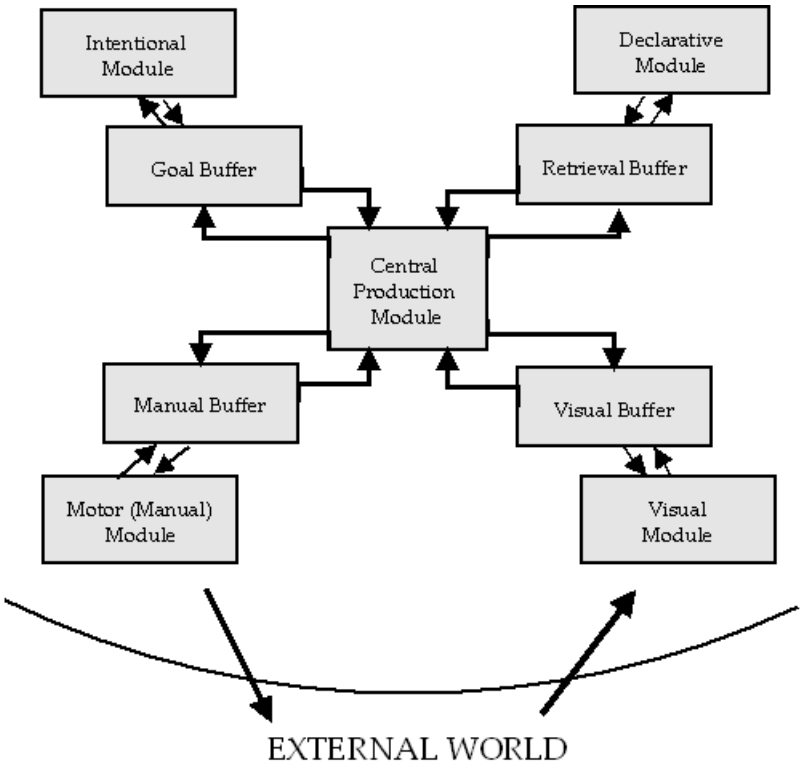


Figure 7: Overview of ACT-R cognitive architecture (Anderson et al., 2004).

term declarative knowledge. Recently, a fifth module, imaginal module is introduced which is similar to intentional module in its functioning. This module is generally used for keeping mental representations in problem-solving tasks. Each module has its own buffer which can hold one memory structure at a time. This basic memory structure is referred as chunk in ACT-R. These four buffers constitute the short-term memory of the cognitive architecture.

Each chunk in the declarative memory is retrieved according to certain rules based on the activation of the chunk. When a certain chunk is requested from declarative memory, declarative memory calculates the activation of all matching chunks, recalls the chunk with the highest activation among matching chunks and pushes the recalled chunk into the memory buffer. Activation of all matching chunks are calculated according to activation equation. First term in the activation equation is base-level activation and reflects past references to the chunk. Base-level activation is highest when there are multiple references to the chunk and when the chunk is recently added to the declarative memory or recalled from the memory. When a chunk is added to the

declarative memory, declarative memory module checks all the chunks in the memory. If declarative module finds a chunk with the same content, instead of adding the chunk as a new entry, the module merges two chunks as a single entry. Addition of the new chunk is recorded as a reference to the existing chunk, therefore increases activation of the original chunk.

$$A_i = B_i + \sum_j W_j S_{ji} \qquad \text{(Activation Equation)}$$

Second term in the activation equation is attentional activation where W_j symbolizes attentional w of the items in the goal and S_{ji} reflects the strength of association between item j and chunk i. Strength of association depends on the degree of similarity between goal items and chunk items. As the number of matching items between goal and memory chunks increases, association strength and attentional activation increase accordingly.

Base-level activation of each chunk is modified according to past use of the chunk. Each reference to the chunk increases the activation of the chunk. Retrieval of the chunk from memory elevates the chance of retrieval in the future by increasing the activation level of the chunk. However, activation of each chunk decreases with time according to a decay rate which can be modified by the programmer. Base level activation depends on time past since last reference t_{j} and a certain decay factor d. The decay factor is usually set to 0.5 which has proved to be a good approximation in several simulations (Lebiere et al., 2000: Anderson et al., 2004).

$$B_i = ln(\sum_{j=1}^{n} t_j^{-d}) \qquad \text{(Base Level Activation Equation)}$$

Each memory structure in the declarative module is associated with an activation which depends on the matching between current context and recent or frequent usage of the chunk. Activation determines the retrieval time and chunks with low base activation may not be retrieved at all. The retrieval of a certain chunk depends on activation of that chunk, the threshold of retrieval and noise level in the activation process. Retrieval probability of a certain chunk is determined by the following equation:

$$P_i = \frac{1}{1+e^{-(A_i-\tau)/s}} \qquad \text{(Probability of Retrieval)}$$

In the formula, τ symbolizes retrieval threshold and s sets the noise level in the activation process. These parameters are controlled by the experimenter according to simulation requirements.

ACT-R includes a central production unit which constitutes the procedural memory of the system. This central unit is connected to other four modules indirectly through buffers. Production unit coordinates the processing of the other modules. Production unit processes information chunks in the buffers and it may change the content of the buffers which later may trigger different processes in modules such as initiating an action in motor module or retrieving a memory chunk from declarative module.

Central production system stores production rules which are activated by the matching chunks in the buffers, their activation also depends on the base activation of the chunks. Each production rule is associated with a utility which is an estimation of the success of achieving the goal when this particular rule is chosen and a cost associated with the production rule. Cost is determined by time in ACT-R. Production rule with the highest utility is selected for execution according to the following equation:

$$U_i = P_i G - C_i \qquad \text{(Production Utility Equation)}$$

C_i represents the cost of the production rule and determined by the time. P_i reflects the probability of success in achieving a certain goal by using production rule i. G reflects the value of the goal and is estimated by the experimenter. U_i symbolizes the utility associated with the production rule and determines which production rule is chosen.

$$p_i = \frac{e^{U_i/t}}{\sum_{j}^{n} e^{U_j/t}} \qquad \text{(Production Selection Equation)}$$

Probability of firing for the production rule is determined by the equation above, production rule with the highest probability fires. However there is also a noise parameter which allows production rules with lower utilities to fire with a certain probability.

Noise parameter allows for different production rules to be explored by the central module. Noise parameter is set by the programmer.

Learning in ACT-R is implemented in different levels. Base activation levels are adapted regularly, when chunks are used by production rules, their base activation levels increase, otherwise they decay. Similarly, cost and success rate of production rules are updated according to the observed behavior of the system. Production rules are in conditional forms, new rules are learned as combination of successful rules or generalization of successful solutions (Langley et al., 2009: Anderson et al., 2004).

ACT-R cognitive architecture is applied to several research questions about Cognition. Computer simulations of experiments attempt to provide mechanisms for cognitive tasks in fields such as Memory, Visual Search, Attention, etc. Main goal of ACT-R research is to build an integrated theory of Mind and Cognition (Anderson et al., 2004). Results of ongoing brain research and imaging studies are also incorporated in ACT-R theory by continuous updates in architecture. ACT-R architecture in its classical form has its own programming language and an interpreter which is based on LISP programming language.[1] There are also different versions of architecture that are implemented with Java (jACT-R) and Python (Python ACT-R) programming languages. Users can modify architecture, introduce new modules and processing rules, in order to meet the requirements of the cognitive task in hand.

Decision Making and Learning in Decision Making are also studied using ACT-R cognitive architecture. Recently an important decision making task, Iowa Gambling Task is investigated by Stocco et al. (2005). Researchers provided a model of decision making, game representation and learning for the Iowa Gambling Task. Simulation results suggested an explanation for discrepancies in behavioral results between neurologically intact subjects and subjects suffering from brain damage. ACT-R cognitive architecture is also used for constructing models of learning and decision making in different games.

[1]Documentation, software and tutorials for ACT-R cognitive architecture is available at http://act-r.psy.cmu.edu/

1.6 A Memory Based ACT-R Model of Iterated Prisoner's Dilemma by Lebiere et al. (2000)

Games are extensively studied not only in economical and social sciences but in computer science as well. Games are particularly important for artificial intelligence studies, since they provide a suitable environment for testing problem solving, reasoning, decision making and learning skills. ACT-R cognitive architecture introduces helpful tools and structures for creating and testing models of decision making in games.

Lebiere & West (1999) presented a memory based ACT-R model of Paper Rock Scissors game. This game is a 2-player zero-sum game. Their model relies only on declarative memory module of ACT-R Cognitive Architecture and activation based learning mechanism of the memory module. This agent model keeps the last three actions of the opponent in its declarative memory. When playing the game, agent utilizes the information about the action patterns of the other player in order to predict the next move of the second player. After predicting the next move of the opponent, agent chooses the best move to beat the other player. After making the move, and observing other player's action, player modifies its declarative memory accordingly. This strategy is summarized in the following algorithm, provided by Lebiere et al. (2000) , p.188:

```
IF the opponent played moves A2 and A1
    the moves A2 and A1 are most likely followed by move A
    the move A is beaten by move M
THEN make move M, record opponent's move and push a new goal to make the
next play
```

Following this study, Lebiere et al. (2000) modified this activation-based memory model for Iterated Prisoner's Dilemma game. In Iterated Prisoner's Dilemma game, defect is the best counter move irrespective of the opponent's choice of action. Therefore predicting opponent's move in the next round is not helpful for the decision making process. Hence, Paper Rock Scissors model can not be directly applied for Iterated Prisoner's Dilemma game. Instead of trying to predict the next move of the other player, memory model predicts the resulting outcome of possible actions. Player re-

calls the most likely outcome of making defect move and the most likely outcome of playing cooperate from declarative memory. Then, player compares these two outcomes, and selects the move with the highest payoff. Outline of the strategy is presented below:

```
IF the goal is to solve prisoner's dilemma
    the most likely outcome of making move A is payoff a
    the most likely outcome of making move B is payoff b
THEN make move with the highest payoff,
    record opponent's move and push a new goal to make the next play
```

Model operates on the variant of the Prisoner's Dilemma game which was first used in behavioral experiments conducted by Rapoport et al. (1976). Payoff matrix of the game is presented below:

		Player 2	
		Cooperate	Defect
Player 1	Cooperate	1 , 1	-10 , 10
	Defect	10 , -10	-1 , -1

Figure 8: Payoff matrix for Prisoner's Dilemma Game used in Lebiere et al. (2000); Rapoport et al. (1976).

Each outcome is encoded as a memory chunk in declarative memory. Four memory chunks translate this payoff matrix into declarative knowledge for the model. Each chunk represents an outcome, moves and payoff structure associated with this outcome. These chunks are encoded in ACT-R programming language:

```
(C1-C2 isa outcome move1 C move2 C payoff1 1 payoff2 1)
(C1-D2 isa outcome move1 C move2 D payoff1 -10 payoff2 10)
(D1-C2 isa outcome move1 D move2 C payoff1 10 payoff2 -10)
(D1-D2 isa outcome move1 D move2 D payoff1 -1 payoff2 -1)
```

According to the memory model, each chunk is activated when it is retrieved from memory or observed as the outcome of the previous round. One important aspect of this memory model is related to access to the memory structure. In this model both players use same memory module during decision making process. Therefore, each memory chunk is reinforced as it is recalled by either first or second player or as it is

observed as the result of the previous round. After a memory request is made to the declarative memory module, each player retrieves the most active chunk. Activation of each chunk is determined by the base-level activation equation used in the model:

$$A = ln\sum_{j=1}^{n} t_j^{-d} + N(0, \frac{\pi.s}{\sqrt{3}}) = ln\frac{n.l^{-d}}{1-d} + N(0, \frac{\pi.s}{\sqrt{3}})$$

In this activation equation, first term illustrates the sum of reinforcement for the chunk, t_j is the amount of time since the jth reference, n represents the number of references to the chunk, d is the decay rate and refers to the decay level of the memory chunks. First term can be approximated by a logarithmic function which depends on n, number of references; d, decay rate and l total life time of the chunk. Second term in the equation refers to the variation in the retrieval process, it is a normally distributed noise with mean zero and variance is determined by s parameter. As Lebiere et al. (2000) states, behavior of the model depends on the parameters. Researchers set decay parameter as 0.5 and noise as 0.25, since these values are widely used in similar models. Initial number of references to each outcome chunk, L parameter is determined as 10 and it is a specific parameter for this model.

Table 1: Frequencies of Four Outcomes in Iterated Prisoner's Dilemma Game.

(a) Data (Rapoport et al., 1976)

Pair	C1-C2	C1-D2	D1-C2	D1-D2
1	97	1	1	1
2	92	1	1	7
3	83	2	1	14
4	86	5	5	4
5	72	3	4	21
6	66	5	5	24
7	27	7	12	54
8	11	52	2	34
9	12	5	25	58
10	3	4	9	83
Mean	55	8	7	30

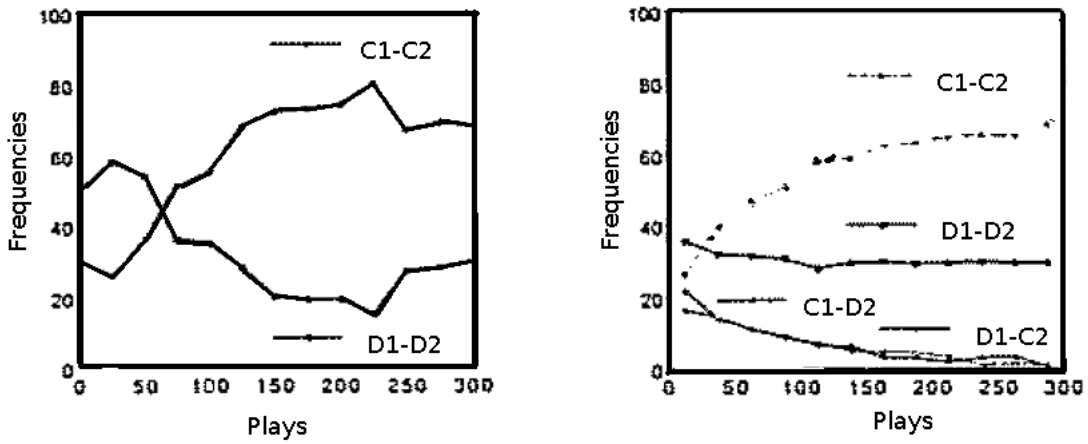
(b) Model (Lebiere et al., 2000)

Pair	C1-C2	C1-D2	D1-C2	D1-D2
1	65	12	13	10
2	97	2	0	1
3	65	12	19	4
4	1	3	4	92
5	1	3	3	93
6	96	2	1	1
7	0	2	3	95
8	48	18	21	13
9	87	2	9	2
10	81	10	4	5
Mean	54	6	8	32

After setting the parameters, they conducted several runs of the model and selected ten runs in order to compare simulation results and empirical results. This sample of ten runs is chosen due to its resemblance to empirical results as it consists of six

cooperative runs, three defective runs and a mixed run. Outcome frequencies of the model and empirical data are presented in Table 1. Average frequency of asymmetric outcomes, [cooperate, defect] and [defect,cooperate] outcomes are lower compared to symmetric outcomes, [cooperate, cooperate] outcome is chosen more than fifty percent of games and [defect, defect] outcome is observed with thirty percent frequency.

Learning thorough the course of the game for this model and empirical data are compared in Figure 9. Similar learning trends observed in these two figures. At the initial rounds of the game, frequency of [defect, defect] outcome is higher compared to other outcomes. However, this frequency decreases with time up to thirty percent. Frequency of [cooperate,cooperate] outcome on the other hand, increases up to seventy percent through the course of the game. Percentages of non-symmetric outcomes decrease to around ten percent at the later stages of the game. At the initial stages of the game, players are inclined to select defect move. Hence, [defect, defect] outcome is reinforced. At later stages, players base their decision on the comparison between [defect,defect] and [cooperate,cooperate] outcomes. Since payoff structure of the latter is advantageous, cooperative outcome is reinforced at the later stages of the game. Therefore, frequencies of asymmetric outcomes decrease as the game progresses.



(a) Data (Rapoport et al., 1976) (b) Model (Lebiere et al., 2000)

Figure 9: Change of outcome frequencies over time in Iterated Prisoner's Dilemma Game.

The ACT-R model of Lebiere et al. (2000) relies mainly on the activation based declarative memory system and reinforcement of memory chunks which represent

game outcomes. Each memory chunk is reinforced in two cases. First, when it is created before the start of the rounds or re-created in the course of the game when outcome of the previous round is observed. Second, a memory chunk is activated and reinforced as it is recalled from memory after a memory request is made. As Lebiere et al. (2000) state, ACT-R memory system reinforces a memory chunk when it is experienced or when it is recalled from declarative memory module.

In Lebiere et al. (2000) model, both players use same declarative memory module when they are playing the game. This means that when a player recalls a memory chunk, this chunk is reinforced for the other player as well. Lebiere et al. (2000) claims that this feature of the model resembles the decision making process where each player takes the other player's point of view.

Another characteristic of the ACT-R model is that decision-making depends only on relative ordering of payoffs, not on their actual values. The model can also be applied to other 2x2 games, 2 player games with 2 possible moves with different payoff structures. A classification of 2x2 games according to their payoff matrices is provided by Rapoport et al. (1976). Finally, Lebiere et al. (2000) state that further research is needed, in order to explain the significance of model characteristics with respect to simulation results.

1.7 Model Characteristics and Assessment of Models

In this study, four ACT-R based memory models are implemented. Following ACT-R model implemented by Lebiere et al. (2000), models with similar procedural and memory functions are explored. Main distinction between our models and Lebiere et al. (2000) model is the architectural difference related to agent memories. In Lebiere et al. (2000) model, players use one declarative memory module together. However, in this study agents are implemented with separate declarative memory modules. Moreover, different decision making procedures are implemented in order to investigate their impact on the performance in Iterated Prisoner's Dilemma Game. The cognitive architectures, decision making and memory processes of different models are discussed in

the second chapter.

After the adjustment of parameters, models are evaluated in terms of their performances against basic Prisoner's Dilemma strategies. In order to evaluate models by testing several hypothesis related to behavioral and neuroscientific studies of Iterated Prisoner's Dilemma Game, we have conducted simulated experiments where the model plays against itself. However, the experiments between different models are not included in this work.

First, behavioral experiments show that [cooperate, cooperate] outcome is observed more frequently (% 55) than other outcomes, whereas [defect, defect] outcome has a smaller frequency (% 30). Other outcomes have very low frequency when compared to cooperative and defective outcomes, less than 10 % (Lebiere et al., 2000). Therefore, when testing the models, we expect two agents to succeed in exhibiting cooperative behavior where [cooperate, cooperate] outcome will exceed other outcomes in frequency.

Second, as the iterated game progresses, frequency of [cooperate, cooperate] outcome increases, whereas frequency of other outcomes decrease with time. Agents are expected to learn to overcome the temptation of unilateral defect behavior in order to achieve cooperative equilibrium.

Third, Wedekind & Milinski (1996) claim that human population can be divided in two classes in terms of strategies they employ while playing Iterated Prisoner's Dilemma. First group use a Pavlovian-type strategy, whereas second group employs a strategy similar to Forgiving Tit-for-Tat strategy. Behavior of different memory models can be analyzed in terms of conditional cooperation probabilities with respect to the outcome of the previous round. Although more detailed analysis of conditional cooperation probabilities with respect to the result of last two or three games, there are not any studies in the literature which will allow us to compare ACT-R models. After calculating conditional cooperation probabilities, we can classify agents as Pavlovian and Tit-for-Tat players by analyzing their behavior after [defect, cooperate] and [defect, defect] outcomes.

Forth, agents modify their strategy according to the strategy employed by their part-

ners. According to Baker & Rachlin (2002), human players adopt a learning strategy (Pavlovian) against a teaching strategy (TFT) and they use a teaching strategy (TFT) against a learning strategy (Pavlovian). Therefore, we can evaluate the model performance against Pavlovian and Tit-for-Tat players according to their success in adopting teaching and learning behavior.

Finally, it is expected that the models to successfully detect the unconditionality in behaviors of All-Cooperate, All-Defect and Random players.

2 MEMORY BASED ACT-R MODELING OF ITERATED PRISONER'S DILEMMA

Game environment for Iterated Prisoner's Dilemma, basic game strategies and four memory models are implemented with Python ACT-R programming language[2]. The structure of game environment and its interaction with models is explained below. Moreover this chapter presents behavioral characteristics, decision rules and cooperation probabilities for basic Iterated Prisoner's Dilemma strategies such as All-Cooperate, All-Defect, Random, Tit-for-Tat, Tit-for-Two-Tats, Forgiving-Tit-for-Tat and Pavlovian strategies. Then, decision algorithms and memory structure and memory processes are discussed.

Decision making in all models depends on the activation levels of memory chunks. Memory models predict the most likely outcome of different moves and select the move with the highest expected payoff. Each model keeps track of game history in terms of memory chunks. First memory model encodes game history in outcome chunks, whereas second model records outcome patterns observed in iterated game. Third model make use of both information encoding systems. Forth model is similar to first model and records only outcome chunks. However, retrieval of outcome chunks depends on the contextual information related to previous round provided in the goal buffer of the cognitive architecture. Detailed analysis of decision algorithms, memory processes and model parameters are presented in this chapter.

2.1 Iterated Prisoner's Dilemma Game and Game Environment

Iterated Prisoner's Dilemma game is played repeatedly by two players in a simultaneous fashion. Players infer about the actions of other players when the outcome of the previous round is observed by both players. The specific version of Prisoner's Dilemma used by Rapoport et al. (1976), Lebiere et al. (2000) and (Kim et al., 2004) is implemented in game environment using Python ACT-R programming language. Pay-

[2]Documentation, software and tutorials for Python ACT-R is available at http://terry.ccmlab.ca:8080/project/ccmsuite

off structure of this version of Prisoner's Dilemma is presented in Figure 10. Similar to these examples in literature, simulated models play the game for 300 rounds.

		Player 2	
		Cooperate	Defect
Player 1	Cooperate	1 , 1	-10 , 10
	Defect	10 , -10	-1 , -1

Figure 10: Payoff matrix for Prisoner's Dilemma Game used in ACT-R Model.

According to payoff matrix of the game, when both players cooperate mutually, both players receive one as payoff. However, when defect move is selected by both players, they receive -1. According to this payoff scheme, each [defect, defect] outcome offsets the gains of [cooperate, cooperate] outcome. When a cooperate move is matched by a defect move from other player, cooperating player receives -10, whereas defecting player gets 10 as payoff. According to this payoff scheme, when a non-symmetric outcome occurs, the sum of payoffs equals to zero. Moreover, the number of each [cooperate, defect] outcome offsets the payoffs of [defect, cooperate] outcome.

It can be concluded that when number of two different symmetric outcomes are equal and number of two different non-symmetric outcomes are equal, both players finish the game with zero payoff. When number of two types of symmetric outcomes are different, there are two scenarios. When cooperating outcome is larger in number than the defective outcome both players are better off compared to zero payoff case. When number of defective outcomes are higher than cooperative outcomes, both players incur losses. If number of [cooperate, defect] and [defect, cooperate] outcomes are different at the end of the game, due to the payoff structure of non-symmetric outcomes, player who defects more has positive results and other player who cooperates more in non-symmetric outcomes has negative payoffs.

Game environment for Prisoner's Dilemma serves as a medium which takes each player's choice, determines resulting outcome of each round, gives feedback to players about the outcome and pushes new goals of playing Prisoner's Dilemma until the last

round of the game. Functioning of the game environment is summarized below:

```
Set strategy of Player 1
Set strategy of Player 2
FOR round 1 to 300
    record each player's choice of action
    determine the outcome of the round, modify scores of players
    give feedback to players about the outcome
Determine winner at the end of the game and print results
```

Interaction between game environment and player strategies are illustrated in Figure 11. Game environment reads actions of each player and translates them into feedback about outcome of previous round. This feedback is used for decision making process in the next round. Each player modifies goal buffer and memory module according to this information provided by the game environment.

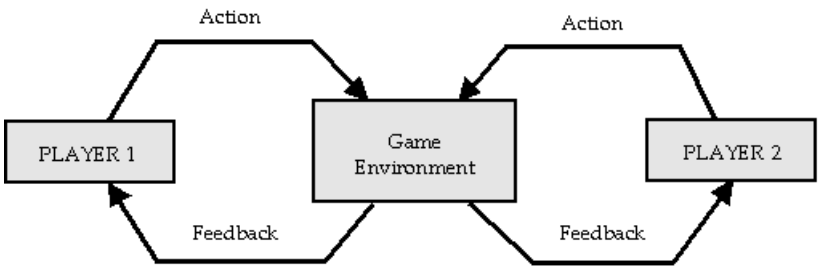


Figure 11: Interaction between Game Environment and Player Strategies.

Strategy of each player is selected before the start of 300 rounds. These strategies can be ACT-R models of Prisoner's Dilemma or other basic conditional, unconditional and random Prisoner's Dilemma strategies.

In addition to basic Iterated Prisoner's Dilemma strategies, we have implemented an application which enables human subjects to play the game with other models and basic strategies. Human Player application takes input from keyboard at each round and translates that input into actions in ACT-R game environment. This feature can be used in future experiments in order to evaluate performance of human subjects against ACT-R models, basic strategies or other human subjects.

2.2 Memory Model of Iterated Prisoner's Dilemma Based on Outcome History

First model, MODEL 1 is a memory-based model and it is implemented using Python ACT-R. This model is similar to the memory model of Iterated Prisoner's Dilemma game of Lebiere et al. (2000). Four memory chunks encode the Prisoner's Dilemma payoff structure into ACT-R memory items. These chunks are the basic building blocks of the memory model. Decision making process depends on the relative activation of different memory chunks.

Each memory chunk in the model has five slots, first slot represents the name of the chunk. Second slot is named as move1 and it refers to the player's own moves, third slot refers to the other player's actions. Forth and fifth slots represent player's own payoff and other player's payoff respectively, after the outcome represented by this chunk is observed. Four basic memory chunks in our model are presented below:

```
(C1-C2 move1:Cooperate move2:Cooperate payoff1:1 payoff2:1)
(C1-D2 move1:Cooperate move2:Defect payoff1:-10 payoff2:10)
(D1-C2 move1:Defect move2:Cooperate payoff1:10 payoff2:-10)
(D1-D2 move1:Defect move2:Defect payoff1:-1 payoff2:-1)
```

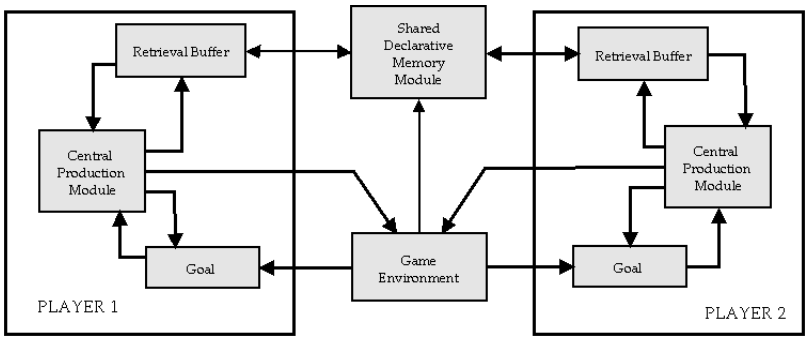
These chunks represent a specific Prisoner's Dilemma game with a certain payoff structure. We can adapt the memory models for other 2x2 games by modifying memory chunks according to their payoff matrices. These four memory chunks are created at the beginning of the game in each player's memory module. After creation, these memory chunks are reinforced for a certain number of times, in order to ensure a certain level of activation for each chunk at the beginning of the game. Each chunk is reinforced as they are recreated in the memory. ACT-R language automatically detects recreations and merges them with the original chunk and records each recreation as a reference for the original chunk. Initial reinforcement level is controlled by L parameter in our model. It represents the initial number of references to chunks and it is same for each chunk. L parameter determines the initial activation level for each chunk. Activation of each chunk changes during the course of the game according to the activation equation:

$$A = ln\sum_{j=1}^{n} t_j^{-d} + N(0, \frac{\pi.s}{\sqrt{3}})$$

First term in the activation equation demonstrates the reinforcement level. Number of references is represented by n, t_j is the time passed since jth reference is made. As game progresses and agents observe the outcome of the previous round, the chunk that represent this outcome is recreated and reinforced in their declarative memory. Thus, if an outcome is experienced more than others, chunk that refers to this outcome is reinforced more than others. Activation of each chunk also depends on decay rate which is represented by d in the equation. Decay rate is the level of forgetting in the declarative memory. If a certain chunk is not used, recalled or reinforced, its activation level decreases with time according to decay level. Therefore, it can be concluded that the activation of each chunk depends on the level of reinforcement and the recency of reinforcement. A memory chunk is also reinforced when it is recalled from declarative memory according to functioning of ACT-R cognitive architecture. This means that if a certain chunk is recalled during decision making process, it is reinforced as if it is experienced by the player. Memory model requests two outcomes from declarative memory during decision-making in each round. This process is summarized below:

```
IF the goal is to solve prisoner's dilemma
    the most likely outcome of making move C is payoff c
    the most likely outcome of making move D is payoff d
THEN make move with the highest payoff,
    record outcome and push a new goal for the next play
```

In conclusion, activation level of memory chunks depends on four things. First, the number of initial reinforcement which is controlled by parameter L increases activation. Second, the number of times an outcome is experienced as the game progresses and third, the number of times a chunk is recalled from memory during decision making increase reinforcement and activation level of the chunk. These three cases are controlled by the first equation which depends on the decay rate. Finally, activation of each chunk depends on the second term in the equation; noise level which is controlled by parameter s. This term adds stochasticity to activation equation, enables model to recall chunks with low activation levels and explore different action options.



Lebiere et al. (2000) Model

Figure 12: Basic architecture of Lebiere et al. (2000) Model.

In our memory model, each player has its own declarative memory module. This is the main difference between our model (Figure 13) and memory model which is implemented by Lebiere et al. (2000) (Figure 12). In Lebiere et al. (2000) model activation level of chunks are determined not only by the decision making process of the player, but by the decision making process of the other player as well. Apart from this structural difference, models are similar to each other in terms of decision-making structure and functioning of the declarative memory. Architectures of two models and their relationship with game environment are depicted in Figure 12 and Figure 13.

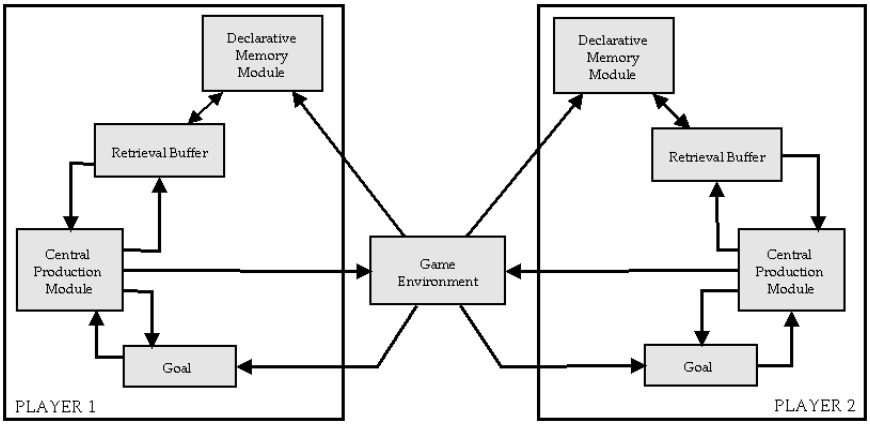


Figure 13: Basic architecture of MODEL 1, MODEL 2 and MODEL 3.

At every round of the Prisoner's Dilemma game, player evaluates the most likely result of each possible move. In order to find out the most likely outcome of making defect move, player requests a memory chunk which has defect value in its second slot:

```
(?  move1:Defect move2:?  payoff1:?x payoff2:?)
```

Declarative memory returns the chunk with the highest activation level amongst two possible chunks:

```
(D1-C2 move1:Defect move2:Cooperate payoff1:10 payoff2:-10)
(D1-D2 move1:Defect move2:Defect payoff1:-1 payoff2:-1)
```

Then player records the value of payoff1 slot, x as the value of making defect move to imaginal buffer. Then procedural module makes another request to declarative memory, in order to find out the most likely result of making cooperate move:

```
(?  move1:Cooperate move2:?  payoff1:?y payoff2:?)
```

From the two chunks which match this request, the one with the highest activation is recalled:

```
(C1-C2 move1:Cooperate move2:Cooperate payoff1:1 payoff2:1)
(C1-D2 move1:Cooperate move2:Defect payoff1:-10 payoff2:10)
```

The value of payoff1 slot, y is recorded from retrieval buffer to imaginal buffer as the value of cooperate move. After the evaluation of the most likely result of each move, player makes a comparison between expected payoffs of the two moves. If $x > y$ is true, player chooses to defect and cooperates if the opposite is true. After selecting the move with the highest expected payoff, player observes the outcome of this round and reinforces the chunk which corresponds to the observed outcome. In order to continue playing the game, player pushes a new goal from game environment. Decision making process for MODEL 1 is illustrated as a flowchart in Figure 14.

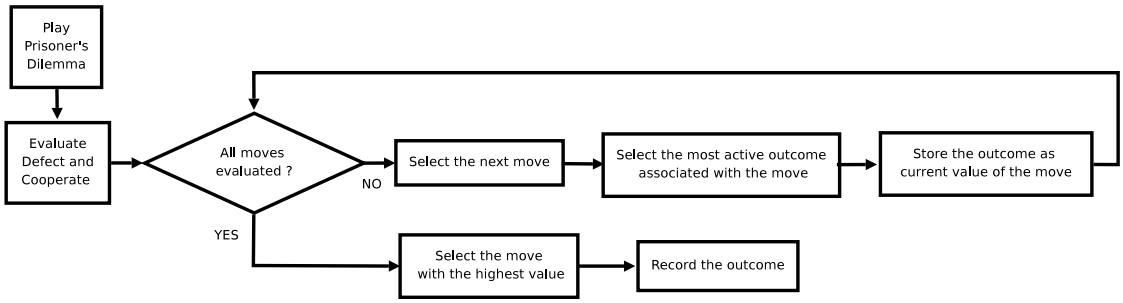


Figure 14: Flowchart of decision making in Prisoner's Dilemma Game used by the ACT-R MODEL 1.

Decision making precess in MODEL 1 depends on the retrieved chunks from the declarative memory. When a retrieval request is made to the declarative module, the chunk with the highest activation level is recalled. Change in activation level of chunks

at the first round of the game without forgetting ($d = 0$) and noise ($s = 0$) is depicted in Table 2. Activation level depends only on the initial references to the chunk A_{i}, in the first game. As game progresses, activation level of a recalled chunk increases and new activation level becomes A_{i}+A_{r}. After each player makes its move according to comparison of recalled chunks, they observe the outcome of this round. This observation elevates the activation level of observed chunk and its activation becomes A_{i}+A_{r}+A_o. As player starts the evaluation process in the next round, these changes in the activation level determines the retrieval of chunks, since declarative memory retrieves the chunk with the highest activation among the matching chunks.

Table 2: Change in Activation Levels of Chunks During First Round of the Decision Making Process.

Player 1					Player 2				
Chunks	[C, C]	[C, D]	[D, C]	[D, D]	Chunks	[C, C]	[C, D]	[D, C]	[D, D]
Initial Activation	A_{i}	A_{i}	A_{i}	A_{i}	Initial Activation	A_{i}	A_{i}	A_{i}	A_{i}
Recalled Chunk	[C, C]	-	-	[D, D]	Recalled Chunk	-	[C, D]	[D, C]	-
Activation Level	A_{i}+A_{r}	A_{i}	A_{i}	A_{i}+A_{r}	Activation Level	A_{i}	A_{i}+A_{r}	A_{i}+A_{r}	A_{i}
Selected Move	Cooperate				Selected Move	Defect			
Outcome	-	[C, D]	-	-	Outcome	-	-	[D, C]	-
Activation Level	A_{i}+A_{r}	A_{i}+A_{o}	A_{i}	A_{i}+A_{r}	Activation Level	A_{i}	A_{i}+A_{r}	A_{i}+A_{r}+A_{o}	A_{i}

At every round, memory model recalls the most active chunk associated with making defect move and the most active chunk associated with making cooperate move. Model makes its decision according to payoff scheme of the recalled chunks. When [cooperate, defect] chunk is recalled after the request for the result of cooperate action, irrespective of the type of the recalled chunk from defect request, player chooses to play defect move. Since compared to [cooperate, defect] outcome both [defect, cooperate] and [defect, defect] have higher payoffs. When [cooperate, cooperate] is retrieved from memory, decision of the player depends on the result of its request of the possible outcome of the defect move. If [defect, cooperate] is recalled, defect move is selected since [defect, cooperate] has the highest payoff in Prisoner's dilemma game. However, when [defect, defect] is selected, player chooses to cooperate, since [cooperate, cooperate] pays higher than [defect, defect] outcome. Consequently, cooperate

move is selected only in one of the four possible cases. This move selection process is illustrated in Figure 15.

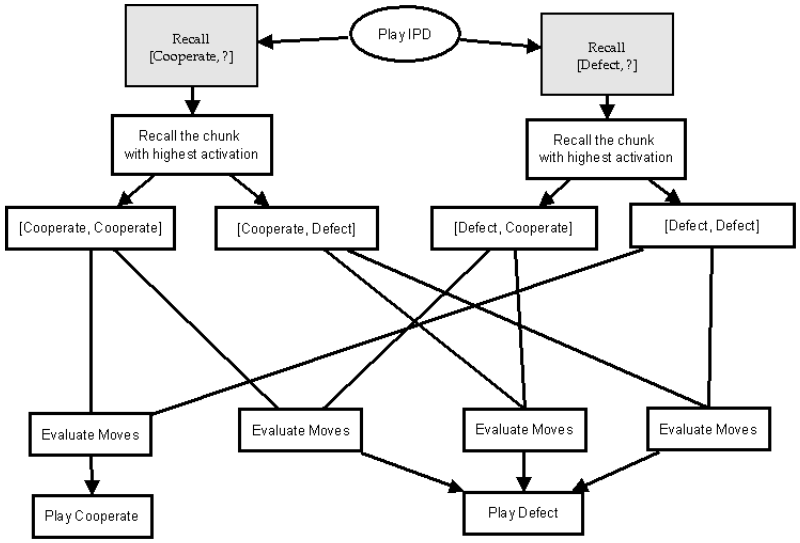


Figure 15: Decision making process for ACT-R MODEL 1.

After each player makes a move, game environment reads these moves and provide feedback for each player about the outcome of the Prisoner's Dilemma. Feedback about the outcome elevates the activation level of the chunk which represents that outcome. At each round, player's actions determine the outcome of the round, this process is demonstrated in Figure 16.

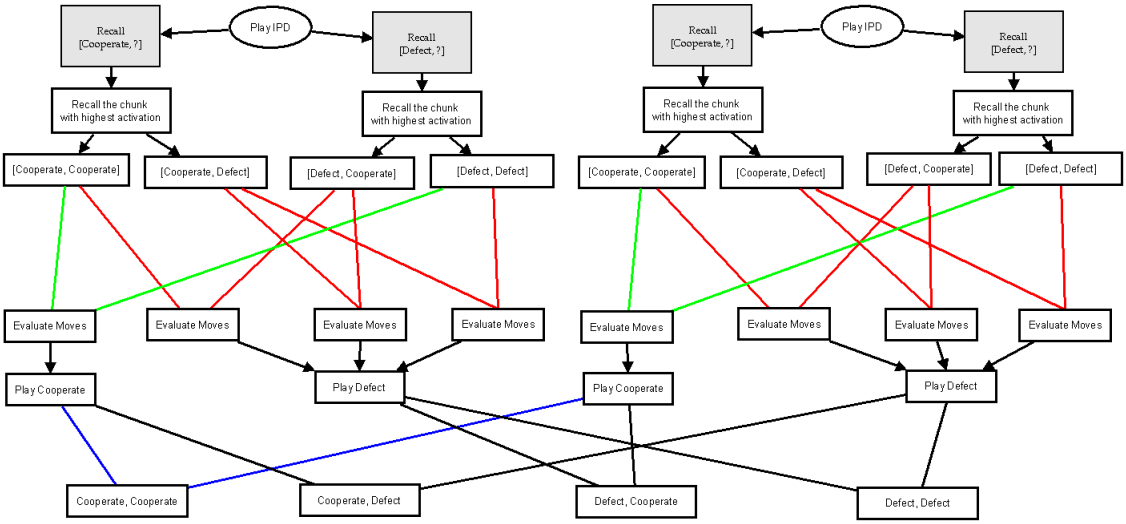


Figure 16: Outline of decision-making and resulting outcomes at a single round of Prisoner's Dilemma Game.

In the first round of the game, activation levels of the all four chunks are equal.

When MODEL 1 requests the chunk which represents the most likely outcome of making cooperate move, declarative memory returns one of the two matching chunks with equal probability, either [cooperate, defect] or [cooperate, defect]. Model records payoff of the chunk as the value of cooperate move. When this process is repeated for the defect move, memory module recalls either [defect, cooperate] or [defect, defect] chunks with equal likelihood. Therefore, in the first round retrieval probability of each chunk is equal to 1/2. Player chooses to play cooperate only when both [cooperate, cooperate] and [defect, defect] outcome chunks are retrieved from the declarative memory module. Therefore, cooperate move is selected by each player with probability of 1/4 and players choose to defect with 3/4 probability. According to this probability structure, [cooperate, cooperate] outcome is observed with a probability of 1/16 in the first round of the game. Probability of recall, move selection and outcomes at the first round is summarized in Table 3.

Table 3: Retrieval Probability for Chunks, Probability of Moves and Resulting Outcomes at the First Round of the Game.

First Round of Prisoner's Dilemma								
Players	Player 1				Player 2			
Chunks	CC	CD	DC	DD	CC	CD	DC	DD
Initial Activation	A_i	A_i	A_i	A_i	A_i	A_i	A_i	A_i
Probability of Recall	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2
Request	C?		D?		C?		D?	
Recalled Chunks: [C?, D?]								
Recalled Chunks	CC, DD	CC, DC	CD, DD	CC, DC	CC, DD	CC, DC	CD, DD	CC, DC
Probability of Recall	1/4	1/4	1/4	1/4	1/4	1/4	1/4	1/4
Selected Move	Cooperate	Defect			Cooperate	Defect		
Probability Of Move	1/4	3/4			1/4	3/4		
Observed Outcome: [Player1, Player2]								
Outcome	CC		CD		DC		DD	
Probability of Outcome	1/16		3/16		3/16		9/16	

After the first round of the game, activation levels of chunks are modified according to base level activation equation. In consecutive rounds, retrieval probability changes according to outcome history and recall history of the outcome chunks. If activation learning in the model is deactivated, activation levels of the outcome chunks are not modified and they are equal to initial activation level throughout the game. Therefore,

recall probability of all four chunks and likelihood of outcomes are equal throughout the game. Figure 17 shows the distribution of outcomes over 1000 runs of 300 round - Iterated Prisoner's Dilemma game without base level activation learning. Results are compatible with probability structure which is presented in Table 3.

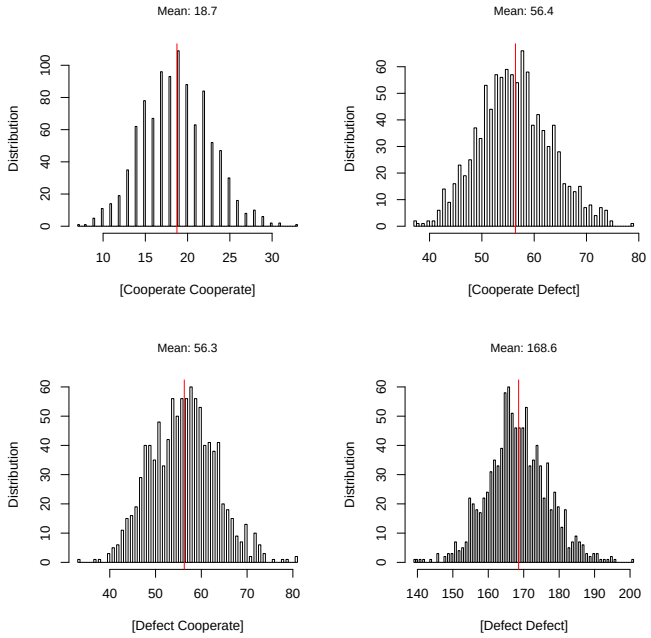


Figure 17: Distribution of outcomes over 1000 runs for Iterated Prisoner's Dilemma game when ACT-R Model without activation learning plays against itself.

In short, MODEL 1 is a memory-based learning and decision making model of Iterated Prisoner's Dilemma Game. Model uses its declarative memory for keeping track of outcome chunks which represents four possible outcomes of the game and their payoff structures. Activation level of four outcome chunks are updated repeatedly in the course of the game. Activation levels are elevated when an outcome is experienced or recalled from memory. Otherwise, activation levels of all chunks are subject to decay according to a certain forgetting rate. Memory model retrieves a particular chunk from the memory only if its activation is the highest among all matching chunks which meets the requirements of the memory request. According to the functioning of ACT-R declarative memory system, player recalls memory items that are recently observed or used by the player and all memory items suffer from the inherent forgetting mechanism in the memory module. In the decision-making process, players try to predict the most

likely outcome of making two possible moves, cooperate or defect, and they select the move with the highest possible expected payoff. Decision process depends mainly on the activation equation and outcome history for MODEL 1.

2.3 Memory Model of Iterated Prisoner's Dilemma Based on Outcome Patterns

Second memory model, MODEL 2 keeps track of outcome patterns that are observed in the course of the game and model evaluates the result of making a possible move according to these outcome patterns. Player selects the move with the highest expected payoff. After each player makes a move, MODEL 2 records the outcome of the last two rounds as an outcome pattern. In other words, each player records the outcome of the last round not as a single entity but in relation to the result of the round previous to that one, thus we can conclude that this player extracts a different type of information from game history. Apart from this basic difference in the manner of tracking game history, MODEL 2 is identical to first memory model in its decision-making and memory processes, these processes are explained in this section.

In our memory model, each player has its own declarative memory module. Architecture of players and game environment is illustrated in Figure 13. Similar to first model, MODEL 2 represents the Prisoner's Dilemma payoff matrix with four outcome chunks. Each chunk encodes the name of the chunk, moves of the players and corresponding payoff structure. These four basic memory chunks are presented below:

```
(C1-C2 move1:Cooperate move2:Cooperate payoff1:1 payoff2:1)
(C1-D2 move1:Cooperate move2:Defect payoff1:-10 payoff2:10)
(D1-C2 move1:Defect move2:Cooperate payoff1:10 payoff2:-10)
(D1-D2 move1:Defect move2:Defect payoff1:-1 payoff2:-1)
```

These chunks are created solely for the purpose of game representation and they are not functional in the decision making process. Players use the outcome chunks in order to determine payoffs of certain outcomes. Decision making process depends on the outcome patterns recorded by the player after each round of the game. Players keep outcome patterns in memory chunks, an illustratory chunk is presented below:

```
(pattern move1:Cooperate move2:Cooperate pmove1:Defect pmove2:Cooperate)
```

First slot of the outcome pattern chunk states that the chunk is a pattern chunk, second and third slot represents the moves of first and second player at the last round respectively. Forth and fifth slots refer to the round before the last round. According to this specific chunk, player records that after the player selected defect when the other player cooperated in the second last round, both players chose to cooperate at the previous round. Second and forth slots refer to player's own actions in the previous round and the round before that. Third and fifth slots represents the actions of the other player at the last two rounds.

Player keeps the outcome of the previous round in its goal module, goal chunk is presented below:

```
(play pd defect cooperate)
```

This specific goal chunk states that the goal of the player is playing Prisoner's Dilemma game and the player played defect, whereas the other player chose to cooperate in the previous round.

At each round of the Prisoner's Dilemma game, player evaluates the most likely result of each possible move. In order to find out the most likely outcome of making defect move, player requests the pattern memory chunk which has defect value in its second slot and outcome of the last round is represented in its forth and fifth slots. When the outcome of the last round is [defect, cooperate], player requests the chunk with the following attributes:

```
(pattern move1:Defect move2:?md pmove1:Defect pmove2:Cooperate)
```

Declarative memory returns the pattern chunk with the highest activation level between the two possible chunks:

```
(pattern move1:Defect move2:Defect pmove1:Defect pmove2:Cooperate)
(pattern move1:Defect move2:Cooperate pmove1:Defect pmove2:Cooperate)
```

Therefore, value of *md* variable is either cooperate or defect. After recalling this value, player requests the payoff associated with this specific outcome:

```
(?  move1:Defect move2:?md payoff1:?x payoff2:?)
```

There is only one matching chunk for this memory request. If the value of *md* is cooperate, $x = 10$ and when the value of md is defect, $x = -1$ according to the payoff

matrix of the game. Then player records the value of payoff1 slot, x as the value of making the defect move to imaginal buffer. Then the module makes another request to declarative memory, in order to find out the most likely result of making cooperate move when the outcome of the previous round is [defect, cooperate]:

```
(pattern move1:Cooperate move2:?mc pmove1:Defect pmove2:Cooperate)
```

Declarative memory returns the pattern chunk with the highest activation level between the two possible chunks:

```
(pattern move1:Cooperate move2:Defect pmove1:Defect pmove2:Cooperate)
(pattern move1:Cooperate move2:Cooperate pmove1:Defect pmove2:Cooperate)
```

Retrieval process determines the value of *mc* variable, it is either cooperate or defect. After the value is recalled from the declarative memory, then the player requests the payoff associated with this specific outcome:

```
(?  move1:Cooperate move2:?mc payoff1:?y payoff2:?)
```

There is only one matching chunk for this memory request. If the value of *mc* is cooperate, $y = 1$. On the other hand, y is equal to -10, when the value of *mc* is defect. These values are determined according to the payoff matrix of the Prisoner's Dilemma game. Value of payoff1 slot, y is recorded from retrieval buffer to imaginal buffer as the value of cooperate move. After evaluating the most likely outcome of each move and payoffs associated with these moves, player makes a decision. If the value of x is higher than y, player chooses to defect and cooperates if $y > x$ is true. After selecting the move with the highest expected payoff, player observes the outcome of this round and reinforces the pattern chunk which represents this outcome in relation to the outcome of the previous round. If a pattern chunk which represents the experienced pattern of last two round does not exist, player creates a new chunk. Next step is pushing a new goal from game environment. Decision making process of MODEL 2 is depicted as a flowchart in Figure 18.

According to MODEL 2, decision making process is based on the differences in activation levels of pattern chunks. When the player requests an outcome pattern from declarative memory given the outcome of the last round, declarative memory retrieves the pattern chunk with the highest activation level. If there are no matching patterns in

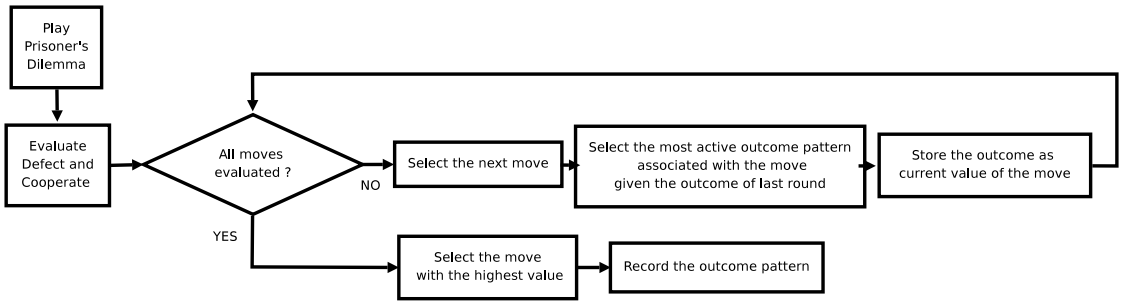


Figure 18: Flowchart of decision making in Prisoner's Dilemma Game used by the ACT-R MODEL 2.

the declarative memory, declarative memory declares a memory failure. This situation is observed especially in the initial stages of the iterative game, since outcome space and pattern space are not fully explored by the players. When players encounter with memory failures, they make a random choice between defect and cooperate.

Memory chunks which represent the outcome patterns do not exist before the start of the game. Whenever a player experiences a new outcome pattern, a corresponding pattern chunk is created in the declarative memory. Each pattern chunk is reinforced whenever it is experienced in the game history and whenever it is recalled from declarative memory module. Activation level and probability of retrieval for each pattern chunk depend on the activation equation. Activation equation is identical to the activation equation of the other memory models:

$$A = ln\sum_{j=1}^{n} t_j^{-d} + N(0, \frac{\pi.s}{\sqrt{3}})$$

L parameter which determines the initial number of reinforcements is not important in this model, since pattern chunks are not created before the start of the iterated game. Only outcome chunks are created before the first round and their initial reinforcement is determined according to L parameter. Decision making process in this model makes use of only the pattern chunks and action selection does not depend on the activation level of the outcome chunks. Therefore, it can be concluded that L parameter is not an effective parameter for this model. Other parameters, d which refers to decay rate and s parameter which determines the noise level are crucial for the decision making process. Activation levels of the pattern chunks depend on the decay parameter which controls the forgetting rate of the memory chunks with respect to time past. Activation level of

a pattern chunk depends on the noise parameter too. Noise in the activation equation allows pattern chunks with low activation levels to be retrieved from declarative memory. Memory model requests two pattern chunks from the declarative memory model in order to evaluate two possible moves. After the model determines the resulting payoff structure associated with the most likely outcome of making cooperate and defect moves, model selects the move with the highest payoff. Outline of the decision making process is presented below:

```
   IF the goal is to solve prisoner's dilemma and outcome of the last round
is O
       the most likely outcome of making move C is payoff c given O
       the most likely outcome of making move D is payoff d given O
   THEN make move with the highest payoff,
       record outcome and outcomes of last two rounds as a pattern,
   and push a new goal for the next play
```

At every round, memory model recalls the most active pattern chunk associated with making defect move and the most active pattern chunk associated with making cooperate move given the outcome of the previous round. Then player evaluates cooperate and defect moves according to recalled pattern chunks. Memory model makes its decision according to payoff scheme associated with each move. Similar to first model, cooperate move is selected only in one of the four possible cases. This decision making process is illustrated in Figure 19.

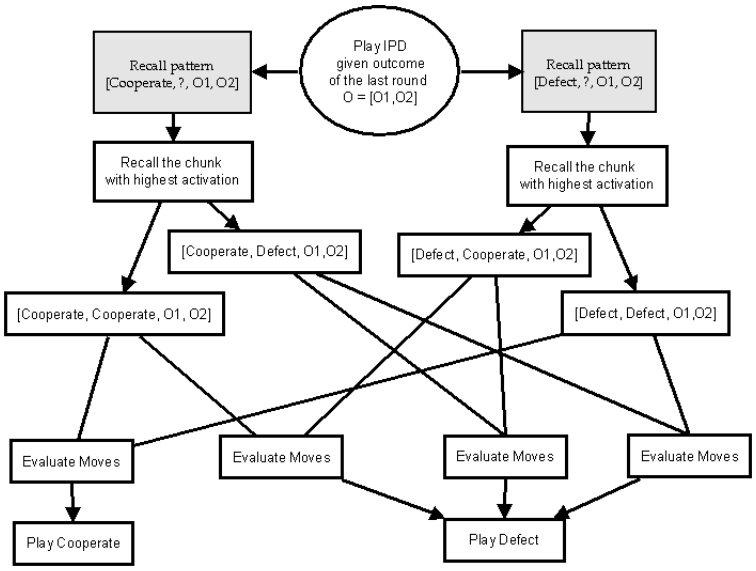


Figure 19: Decision making process for ACT-R MODEL 2.

In conclusion, MODEL 2 is an ACT-R memory model based on outcome patterns. Memory model employs outcome pattern chunks in order to record the outcome history of the Iterated Prisoner's Dilemma Game. Outcomes of the last two rounds are encoded in a single outcome pattern chunk. Activation levels of pattern chunks depends on the forgetting rate, noise level and reinforcement of the chunk. An outcome pattern chunk is reinforced when it is experienced by the player or it is retrieved from memory module. After observing the result of the last round, player tries to recall the most likely outcome of making defect and cooperates moves. Declarative memory module retrieves the pattern chunk with the highest activation level. Then the player evaluates the possible moves according to associated payoffs and selects the move with the highest payoff. MODEL 2 makes decisions according to outcome patterns and activation levels of memory items determined by the activation equation.

2.4 Predictive Memory Model of Iterated Prisoner's Dilemma Based on Outcome History and Outcome Patterns

Predictive memory model is a combination of first and second models. Third memory model records outcome history in two different formats. Outcome history is recorded as outcome chunks and outcome pattern chunks. MODEL 3 extracts two different types of information from game history: First, player collects information about the frequency and recency of different outcomes. Second, model also records the outcome patterns after each round.

In order to evaluate different moves, MODEL 3 first recalls the most active outcome chunk associated with the move and records the payoff as the present value of the move. Then player goes one step further in the evaluation of the chunk and investigates the consequences of making the move. In this next step, model requests the most active pattern chunk in order to determine the resulting future outcome. Player tries to predict the future outcome after the most active outcome associated with the move is observed. Player records the payoff of this outcome as future value of the move. After each possible move is evaluated, player adds present and future payoff values of the

moves according to certain weights. Player selects the move with the highest expected payoff. After each player makes a move, MODEL 3 records the result of the round as an outcome chunk and the outcomes of the last two rounds as an outcome pattern. Therefore, this model resembles first two models in terms of information extraction from game history. However, this player collects more information compared to the first two memory models. Apart from these differences, decision making, learning and functioning of the memory is similar to first two models.

In predictive memory model, cognitive architecture of players and and their relation to game environment is identical two first two memory models (Figure 13). Predictive model encodes the specific Prisoner's Dilemma game in terms of four basic outcome chunks. These chunks represents four possible outcomes and payoff matrix of the Prisoner's Dilemma game:

```
(C1-C2 move1:Cooperate move2:Cooperate payoff1:1 payoff2:1)
(C1-D2 move1:Cooperate move2:Defect payoff1:-10 payoff2:10)
(D1-C2 move1:Defect move2:Cooperate payoff1:10 payoff2:-10)
(D1-D2 move1:Defect move2:Defect payoff1:-1 payoff2:-1)
```

Player recreates the chunk which represent the resulting outcome at every round of the game. Outcome chunks are also recalled during decision making process in order to find out the consequence of a making a specific move. Since retrieval of outcome chunks depends on the activation level, initial activation level is important for the model. Initial activation levels are determined by the L parameter which controls the number of references made to every outcome chunk before the start of the game. These four outcome chunks created before the start according to L parameter.

Pattern chunks, on the other hand, are created according to the feedback from game environment about the resulting outcome of the previous round. Decision making process depends on both outcome chunks and pattern chunks recorded by the player after each round of the game. An exemplary pattern chunk is presented below:

```
(pattern move1:Cooperate move2:Defect pmove1:Defect pmove2:Cooperate)
```

This chunk states that the player chose to cooperate after defecting in the previous round. However, other player shifted to defect move as its cooperation is not recip-

rocated in the previous round. Second and third slots keep the information about last round and the last two slots represent the outcome of the round before the last one.

At the first step of the decision making process, player requests the most active chunk associated with a certain move. As the model tries to predict the most likely outcome of making defect move, an outcome chunk request is made to the declarative memory module in the following format:

```
(?  move1:Defect move2:?  payoff1:?x1 payoff2:?)
```

Declarative memory returns the chunk with the highest activation level amongst two possible chunks:

```
(D1-C2 move1:Defect move2:Cooperate payoff1:10 payoff2:-10)
(D1-D2 move1:Defect move2:Defect payoff1:-1 payoff2:-1)
```

If it is assumed that [defect, defect] is recalled from memory module, retrieval buffer contains the following chunk:

```
(D1-D2 move1:Defect move2:Defect payoff1:-1 payoff2:-1)
```

Player records the value of $x1$ into the imaginal buffer as the present value of defect move. Next step is to determine other consequences of selecting this move. In the next step, player recalls the pattern chunk, in order to predict future outcome of the game after [defect, defect] is observed:

```
(pattern move1:?m1 move2:?m2 pmove1:Defect pmove2:Defect)
```

Declarative memory returns the pattern chunk with the highest activation level among the four possible chunks:

```
(pattern move1:Defect move2:Defect pmove1:Defect pmove2:Defect)
(pattern move1:Defect move2:Cooperate2 pmove1:Defect pmove2:Defect)
(pattern move1:Cooperate move2:Defect pmove1:Defect pmove2:Defect)
(pattern move1:Cooperate move2:Cooperate pmove1:Defect pmove2:Defect)
```

If the declarative memory returns the following pattern chunk:

```
(pattern move1:m1 move2:m2 pmove1:Defect pmove2:Defect)
```

Then the player records the payoff value, $x2$ associated with the outcome as the future payoff of the defect move :

```
(?  move1:m1 move2:m2 payoff1:?x2 payoff2:?)
```

After the evaluation of defect move, model starts to investigate the results of choos-

ing cooperate in the next round. To find out the most likely outcome of making cooperate move, the following request is made to the declarative memory module:

```
(?  move1:Cooperate move2:?  payoff1:?y1 payoff2:?)
```

Among the two chunks which match this request, the one with highest activation is recalled:

```
(C1-C2 move1:Cooperate move2:Cooperate payoff1:1 payoff2:1)
(C1-D2 move1:Cooperate move2:Defect payoff1:-10 payoff2:10)
```

If [cooperate, defect] outcome is more likely, second chunk is recalled from declarative memory:

```
(C1-D2 move1:Cooperate move2:Defect payoff1:-10 payoff2:10)
```

Present value of cooperate move, $y1$ is recorded to the imaginal buffer. Then memory model tries to find out the outcome of the next run after [cooperate, defect] outcome is observed. Following memory chunk is requested:

```
(pattern move1:?m1 move2:?m2 pmove1:Cooperate pmove2:Defect)
```

Declarative memory returns the pattern chunk with the highest activation level among four possible chunks:

```
(pattern move1:Defect move2:Defect pmove1:Cooperate pmove2:Defect)
(pattern move1:Defect move2:Cooperate2 pmove1:Cooperate pmove2:Defect)
(pattern move1:Cooperate move2:Defect pmove1:Cooperate pmove2:Defect)
(pattern move1:Cooperate move2:Cooperate pmove1:Cooperate pmove2:Defect)
```

If declarative memory returns the following pattern chunk:

```
(pattern move1:m1 move2:m2 pmove1:Cooperate pmove2:Defect)
```

Then the player records $y2$, the payoff value associated with this outcome as the future payoff of the cooperate move :

```
(?  move1:m1 move2:m2 payoff1:?y2 payoff2:?)
```

The value of payoff1 slot, $y2$ is recorded from retrieval buffer to imaginal buffer as the future value of cooperate move. After recording the present and future payoffs of different moves, the player compare the payoff scheme of defect and cooperate moves. The player adds the payoffs associated with a certain move according to a w parameter defined by the programmer. The w parameter α, determines the strength of the future payoff when the player is calculating expected payoff of a specific move.

Total payoff for defect move is calculated as $x1 + \alpha \cdot x2$, whereas total payoff is equal to $y1 + \alpha \cdot y2$ for cooperation. After evaluating the payoff structure associated with each move, player takes a decision according to following rule . If the value of $x1 + \alpha \cdot x2$ is higher than $y1 + \alpha \cdot y2$, player chooses to defect and cooperates only if the opposite is true. After selecting the move with the highest expected payoff, player observes the outcome of this round and reinforces both the outcome chunk and the pattern chunk which represents this outcome in relation to the outcome of the previous round. If a pattern chunk which represents the experienced pattern of last two rounds does not exist, player creates a new chunk. Model also pushes a new goal according to the feedback from game environment. Decision making process of MODEL 3 is depicted as a flowchart in Figure 20.

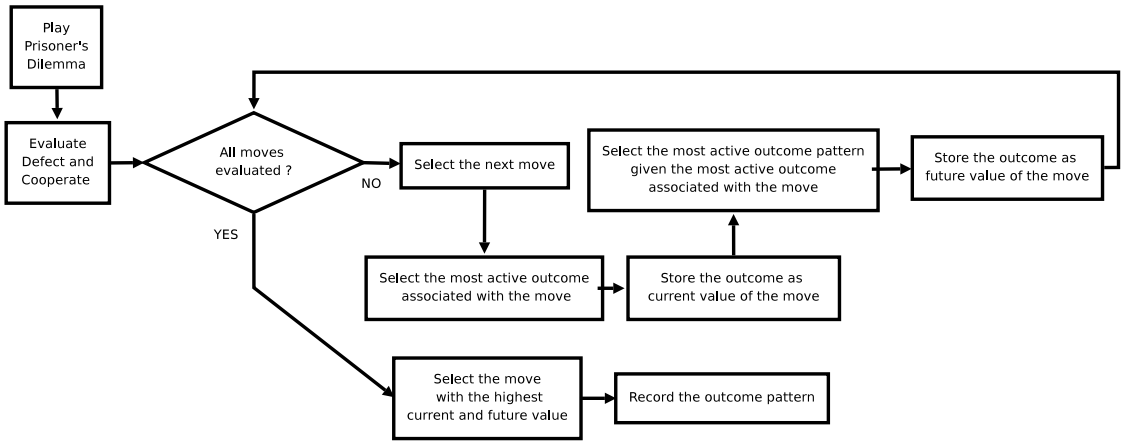


Figure 20: Flowchart of decision making in Prisoner's Dilemma Game used by the ACT-R MODEL 3.

Decision making process for MODEL 3 depends on two different types of information about frequency and recency of outcomes and outcome patterns. In order to evaluate different moves, player requests both outcome chunks and outcome pattern chunks in different steps of the decision making process. Declarative memory module recalls the chunks with the highest activation levels. Outcome chunks exist before the start of the iterated game. Therefore, player does not encounter any memory failures during the recall of outcome chunks. In contrast, pattern chunks are created whenever the player encounters a new outcome pattern. Thus, the pattern chunk request may cause a memory failure, if there are no matching pattern chunks in the declara-

tive memory. In this exceptional case, the player requests the outcome chunk with the highest activation level. Decision making process is designed to handle this situation according to the assumption of the outcome with the highest activation is more likely occur in the future. Memory failures are more likely to occur in the initial stages, since there are more points in the pattern space yet to be experienced in the initial rounds.

After every round of the game, player reinforces the chunk representing the outcome of the round. If the player experiences a new outcome pattern, a corresponding pattern chunk is created in the declarative memory. Each pattern chunk and outcome chunk is reinforced whenever it is experienced in the game history and whenever it is recalled from declarative memory module. Activation equation of the model determines the activation levels and retrieval probability of the chunks. Activation equation is identical to the activation equation of the other memory models:

$$A = ln\sum_{j=1}^{n} t_j^{-d} + N(0, \frac{\pi . s}{\sqrt{3}})$$

Noise and decay parameters are crucial when determining the activation levels of all chunks. Decay parameter, d controls the forgetting rate for the memory items. When decay rate is higher, player focuses more to the recent past. In other words, time span for the impact of a reinforcement is shorter, when decay rate is higher. Noise parameter, s allows the player to recall memory items with lower activation levels. Unlike MODEL 2, L parameter is influential in decision processes. It determines the initial number of reinforcements and affects the activation levels of outcome chunks. MODEL 3 uses outcome chunks in the decision making process, therefore L parameter is important in this model. Memory model starts decision making by requesting two outcome chunks from the declarative memory model in order to evaluate defect and cooperative moves. After recording the payoffs of these two outcomes, player requests two pattern chunks in order to determine the most likely future outcomes of making these moves. Then the model determines the resulting payoffs associated with the most likely outcomes as the future payoff of cooperate and defect moves. Move with the highest present and future payoffs is selected by the model. Outline of the decision making process is presented below:

```
   IF the goal is to solve prisoner's dilemma and outcome of the last round
is O
       the most likely outcome of making move C is payoff c
       the most likely future outcome of making move C has payoff x
       the most likely outcome of making move D is payoff d
       the most likely future outcome of making move D has payoff y
   THEN make move with the highest present payoff,
       record outcome and outcomes of last two rounds as a pattern, and push
a new goal for the next play
```

To sum up, MODEL 3 is an ACT-R memory model based on both outcome history and outcome patterns. Memory model employs outcome and pattern chunks in order to extract different types of information from game history. Outcome chunks are created before the start of the game. Outcome of each round reinforces the memory chunk representing this outcome. Activation levels of outcome chunks are determined by decay rate, noise level, initial number of reinforcements and outcome history. Outcomes of the last two rounds are encoded in a single outcome pattern chunk after each round.

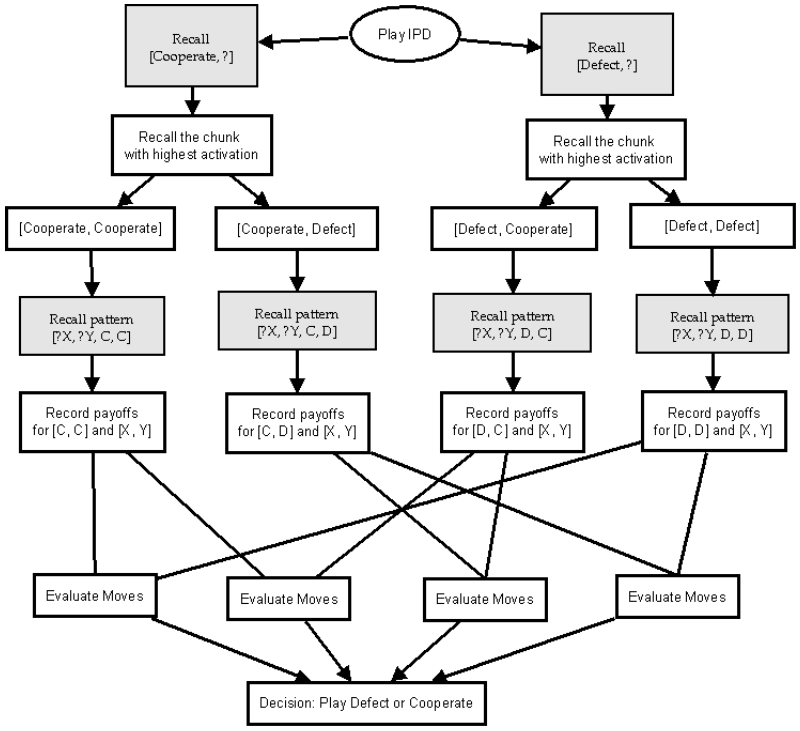


Figure 21: Decision making process for ACT-R MODEL 3.

Activation level of a pattern chunk depends on the forgetting rate, noise level and reinforcement of the chunk. An outcome pattern chunk or an outcome chunk is reinforced when it is experienced by the player or it is retrieved from memory module. Af-

ter observing the result of the last round, player tries to recall the most likely outcome of making defect and cooperates moves. In the next step, player tries to determine the most likely future outcome to be observed after making this move by retrieving the most active pattern chunk. Defect and cooperate moves are evaluated in terms of their present and future payoffs, player selects the move with the highest total payoff. Decision making for MODEL 3 is a two step process which is illustrated in Figure 21. Decision making for MODEL 3 depends on outcome history, outcome patterns and activation levels of memory items determined by the activation equation.

2.5 Associative Memory Model of Iterated Prisoner's Dilemma Based on Outcome History

Fourth memory model, MODEL 4 is similar to first model in its functioning. Model tries to predict the most likely outcome of making a specific move, records the payoff of the outcome. Then the player selects the move with the highest payoff. After receiving feedback about the outcome, the chunk which represent this outcome is reinforced. Activation levels of outcome chunks increase when they are recalled from memory or they are experienced in the course of the game. In contrast to MODEL 1, there is an additional factor which affects the activation level of outcomes. Association factor keeps track of the association between goal buffer and retrieval buffer. Whenever a certain chunk is retrieved from declarative memory, associative strength between the goal chunk and retrieved chunk increases. Therefore, retrieval probability of this specific chunk increases when we have the same goal chunk in the goal buffer. Apart from memory association mechanism, decision making process is similar to first model.

In our memory model, each player has its own declarative memory module. Architecture of players and game environment is illustrated in Figure 22. Similar to first model, MODEL 4 represents the Prisoner's Dilemma payoff matrix with four outcome chunks:

```
(C1-C2 move1:Cooperate move2:Cooperate payoff1:1 payoff2:1)
(C1-D2 move1:Cooperate move2:Defect payoff1:-10 payoff2:10)
(D1-C2 move1:Defect move2:Cooperate payoff1:10 payoff2:-10)
```

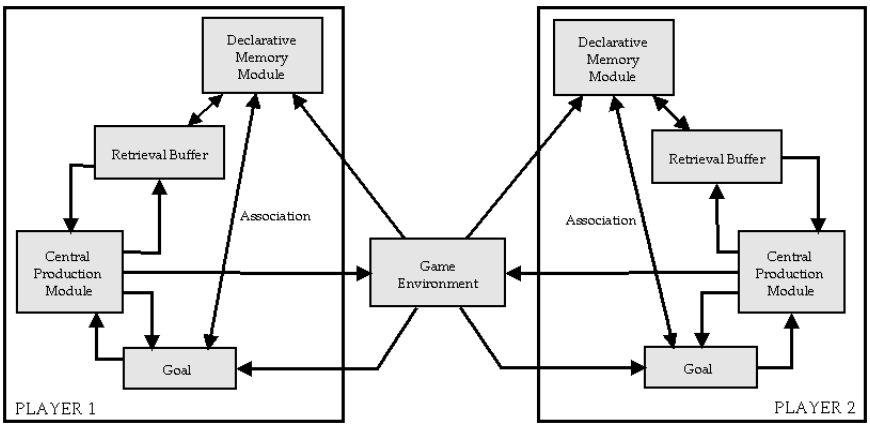


Figure 22: Basic architecture of MODEL 4.

```
(D1-D2 move1:Defect move2:Defect payoff1:-1 payoff2:-1)
```

Player keeps the outcome of the previous round in its goal module, a goal chunk is presented below:

```
(play pd defect defect)
```

Goal chunk functions as a context for retrieval of memory chunks.

At every round of the Prisoner's Dilemma game, player evaluates the most likely result of each possible move. In order to find out the the most likely outcome of making defect move, player requests the memory chunk which has defect value in its second slot:

```
(?  move1:Defect move2:?  payoff1:?x payoff2:?)
```

Declarative memory returns the chunk with the highest activation level amongst the two possible chunks:

```
(D1-C2 move1:Defect move2:Cooperate payoff1:10 payoff2:-10)
(D1-D2 move1:Defect move2:Defect payoff1:-1 payoff2:-1)
```

Then player records the value of payoff1 slot, x as the value of making defect move to imaginal buffer. Then the model makes another request to declarative memory, in order to find out the most likely result of making cooperate move:

```
(?  move1:Cooperate move2:?  payoff1:?y payoff2:?)
```

From the two chunks which match this request, the one with highest activation is recalled:

```
(C1-C2 move1:Cooperate move2:Cooperate payoff1:1 payoff2:1)
```

```
(C1-D2 move1:Cooperate move2:Defect payoff1:-10 payoff2:10)
```

Value of payoff1 slot, y is recorded from retrieval buffer to imaginal buffer as the value of cooperate move. After the evaluation of the most likely result of each move, player makes a comparison between expected payoffs of the two moves. If $x > y$ is true, player chooses to defect and cooperates if the opposite is true. After selecting the move with the highest expected payoff, player observes the outcome of this round and reinforces the chunk which corresponds to the observed outcome. Decision making process of MODEL 4 is illustrated as a flowchart in Figure 23.

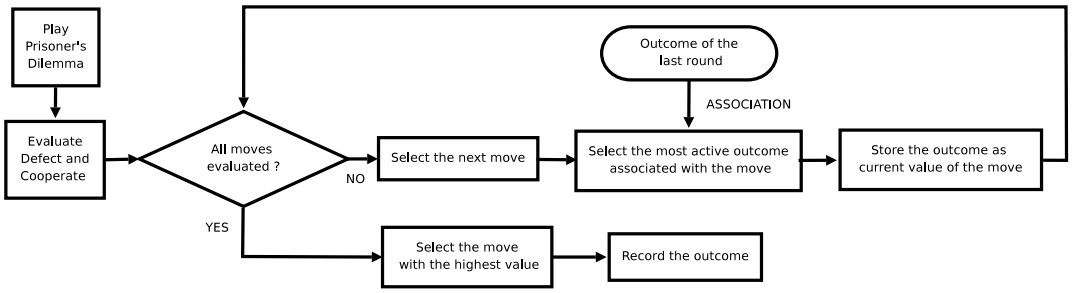


Figure 23: Payoff matrix for Prisoner's Dilemma Game used in ACT-R Model.

In MODEL 4, decision making process is based on the activation differences between outcome chunks. Activation of a certain outcome chunk increases whenever it is experienced in the course of the game or it is retrieved from memory during decision making. Activation level and probability of retrieval for each pattern chunk depend on the activation equation. Activation equation is presented below:

$$A = ln\sum_{j=1}^{n} t_j^{-d} + N(0, \frac{\pi . s}{\sqrt{3}}) + w \cdot S$$

L parameter in our model determines the initial number of reinforcements for the outcome chunks, and it is effective in our model. Parameter d controls the decay rate of activation level and s parameter determines the noise level in the activation equation. weight parameter, w controls the weight of associative strength in the activation equation. S refers to the associative strength between the chunk in the goal buffer and outcome chunks in the declarative memory. Associative strength increases between a goal and memory chunk if the memory chunk is retrieved when that specific goal chunk

is present in the goal buffer. Therefore, simultaneous occurrences increases the associative strength between a memory chunk and a goal chunk. During decision making model retrieves the most likely outcome of making a specific move from declarative module. Then the model compares the resulting payoffs of making defect and cooperate moves, selects the move with the highest payoff. Outline of the decision making process is presented below:

```
IF the goal is to solve prisoner's dilemma when goal is G
    the most likely outcome of making move C is payoff c
    the most likely outcome of making move D is payoff c
THEN make move with the highest payoff,
    record outcome and push a new goal for the next play
```

At every round, memory model recalls the most active chunk associated with making defect move and the most active chunk associated with making cooperate move. Then the player makes its decision according to payoff scheme of the recalled chunks. Similar to first model, cooperate move is selected only in one of the four possible cases. This decision making process is depicted in Figure 24.

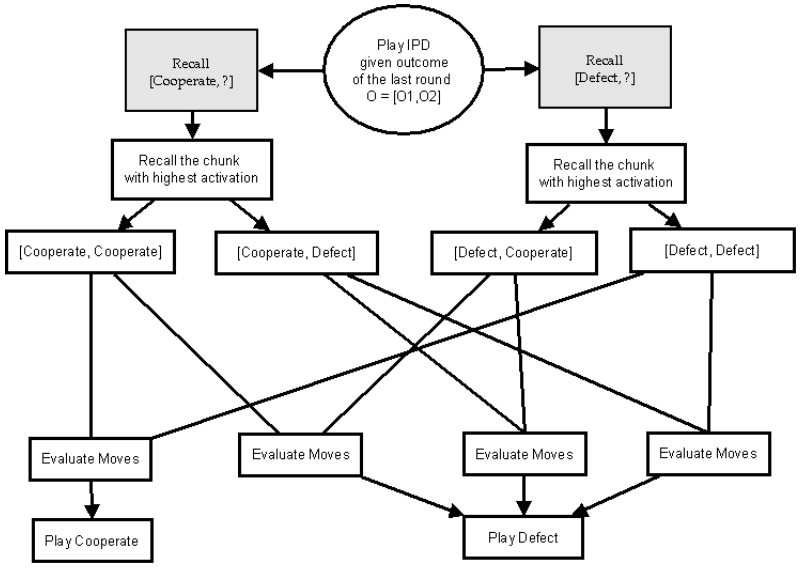


Figure 24: Decision making process for ACT-R MODEL 2.

In conclusion, MODEL 4 is an ACT-R memory model based on outcome history. Memory model uses outcome chunks to encode game history. Decision making process depends on the activation levels of outcome chunks. Initial number of references, noise level, decay rate and w of associative strength are the parameters that control

activation equation. Activation level of an outcome chunk changes in the course of game according to reinforcement via experience and retrieval. Another factor which affects the activation level is associative strength between goal chunks and outcome chunks. Associative strength also depends on the game history. Similar to other models, MODEL 4 evaluates and selects defect and cooperate moves according to most likely payoff of making those moves. It can be concluded that decision making in MODEL 4 depends on the activation equation and outcome history.

2.6 Summary

Implementation of four ACT-R based memory models are discussed in this chapter. All models have similar cognitive architectures which consist of procedural, declarative and goal modules. Each model evaluates the most likely outcome of making defect and cooperate moves according to retrieval of memory chunks associated with each move. Retrieval depends on the activation levels of memory chunks. First model records game history in terms of outcome chunks, model reinforces a memory chunk whenever it is experienced in the iterated game or it is recalled from declarative memory. Activation level of memory chunks depends on decay rate which controls forgetting rate of memory items, number of initial references to the memory chunk and noise level in the activation equation.

Table 4: Model Parameters.

Parameter	MODEL 1	MODEL 2	MODEL 3	MODEL 4
Decay Rate	d	d	d	d
Initial References	L	-	L	L
Noise Level	s	s	s	s
w of Future Payoffs	-	-	α	-
Associative Strength	-	-	-	w

MODEL 2 encodes game history in terms of outcome patterns. After each round, model creates a pattern chunk which records the outcomes of the last two rounds. Memory chunks are created after the start of the game, therefore number of initial references is not an important parameter for the model. Decay rate of memory chunks and noise level in the activation equation are effective in determining activation levels

of pattern chunks. Second model recalls the most active pattern chunks associated with each move given the result of the last round. Similar to first model, model chooses the move with the highest expected payoff.

Third model extracts two different information from game history. After each round, model records the outcome in terms of outcome chunks and outcome pattern chunks. Model employs outcome chunks in order to determine the most likely outcome of making a specific move. Model uses pattern chunks to predict the outcome of the round after next round. Therefore, there are two payoff values, present and future payoffs associated with each possible move. Model calculates an expected payoff according to a certain parameter, α which determines the weight of future payoffs and selects the move with the highest payoff value. Moreover, decay rate, initial number of references to outcome chunks and noise level in the activation equation are crucial parameters for the decision making process.

MODEL 4 is similar to first model in terms of memory processes and decision making algorithm. Model encodes outcome history as outcome chunks. Activation level of outcome chunks depend on the decay rate, noise level and initial number of references to outcome chunks. In addition to these, activation level of an outcome chunk is determined by the association level between outcome chunk and goal chunk which keeps the outcome of the last round. An association is formed between a specific goal chunk and memory chunk whenever they occur simultaneously in goal and retrieval buffers respectively. A specific goal chunk increases the retrieval probability of an outcome chunk if an association between these chunks is formed and reinforced in game history.weight parameter, w controls the weight of associative strength in the activation equation. Apart from association between goal and memory buffers, decision making and memory processes are similar to first model.

3 RESULTS AND DISCUSSION OF ACT-R MEMORY MODELS OF ITERATED PRISONER'S DILEMMA

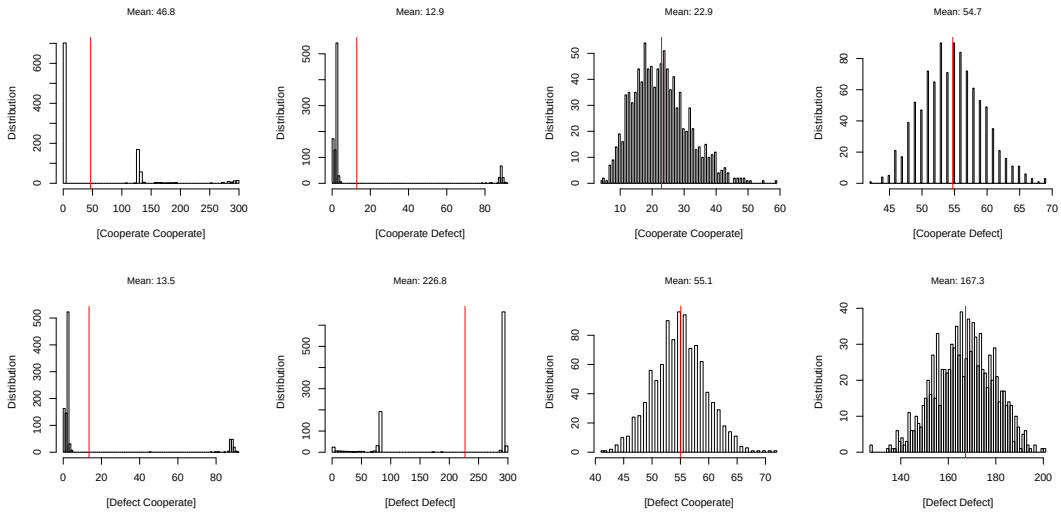
Performance of memory models are evaluated according to simulated experiments. Each model plays Iterated Prisoner's Dilemma Game with basic strategies such as All-Cooperate, All-Defect, Random, Tit-for-Tat, Tit-for-Two-Tats, Forgiving-Tit-for-Tat and Pavlovian strategies. In addition to that, each strategy plays the iterated game against itself. However, this study does not investigate the case where models play the iterated game against each other. Results are based on 1000 simulations of 300 rounds of iterated game between two players.

Model performance is analyzed in terms of distribution of four basic outcomes in 1000 simulations. Moreover, behavioral characteristics of models are evaluated in terms of means and distributions of conditional cooperation probabilities. Learning patterns for different models are illustrated by plots of outcome frequencies with respect to course of the game. In order to provide a detailed analysis of some behavioral patterns, outcome history of certain exemplary games are also illustrated. Performance of models against basic strategies and themselves can also be observed via plots which depict the score of players in each simulated game (Appendix D).

3.1 Memory Model of Iterated Prisoner's Dilemma Based on Outcome History

First memory model, MODEL 1 keeps track of the outcome of each round using outcome memory chunks. There are four outcome chunks which encodes the payoff matrix of Prisoner's Dilemma game. Using the information about frequency and recency of outcomes, player predicts the most likely result of making a specific move. After evaluating two possible moves, player decides to make the move with the highest payoff. After the outcome of the round is realized, declarative memory is updated accordingly.

Behavior of the model is controlled by three parameters in the model. First parameter is L, it refers to the number of initial references to each of the four outcome chunks created before the start of the rounds. L parameter affects the base level acti-

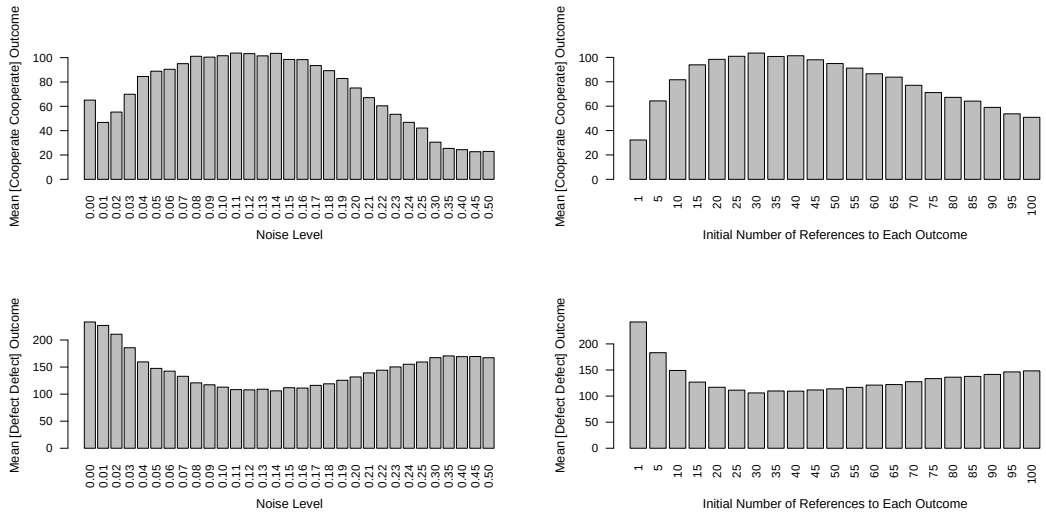


(a) (b)

Figure 25: Distribution and mean of outcomes when MODEL 1 plays against MODEL 1 and (a) decay = 0.5, noise = 0.01, L=30 (b) decay = 0.5, noise = 0.5, L=30.

vation of outcome chunks. Noise level in the activation equation is controlled by the *s* parameter. Noise in the model allows outcome chunks with low activation levels to be retrieved from memory module. Third parameter is *d*, decay rate controls the level of forgetting for memory chunks. Since all three parameters are effective in determining the model performance, we have searched the parameter space for combinations of parameters where cooperation rate is higher. Various combinations of parameters where *L* parameter varies between 5 to 50 with increments of 5, *s* parameter from 0 to 0.2 with increments of 0.01 and *d* parameter between 0.4 and 0.9 with 0.1 increments are examined in terms of model behavior. In many ACT-R models, decay parameter is chosen as 0.5 (Lebiere et al., 2000), therefore we will present results of the case when decay para mater is equal to 0.5. *L* and *s* parameters are chosen to give the highest cooperative outcome when d= 0.5. In addition to that, we will present the effects of increasing decay rate to 0.8 while keeping *s* and *L* constant. Certain combinations of parameters fail to produce meaningful behavioral results.

When noise is level is very low, MODEL 1 players either achieve very high cooperation rates or very high defective outcomes. The frequency of mixed outcomes and cooperative outcomes are low in 1000 rounds. Distribution and mean of four possible outcomes is presented in Figure 25. Although the low frequency of asymmetric out-



(a) (b)

Figure 26: Mean of [cooperate, cooperate] and [defect, defect] outcomes when MODEL 1 plays against MODEL 1 and decay = 0.5 (a) L = 30 constant, noise is variable (b) noise= 0.14 constant, L is variable.

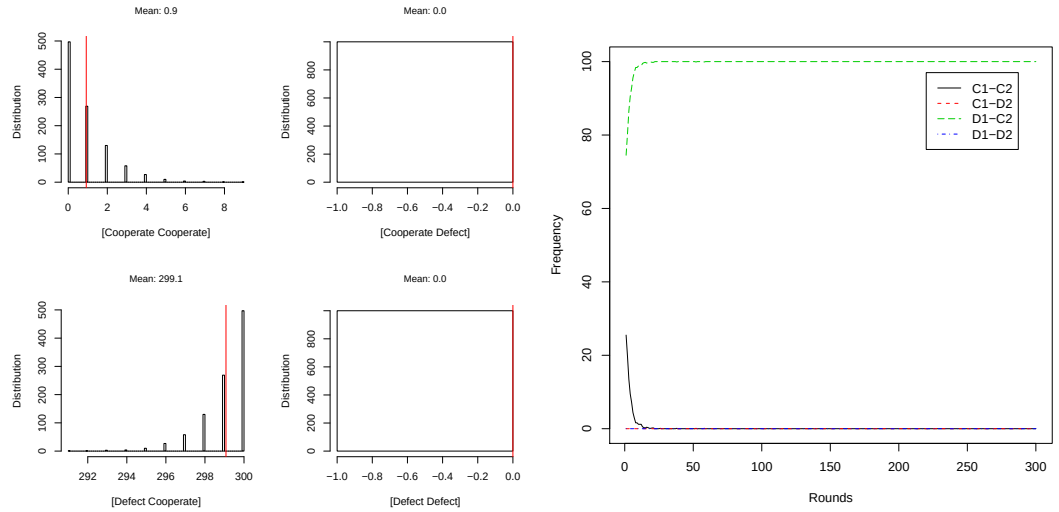
comes is favorable, frequency of cooperation is also low when compared to behavior of human subjects. Variance of outcome distribution is low, players are either fully cooperative or fully defective in most of the rounds when noise level is low.

In the case that noise level is very high, mean of cooperative outcomes is very low. MODEL 1 players fail to achieve high cooperation rates in all rounds. Frequency of asymmetric outcomes is higher compared to the case which the noise level is low. This combination of variables fails to achieve results similar to human subjects. Distribution of outcomes and means of different outcomes are presented in Figure 25.

Higher rates of cooperation is achieved when the noise level is medium level. Mean of [cooperate, cooperate] outcome is plotted with respect noise level when the L is constant and with respect to L parameter when the noise level is constant. These plots show that highest cooperation rate is achieved at $s = 0.14$ and $L = 30$ (Figure 26). However, it can not be claimed that these points indicate the optimum combination, since the variable space is not extensively explored. Variable combination which achieves the highest cooperation rate is used in order to investigate the model performance against basic game strategies.

3.1.1 Model Behavior against Basic Strategies

In this section, performance of MODEL 1 against basic strategies is evaluated. All-Cooperate and All-Defect strategies are basic unconditional strategies, when employed players either cooperate or defect in all rounds.

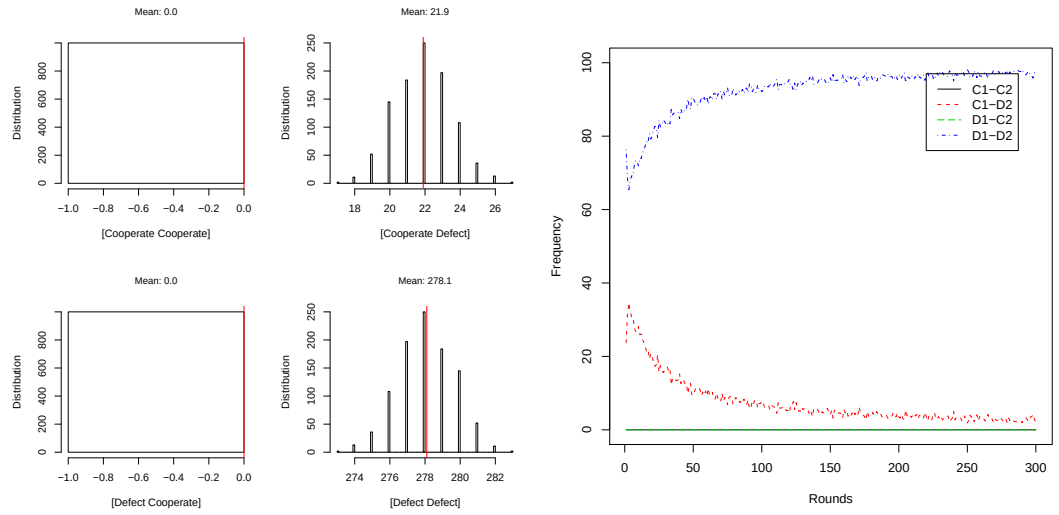


(a) (b)

Figure 27: Distribution of outcomes (a) and frequency of outcomes as game progresses (b) when MODEL 1 plays against ALLC strategy, decay = 0.5, noise = 0.14, L=30.

When MODEL 1 Player plays against All- Cooperate strategy, it detects the unconditionality in the other player's response and shift to defect move immediately in order to exploit this unconditional behavior. Although we observe [cooperate, cooperate] outcomes in the early stages of the game, model learns to defect and [defect, cooperate] outcome becomes prevalent. Since the payoff of this outcome is the highest, player continues to defect in the remaining rounds of the game without changing the behavioral pattern (Figure 27).

All-Defect strategy selects defect move at every round, irrespective of the other player's actions. When playing against ALL-D strategy, MODEL 1 learns defect in order to defend itself. Cooperating against defect strategy is very costly for players, thus model chooses to defect in most of the rounds. Since [defect, defect] outcome is also disadvantageous, the player explores the cooperate option unsuccessfully in some rounds. Then the player shifts to defect move immediately. According to this behavior



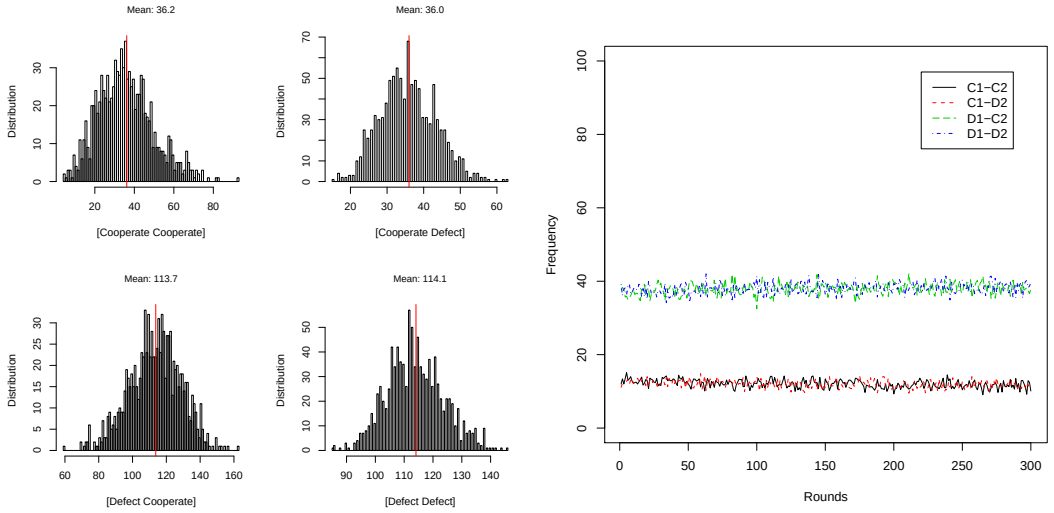
(a) (b)

Figure 28: Distribution of outcomes (a) and frequency of outcomes as game progresses (b) when MODEL 1 plays against ALLD strategy, decay = 0.5, noise = 0.14, L=30.

pattern, [defect, defect] is the dominant outcome in 300 rounds, but [cooperate, defect] outcome is also observed with a small frequency (Figure 28).

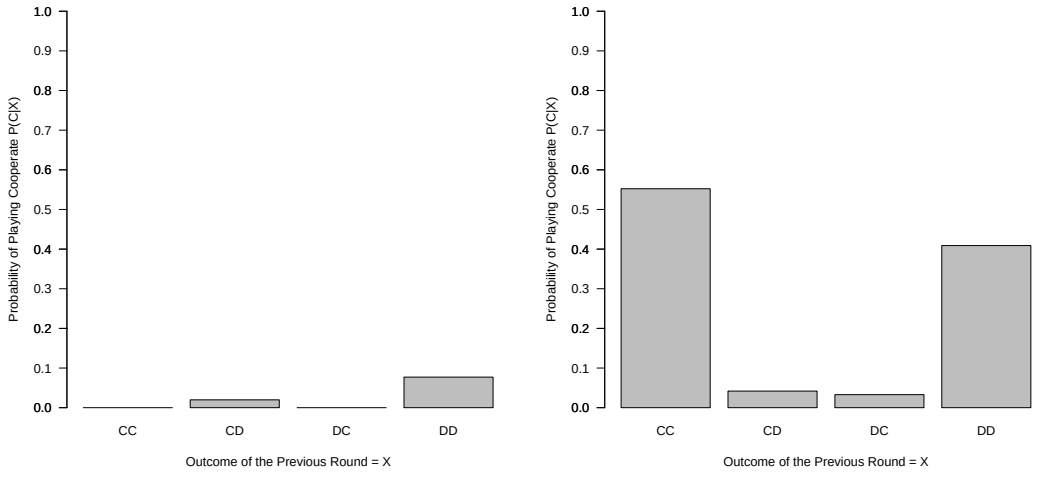
Random strategy plays defect and cooperate with equal probabilities. Choosing to defect against random strategy is much more advantageous than cooperation. By cooperating, player is more likely to incur losses since half of its cooperating moves are answered by defect behavior of Random strategy. Losses due to [cooperate, defect] outcome is higher than the gains of [cooperate, cooperate] outcome. When the player chooses to defect more frequently than cooperate, the chances of attaining a positive score is more likely for the player. Although half of the defect moves are matched by defect move of the Random strategy, gains through [defect, cooperate] surpasses the losses of [defect, defect] outcomes. When playing against Random strategy, MODEL 1 defects more frequently throughout the game and attain a higher score at the end of the game (Figure 29).

Mean of cooperation probabilities conditional to the outcome of the previous round when MODEL 1 is playing against All-Defect and Random strategies are presented in Figure 30. Against All-Defect strategy, cooperation probability of MODEL 1 is close to zero irrespective of the outcome of the previous round. When playing with Random strategy, player follows a Pavlovian like strategy, defects with a probability close to one



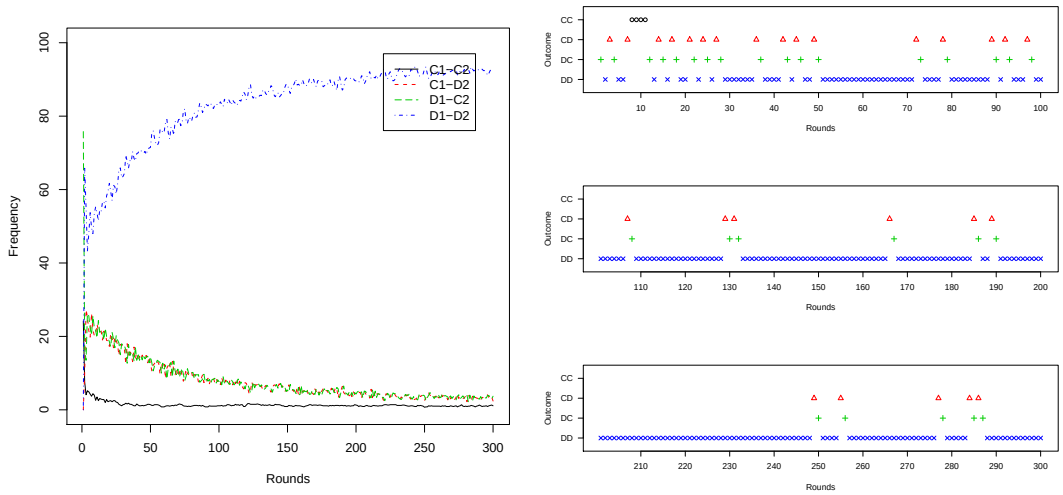
(a) (b)

Figure 29: Distribution of outcomes (a) and frequency of outcomes as game progresses (b) when MODEL 1 plays against Random (RAN) strategy, decay = 0.5, noise = 0.14, L=30.



(a) (b)

Figure 30: Mean of cooperation probabilities given the outcome of previous round when MODEL 1 plays against (a) All-Defect (ALLD) strategy (b) Random (RAN) strategy, decay = 0.5, noise = 0.14, L=30.



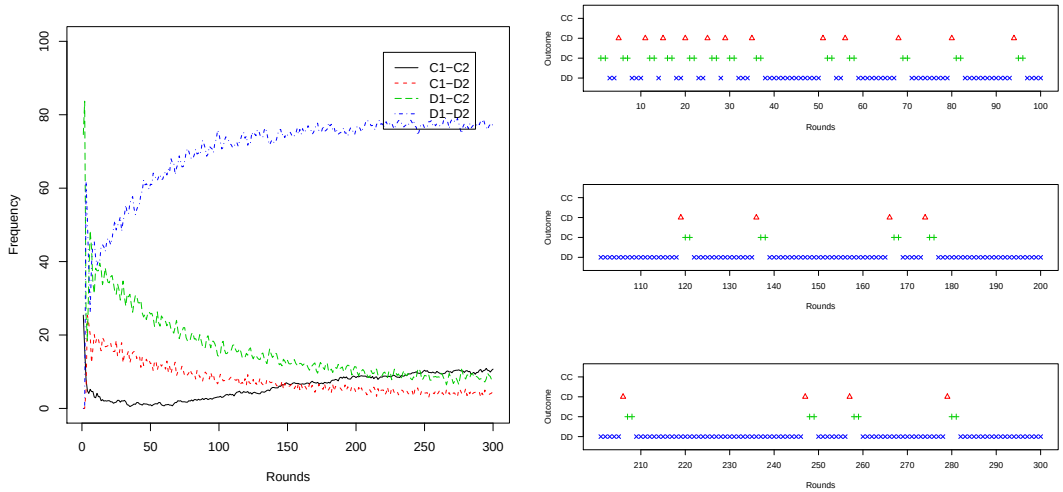
(a) (b)

Figure 31: Frequency of outcomes as game progresses (a) and outcome history of an exemplary game (b) when MODEL 1 plays against Tit-for-Tat (TFT) strategy, decay = 0.5, noise = 0.14, L=30.

after asymmetric outcomes, and cooperates half of the time after a symmetric outcome is observed.

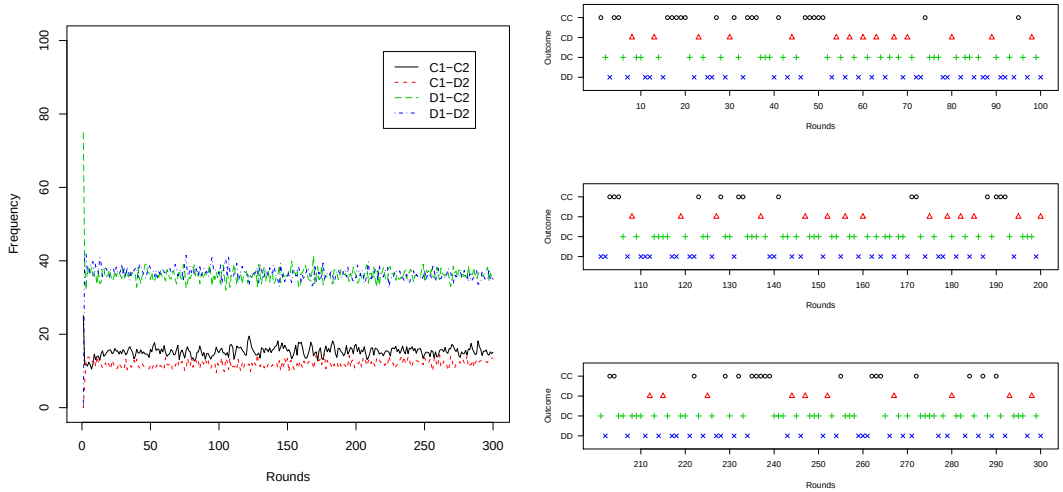
MODEL 1 fails to achieve cooperative equilibrium against teaching Tit-for-Tat player, whenever the model shifts to cooperate from defecting, Tit-for-Tat player answers this action by defecting, but reciprocates cooperation in the next round. Since, MODEL 1 observes disadvantageous [cooperate, defect] outcome, it immediately shifts back to defect move. Hence, players do not succeed in playing the cooperate move simultaneously and can not attain coordination when cooperating. This behavioral pattern is illustrated in Figure 31 (b) which exhibits outcome history of an exemplary game between TFT and MODEL 1 players. Consequently, when playing against each other, Tit-for-Tat strategy and MODEL 1 fails to accomplish cooperation in almost all iterative games (Figure 31 (a)).

When playing against Tit-for-Two-Tats strategy, MODEL 1 exhibits a similar pattern to the TFT case. Main distinction between two cases presents itself in the frequency of asymmetric outcomes. Frequency of asymmetric outcomes are equal when MODEL 1 plays against TFT strategy. However, frequency of [defect, cooperate] is higher than [cooperate, defect] when the model is playing against TFTT strategy.



(a) (b)

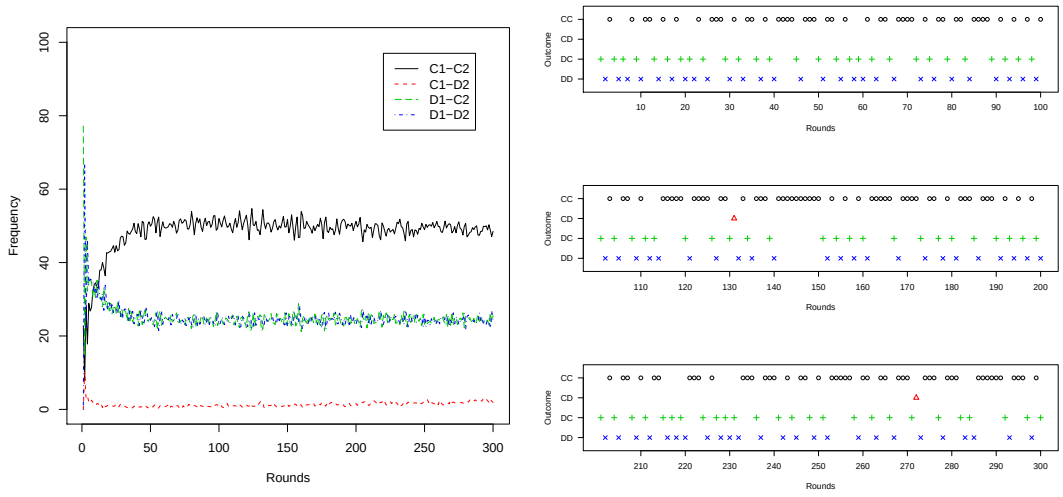
Figure 32: Frequency of outcomes as game progresses (a) and outcome history of an exemplary game (b) when MODEL 1 plays against Tit-for-Two-Tats (TFTT) strategy, decay = 0.5, noise = 0.14, L=30.



(a) (b)

Figure 33: Frequency of outcomes as game progresses (a) and outcome history of an exemplary game (b) when MODEL 1 plays against Forgiving Tit-for-Tat (TFTF) strategy, decay = 0.5, noise = 0.14, L=30.

MODEL 1 takes advantage of the reluctance of TFTT player when shifting to defection. Model achieves a higher score compared to TFTT player due to the higher frequency of the advantageous [defect, cooperate] outcome.

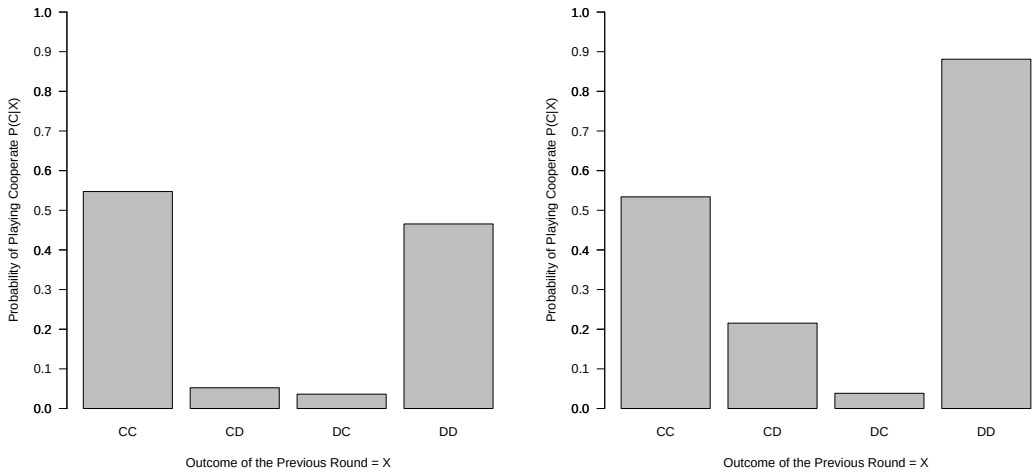


(a) (b)

Figure 34: Frequency of outcomes as game progresses (a) and outcome history (b) when MODEL 1 plays against Pavlovian (PAV) strategy, decay = 0.5, noise = 0.14, L=30.

A common trend observed in these two different cases is the decline in the frequency of asymmetric outcomes as game progresses. Since [cooperate, cooperate] outcome is rarely observed, its activation level decreases in the later stages of the game compared to [cooperate, defect] outcome. Latter becomes the most likely outcome of playing cooperate. As a result, player chooses to defect more frequently, since playing defect becomes more advantageous than playing cooperate. In the second half of the game, frequency of [cooperate, cooperate] outcome slightly increases when MODEL 1 plays against TFTT strategy. However this trend is absent in the case where decay rate is increased to 0.8 (Appendix F).

Forgiving Tit-for-Tat (TFTF) strategy obeys Tit-for-Tat rules but forgives defective moves with a probability of 1/3. When playing against TFTF strategy, MODEL 1 player plays defect move with a higher frequency and achieves a higher score at the end of the game. Throughout the iterated game, frequencies of [defect, cooperate] and [defect, defect] outcomes are almost forty percent and significantly higher than



(a) (b)

Figure 35: Mean of cooperation probabilities given the outcome of previous round when MODEL 1 plays against (a) Forgiving Tit-for-Tat (TFTF) strategy (b) Pavlovian (PAV) strategy, decay = 0.5, noise = 0.14, L=30.

the frequencies of [cooperate, cooperate] and [cooperate, defect] outcomes which are around fifteen percent (Figure 33 (a)). Although high frequency of [defect, defect] outcome is disadvantageous for both players, MODEL 1 manages to attain a higher score due to the high frequency of [defect, cooperate] outcome.

In contrast to the teaching strategies discussed above, Pavlovian strategies is a learning strategy (Camerer, 2003) that cooperates after symmetric outcomes and defects after asymmetric outcomes. MODEL 1 is very successful against Pavlovian strategy, frequency of cooperative outcome is higher than defective outcome and the most disadvantageous outcome for the model is not observed throughout the iterated game (Figure 34 (a)). Although behavior of MODEL 1 depends only on the game history and activation levels of memory chunks, MODEL 1 appears to employ a specific strategy against Pavlovian player.

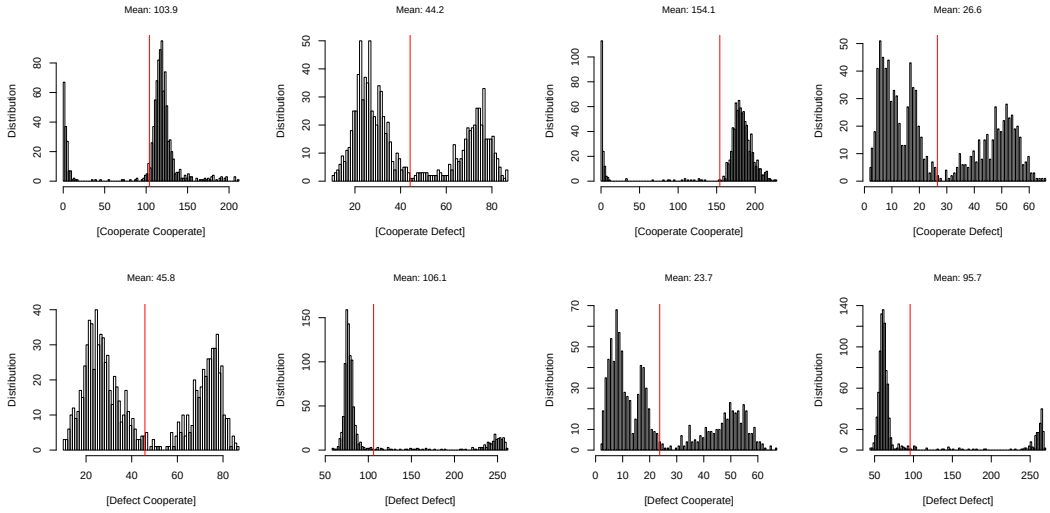
Behavior of MODEL 1 player against Pavlovian strategy is presented in Figure 34 (b). After cooperative outcome is observed, Pavlovian player continues to cooperate. Therefore, both defect and cooperate moves are advantageous for MODEL 1. If model chooses to defect and [defect, cooperate] is observed, in the next round player continues to defect and Pavlovian player shifts to defect move. As a result, [defect, de-

fect] is observed, this leads to change of behavior for both players. Thus, cooperative outcome, [cooperate, cooperate] follows defective outcome. The disadvantageous [cooperate, defect] outcome is almost never observed, since Pavlovian strategy continues to cooperate after cooperative outcome, MODEL 1 player on the other hand may shift to defect in some cases. Model shifts to defect when the activation level of [defect, defect] decreases due to decay rate and [defect, cooperate] is retrieved as the most likely outcome of making defect move.

As means of conditional cooperation probabilities are analyzed we observe that MODEL 1 employs a Pavlovian type strategy against both Tit-for-Tat strategies and Pavlovian strategy. When playing with Forgiving Tit-for-Tat strategy, MODEL 1 defects after asymmetric outcomes. After symmetric outcomes, the model cooperates with a 1/2 probability (Figure 35 (a)). Baker & Rachlin (2002) claim that adopting a teaching strategy against a learning strategy is useful in order to reach cooperative equilibrium. MODEL 1 exercise a learning Pavlovian type strategy against Pavlovian strategy. However model is more forgiving in the Pavlovian case and cooperates with a higher probability after observing the defect move of the other player in the previous round (Figure 35 (b)).

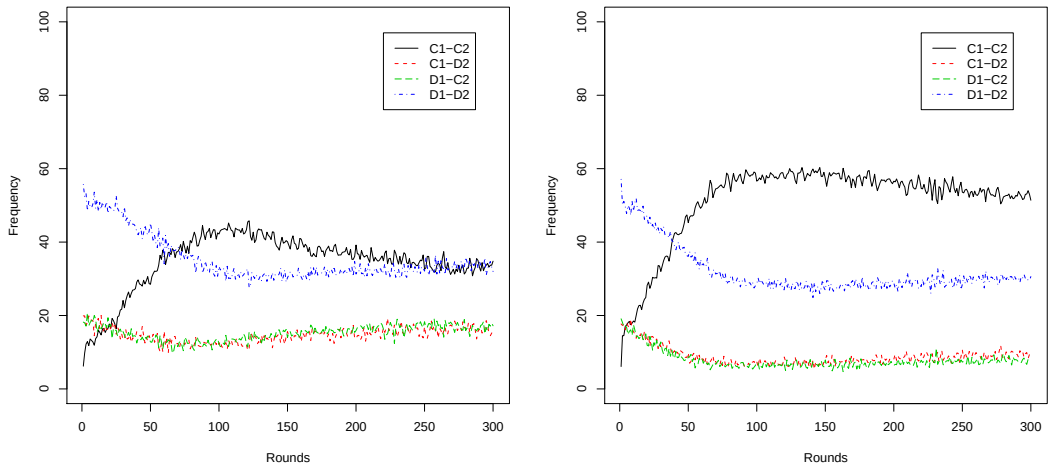
3.1.2 Model Behavior When MODEL 1 Plays with MODEL 1

When MODEL 1 plays against itself, we observe two different types of results in a 300 round game. In first type, there are defective outcomes where frequency of [defect, defect] is high and frequency of other outcomes is close to zero. Second type can be classified as cooperative equilibrium, since [cooperate, cooperate] is the most frequent outcome and almost half of the games are resulted in cooperative outcome and rest of the games have symmetric outcomes. Rest of the games have mixed outcomes and low frequency (Figure 36 (a)). Mean number of cooperative outcomes is low in cooperative equilibria compared to experiments with human subjects. Simulation data can not be compared in terms of the distribution of mixed, cooperative and defect results since the experimental data is not extensive enough to provide distribution information.



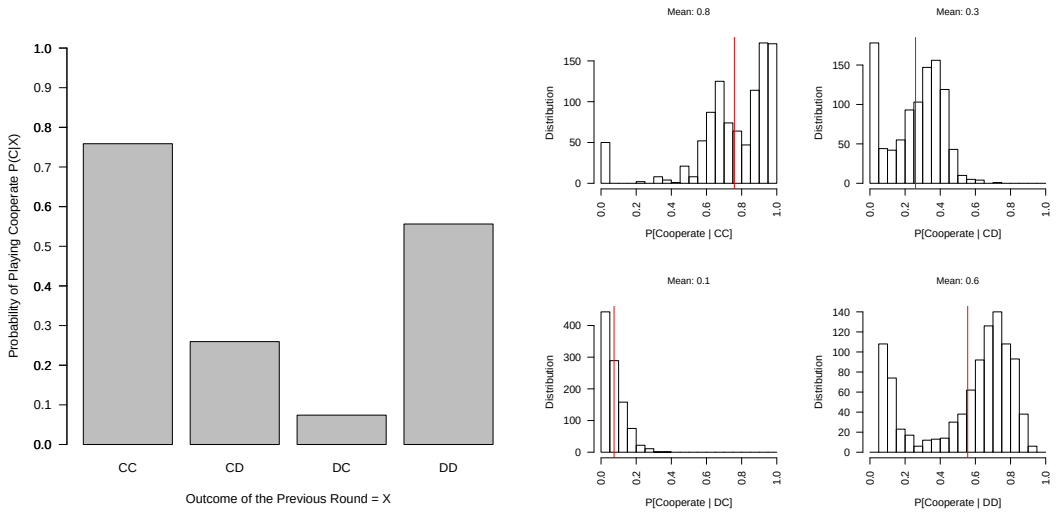
(a) (b)

Figure 36: Distribution of outcomes when MODEL 1 plays against MODEL 1, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30.



(a) (b)

Figure 37: Frequency of outcomes as game progresses when MODEL 1 plays against MODEL 1, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30.

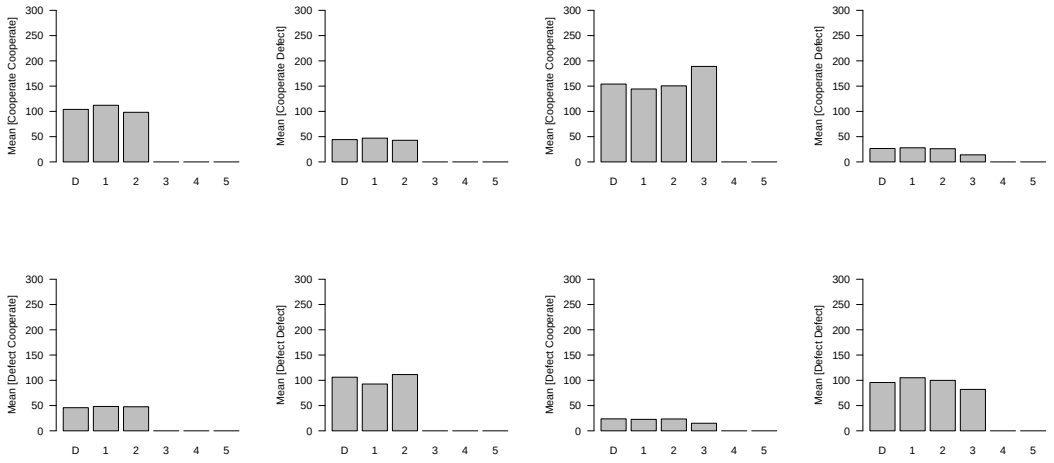


(a) (b)

Figure 38: Mean of cooperation probabilities (a) and distribution of cooperation probabilities (b) given the outcome of previous round when MODEL 1 plays against MODEL 1, decay = 0.5, noise = 0.1, L=30.

If decay rate which controls the forgetting in declarative memory module is increased, performance of the model improves. Both the frequency of cooperative equilibrium in 1000 experiments and the number of [cooperate, cooperate] outcomes in a cooperative equilibrium increase following the increase in decay rate. As a result, mean of [cooperate, cooperate] outcome in 300 rounds increases significantly from 103.9 to 154.1, meanwhile frequencies of defective and asymmetric outcomes decline (Figure 36 (b)). This difference between two cases can be explained through the learning curves which depicts the change of mean frequencies as the game progresses.

At the beginning of the game, frequency of [defect, defect] is the highest as expected. In later rounds, defection frequency decreases while the frequency of cooperative outcome is raising. Although the frequency of asymmetric outcomes decline for a while, they settle to their initial level while the frequency of cooperative outcome is decreasing. Since decay rate is low in the first experiment, high increase in cooperative outcome and the decline in asymmetric outcomes are temporary. In decision making process, model can recall asymmetric outcomes with a higher probability due to low forgetting rate and this will lead to selecting defect with a high probability (Figure 37 (a)). When decay rate is higher, retrieval probabilities of asymmetric outcomes



(a) (b)

Figure 39: Outcome frequencies with respect to cooperative outcomes in initial rounds when MODEL 1 plays against MODEL 1, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30.

decrease. Therefore, model recalls [cooperate, cooperate] and [defect, defect] more often in the decision-making process and selects cooperation with a higher probability (Figure 37 (b)).

When the behavior of MODEL 1 is analyzed, mean of conditional cooperation probabilities point to a Pavlovian type of behavior which cooperates after cooperative and defective outcomes with probabilities of 0.8 and 0.5 respectively. Player defects with a higher probability after asymmetric outcomes, though it is more forgiving than the Pavlovian player after [cooperate, defect] outcome (Figure 38 (a)). However, when we look at the distribution of conditional cooperation probabilities in 1000 iterated games, it can be concluded that there are two distinct type of players when their behavior after [defect, defect] is examined. Two types differ in their reaction after [defect, defect] outcome, first one exhibits a Pavlovian behavior and cooperates with a higher probability, whereas second type shows Tit-for-Tat characteristics and defects after [defect, defect] with a higher probability (Figure 38 (b)).

In addition to model parameters, behavior of model against itself depends on game history, since activation levels are modified after every round of the iterated game. In order to investigate the effects of game history and cooperation at initial rounds, simu-

lation data are analyzed in terms of mean frequency of each outcome in 300 rounds and number of cooperative outcomes at initial rounds (Figure 39). First bar in each diagram represents mean frequency for all simulation data, whereas second column represents mean frequency of outcomes for the instances where result of the first round is [cooperate, cooperate]. In Figure 39 x-axis represents consecutive rounds of cooperative outcome at initial rounds of the game. Cooperation in early rounds is not an indicator of high cooperation level in all rounds of the iterated game according to Figure 39. For MODEL 1, simulation data does not produce any instances where first four and five rounds are resulted in cooperation. Therefore, additional simulation data and other analysis tools are required to investigate the impact of early cooperation on simulation results.

3.2 Memory Model of Iterated Prisoner's Dilemma Based on Outcome Patterns

MODEL 2 records game history as outcome patterns, the results of two subsequent rounds is represented as a memory chunk. After receiving feedback about last round, model tries to predict the most likely outcome of making defect and cooperate moves. MODEL 2 selects the move with highest payoff and records the outcome with the outcome of previous round in the form of a pattern chunk.

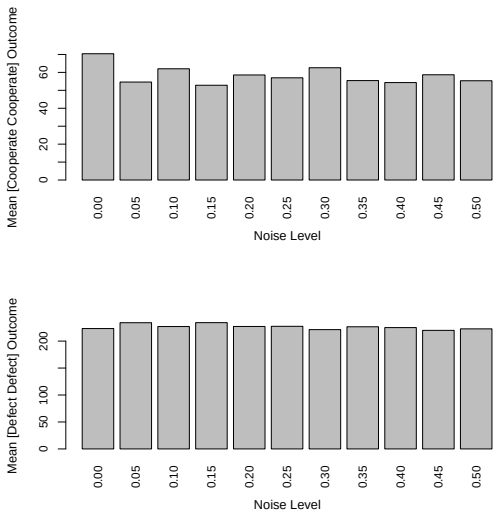
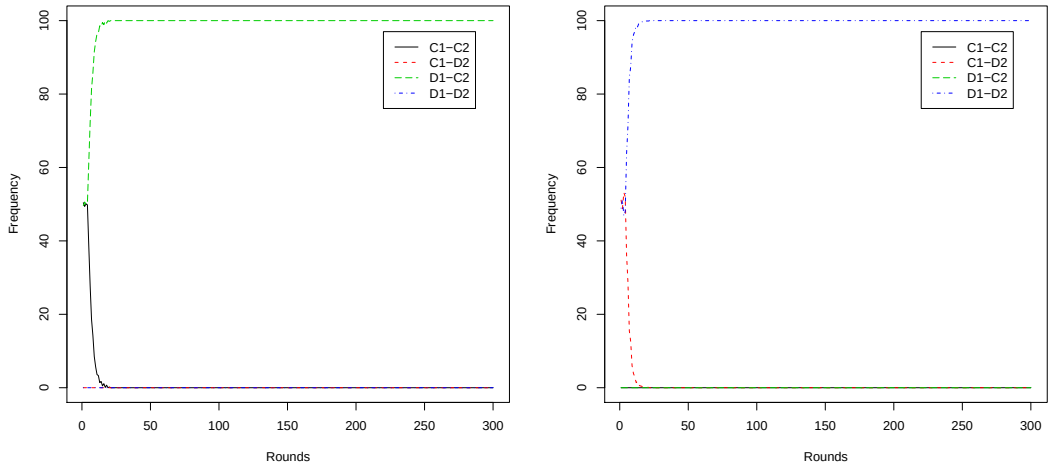


Figure 40: Mean of [cooperate, cooperate] and [defect, defect] outcomes when MODEL 2 plays against MODEL 2 and decay = 0.5, L = 30 are constant, noise is variable.

All pattern chunks are created after the start of the game, so L parameter is not a functional parameter for the second model, it is chosen as 30 similar to other models. Decision making depends on the retrieval of pattern chunks and activation levels of these chunks. As a result, noise parameter, s which determines the noise level in the activation equation and d parameter which refers to the forgetting rate for memory chunks are crucial for the decision making. Parameter space is explored for different values of s parameter from 0 to 0.5 with increments of 0.05 and d parameter between 0.4 and 0.9 with 0.1 increments. As Figure 40 illustrates, noise level does not appear to be effective in determining outcome frequencies.

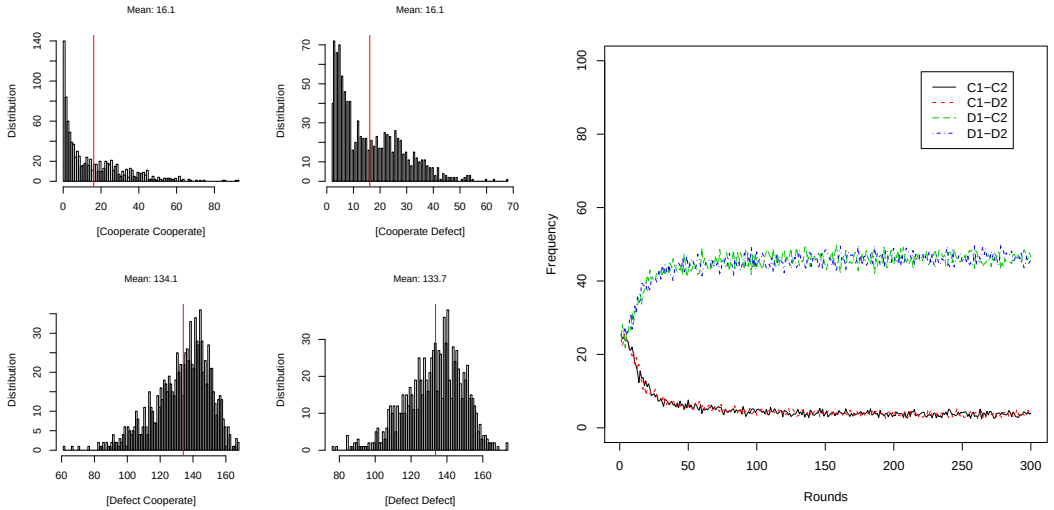
In ACT-R models, decay parameter is usually set as 0.5 (Lebiere et al., 2000), therefore we will present results of the case when decay parameter is equal to 0.5. Noise level is chosen as 0.1 where frequency of cooperative outcome is high and we observe cooperative, defective and mixed outcomes in distribution of outcomes at this noise level when MODEL 2 is playing against itself. Impact of increasing decay rate to 0.8 is also demonstrated.

3.2.1 Model Behavior against Basic Strategies



(a) (b)

Figure 41: Frequency of outcomes as game progresses when MODEL 2 plays against (a) ALLC strategy (b) ALLD strategy, decay = 0.5, noise = 0.1, L=30.

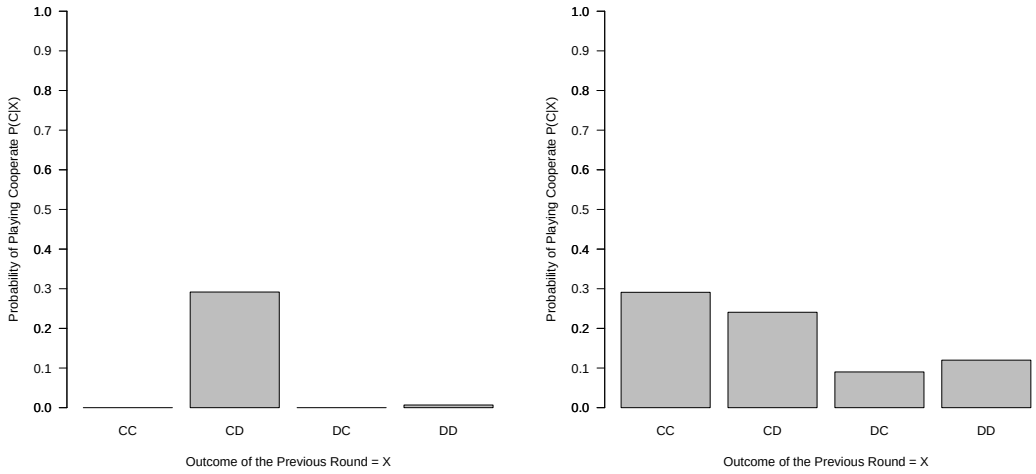


(a) (b)

Figure 42: Distribution of outcomes (a) and frequency of outcomes as game progresses (b) when MODEL 2 plays against Random (RAN) strategy, decay = 0.5, noise = 0.1, L=30.

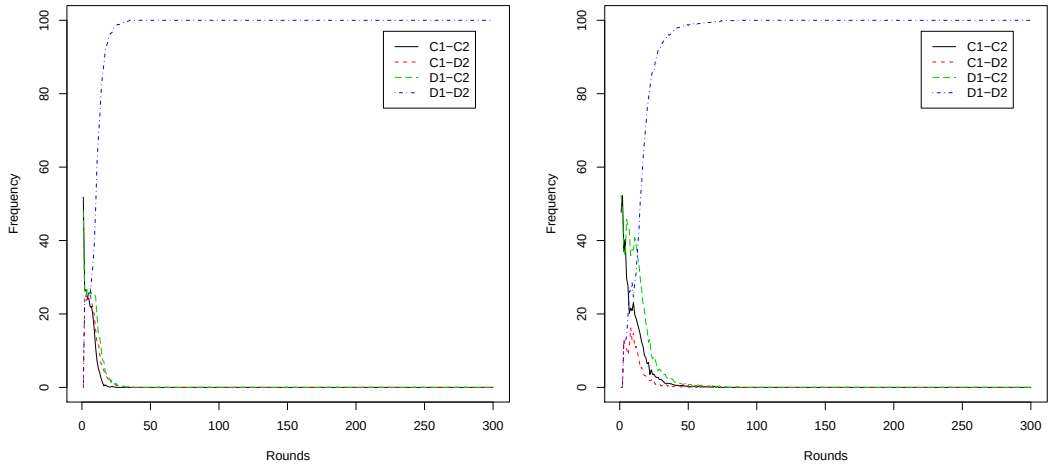
In this section, performance of MODEL 2 against basic strategies is evaluated. Distribution of four outcomes is similar to the first model when MODEL 2 is playing against unconditional strategies. Frequency of outcomes with respect to time course of the iterated game is represented in Figure 41. MODEL 2 successfully shifts to defect against All-Cooperate strategy in order to exploit unconditional cooperation. Thus, [defect, cooperate] becomes the prevalent outcome of the iterated game (Figure 41 (a)). Against All-Defect, second model shows a better performance than the first model. MODEL 2 instantaneously learns to defect against ALLD strategy and continues to defect until the end of the game. In contrast to first model, MODEL 2 never explores the cooperate option (Figure 41 (b)).

Against Random strategy, playing defect is more beneficial, since by playing defect the player may avoid [cooperate, defect] outcome and take advantage of the high frequency of [defect, cooperate] outcome. When playing with Random strategy, MODEL 2 learns to play defect very fast and avoids cooperate move at the rest of the iterated game. Frequency of cooperate move by the second model is lower compared to the first model. Distribution of outcomes and change of frequencies throughout the iterated game are demonstrated in Figure 42.



(a) (b)

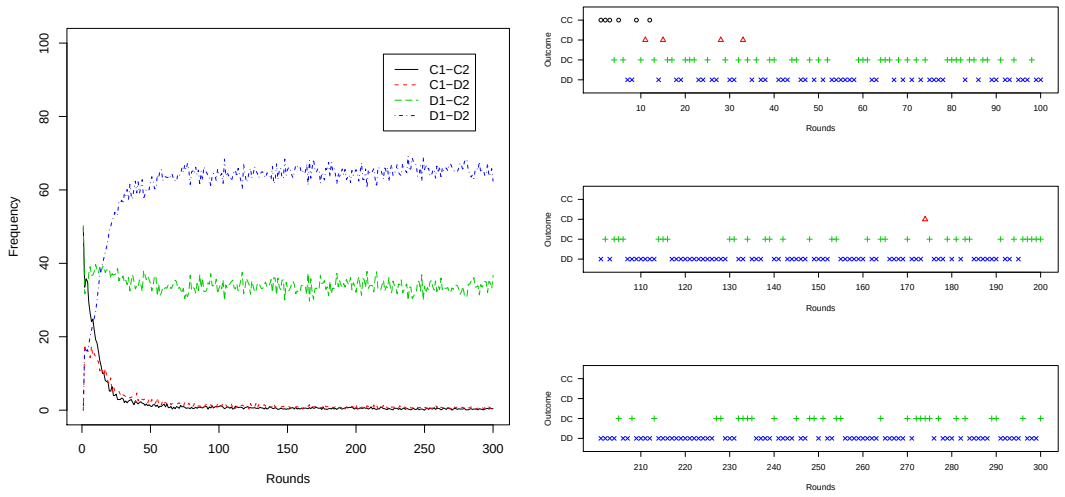
Figure 43: Mean of cooperation probabilities given the outcome of previous round when MODEL 2 plays against (a) All-Defect (ALLD) strategy (b) Random (RAN) strategy, decay = 0.5, noise = 0.1 L=30.



(a) (b)

Figure 44: Frequency of outcomes as game progresses when MODEL 2 plays against (a) Tit-for-Tat (TFT) strategy, (b) Tit-for-Two-Tats (TFTT) strategy, decay = 0.5, noise = 0.1, L=30

Mean cooperation probabilities conditional to the outcome of the previous round are presented in Figure 43. When playing against All-Defect and Random strategies, cooperation probabilities are small as expected. However, compared to first model, MODEL 2 does not exhibit a Pavlovian probability pattern.



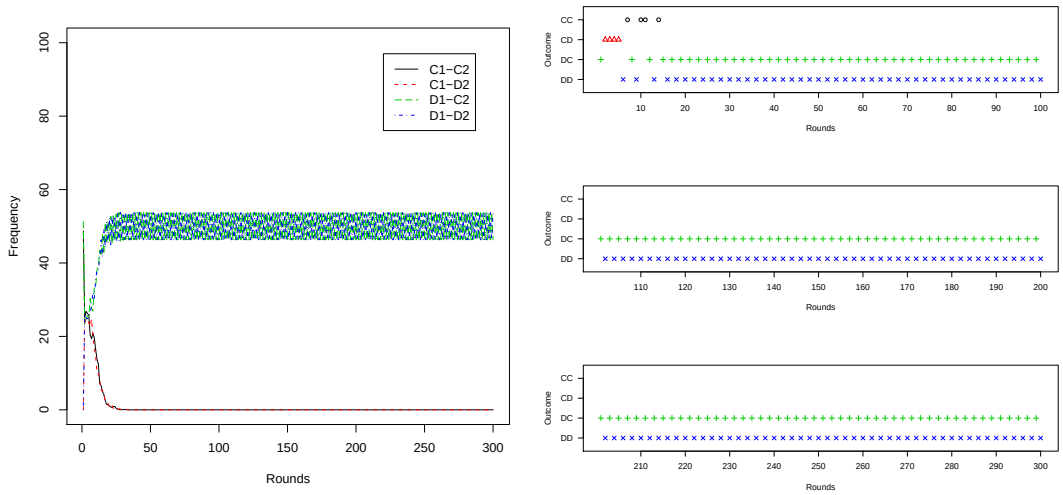
(a) (b)

Figure 45: Frequency of outcomes as game progresses (a) and outcome history of an exemplary game (b) when MODEL 2 plays against Forgiving Tit-for-Tat (TFTF) strategy, decay = 0.5, noise = 0.14, L=30.

Similar to first model, MODEL 2 fails to achieve cooperative equilibrium with Tit-for-Tat and Tit-for-Two-Tats strategies. Model fails to recognize the conditional rules that are employed by the TFT strategy. Since certain outcome patterns become prevalent at the early stages of the game, model does not engage in exploratory behavior. Consequently, Tit-for-Tat strategy and MODEL 1 do not attain cooperation in almost all iterative games (Figure 44 (a)).

When playing against TFTT strategy, MODEL 2 exhibits a similar pattern to the TFT case and can not reach cooperative equilibrium (Figure 44 (b)). Unlike MODEL 1, second model fail to reach a high score against TFTT strategy since frequency of asymmetric outcomes are close to each other. This means that MODEL 2 can not take advantage of behavioral pattern exhibited by TFTT strategy.

Against Forgiving Tit-for-Tat (TFTF), second model shifts to playing defect completely after the initial stages of the game (Figure 45 (a)). Throughout the iterated



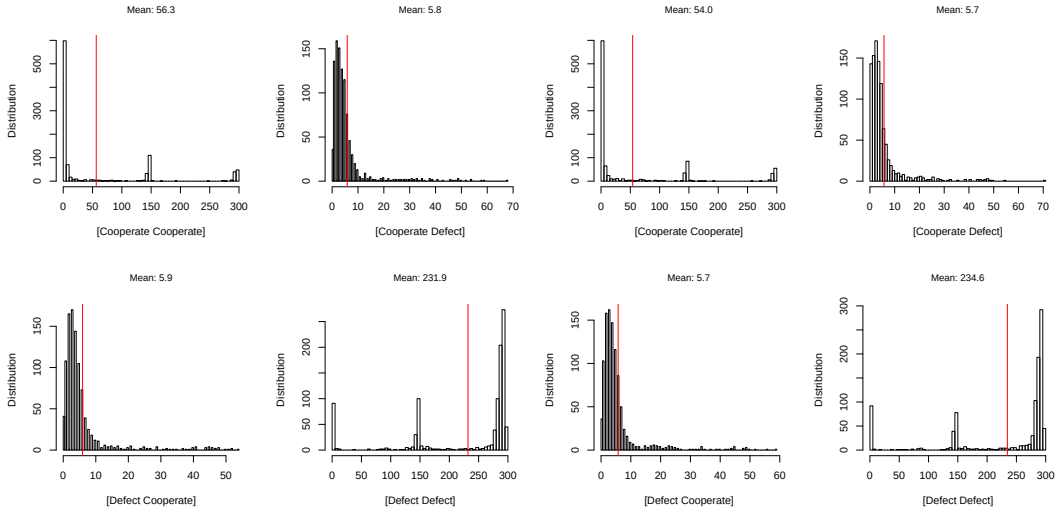
(a) (b)

Figure 46: Frequency of outcomes as game progresses (a) and outcome history (b) when MODEL 2 plays against Pavlovian (PAV) strategy, decay = 0.5, noise = 0.14, L=30.

game, frequencies of [cooperate, cooperate] and [cooperate, defect] outcomes quickly decline to zero. Although the high frequency of [defect, defect] outcome is disadvantageous for both players, MODEL 2 achieves a high score against TFTF strategy due to the high frequency of [defect, cooperate] outcome. When compared to first player, second model is more successful due to the higher frequency of this outcome.

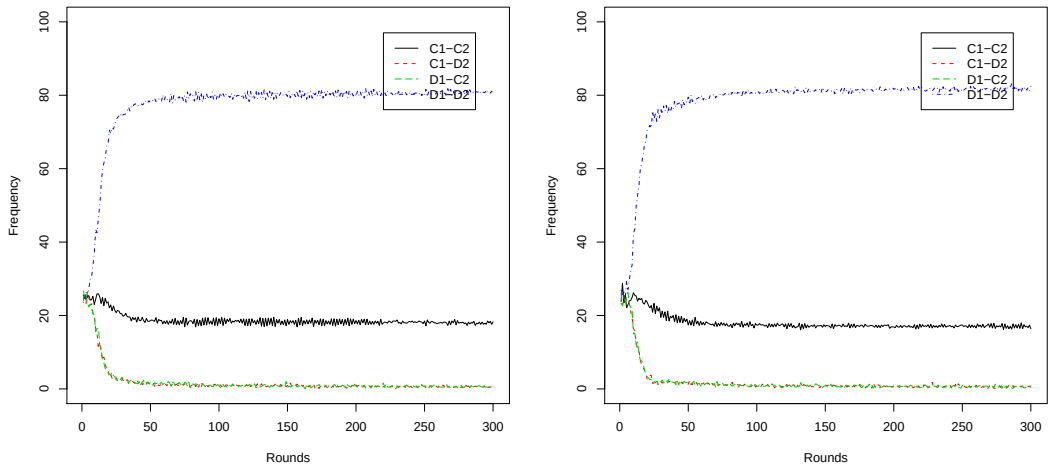
Similar to first model, MODEL 2 is very successful against Pavlovian strategy. Frequencies of the cooperative outcome, [cooperate, cooperate] and the disadvantageous asymmetric outcome, [cooperate, defect] are close to zero (Figure 46 (a)).

MODEL 2 seems to employ a specific strategy against Pavlovian player. Player accurately detects the Pavlovian rule of shifting to cooperation after symmetric outcomes. In order to take advantage of this behavior, MODEL 2 continues to defect after [defect, defect] is observed which leads to advantageous [defect, cooperate] outcome in the next round. Unlike first model, MODEL 2 continues to defect after [defect, cooperate] outcome is observed.



(a) (b)

Figure 47: Distribution of outcomes when MODEL 2 plays against MODEL 2, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30.

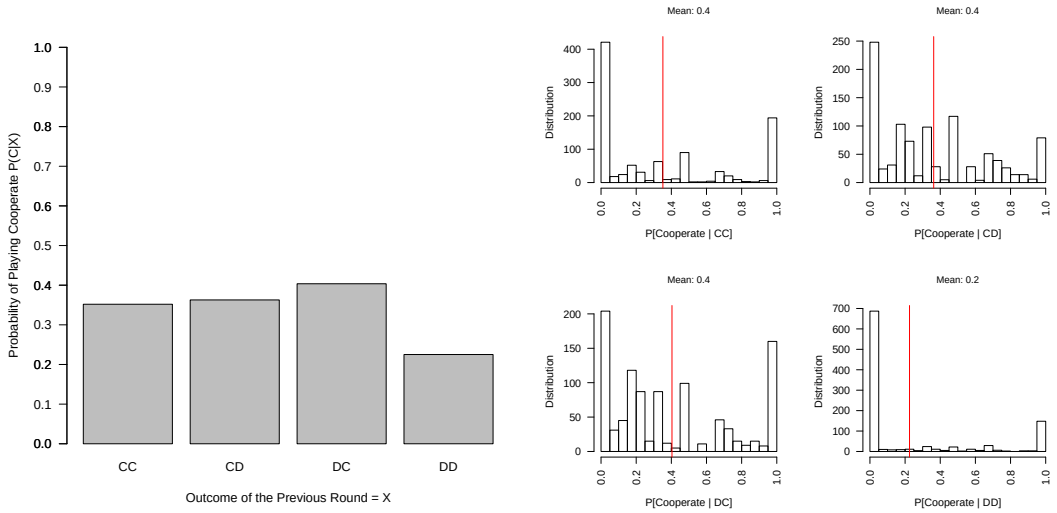


(a) (b)

Figure 48: Frequency of outcomes as game progresses when MODEL 2 plays against MODEL 2, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30.

3.2.2 Model Behavior When MODEL 2 Plays with MODEL 2

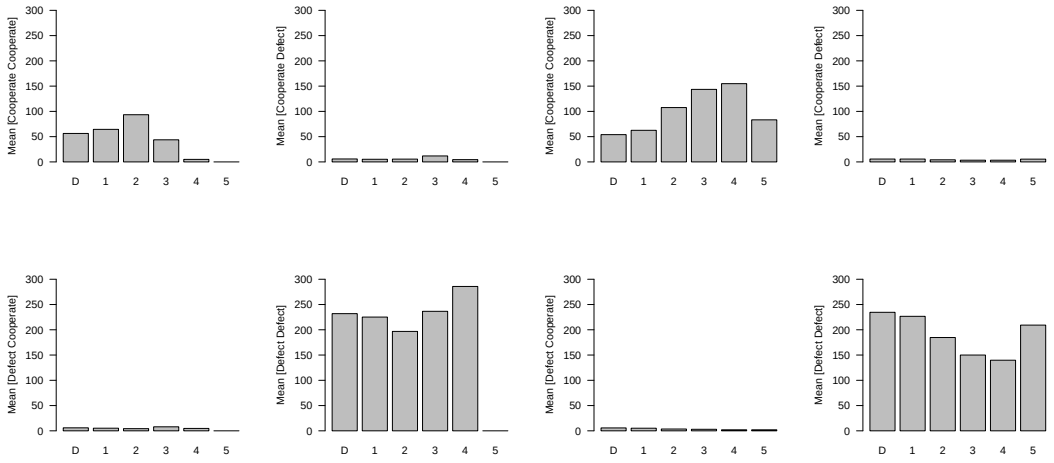
For second model, there are three different types of results in a 300 round game. Almost sixty percent of all games are dominated by the [defect, defect] outcome throughout the game. In second type, the frequencies of defective and cooperative outcomes are equal. Other games are resulted with cooperative equilibrium. In all games, the frequencies of asymmetric outcomes are close to zero. The mean number of cooperative equilibria is low when compared to human behavior (Figure 47 (a)).



(a) (b)

Figure 49: Mean of cooperation probabilities (a) and distribution of cooperation probabilities (b) given the outcome of previous round when MODEL 2 plays against MODEL 2, decay = 0.5, noise = 0.1, L=30.

When forgetting in the model, decay rate is increased to 0.8, there is no significant change in the performance of the model. Frequencies of symmetric and asymmetric outcomes remain at similar levels to the case where d is equal to 0.5 (Figure 47 (b)). When MODEL 2 is playing against itself, players either learn to defect together or cooperate together. Due to high frequency of defect equilibrium in experiments, mean frequency of [defect, defect] outcome is really high throughout the game. As learning plots (Figure 48) illustrate players learn to avoid asymmetric outcomes in early stages of the iterated game. Frequencies of the asymmetric outcomes converge to zero irrespective of the decay rate. Change in outcome frequencies exhibit similar trends for different values of decay rate (Figure 48 (a) (b)).



(a) (b)

Figure 50: Outcome frequencies with respect to cooperative outcomes in initial rounds when MODEL 2 plays against MODEL 2, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30.

When conditional cooperation probabilities are analyzed for second model, we can not assign a characteristic behavior to second model. Mean conditional cooperation probability for MODEL 2 shows that cooperation probability is less than 0.5 after each outcome and it is particularly smaller after [defect, defect] outcome (Figure 49 (a)). Moreover, distribution of conditional cooperation probabilities in 1000 iterated games does not allow us to classify different behavior characteristics. Conditional cooperation probabilities do not exhibit patterns or trends which would signify different behavioral characteristics (Figure 49 (b)).

Similar to first model, behavior of MODEL 2 depends on model parameters and outcome history. Figure 50 depicts the impact of early cooperative outcomes on outcome frequencies. In Figure 50, x-axis represents consecutive rounds of cooperative outcome at initial rounds of the game. Cooperation in early rounds seems to increase overall cooperation frequency. Figure 50 depicts an increase in mean frequency of cooperative outcome and decrease in defective outcome as number of consecutive rounds of cooperation increases. However, this relation is not observed for the case where first five rounds of the game resulted in cooperation. Therefore, in order to investigate the effects of early cooperation on simulation results, additional data and tools are

required.

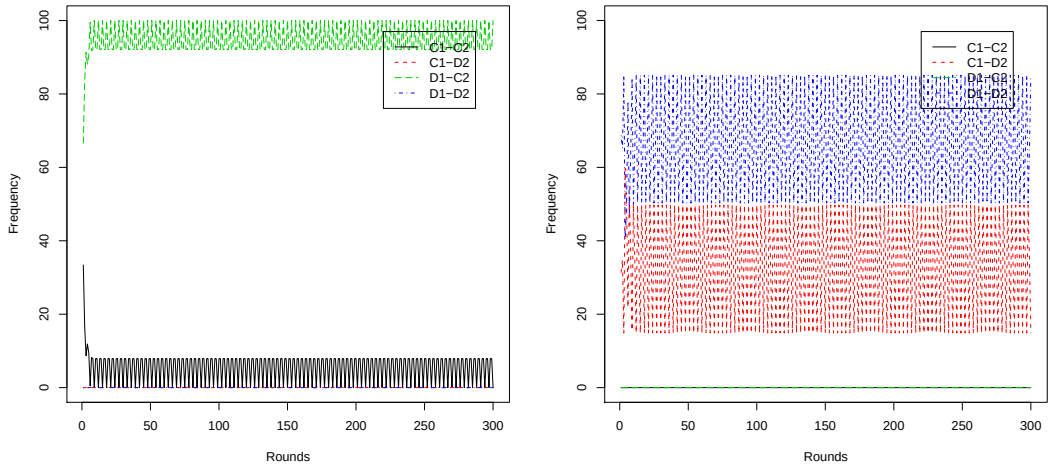
3.3 Predictive Memory Model of Iterated Prisoner's Dilemma Based on Outcome History and Outcome Patterns

When recording outcome history , MODEL 3 uses two different types of information in decision making. This model keeps track of outcome of each round and records results of two consecutive rounds as an outcome pattern. After recording the outcome of the last round, model predicts the most likely outcome of making defect and cooperate moves. Then player tries to determine future outcomes associated with making these moves using information about outcome patterns. Present and future payoffs associated with defect and cooperate moves are evaluated and player selects the move with the highest expected payoff.

Decision making depends on the activation levels of both outcome and pattern chunks. Outcome chunks are created before the start of the game, thus L parameter which represents the number of initial references to the outcome chunks is an important parameter for the model. Moreover, d parameter which controls forgetting rate and s parameter which sets the noise level are also effective in determining the behavior of the model. In addition to these three parameters, third model employs a forth parameter, α which determines the weight of future payoffs in the calculation of expected payoffs for different moves.

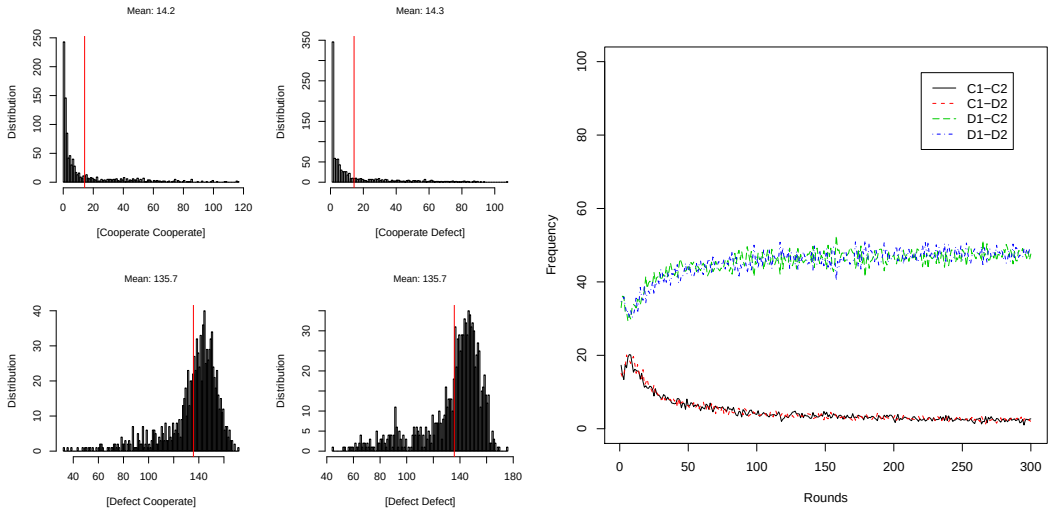
Decision making depends on the retrieval of pattern and outcome chunks and activation levels of these chunks. L parameter is effective and chosen as 30 similar to other models. Parameter space is explored for different values of s parameter from 0 to 0.5 with increments of 0.05, d parameter between 0.4 and 0.9 with 0.1 increments and α parameter between 0.5 to 2.0 with 0.5 increments .

In ACT-R models, decay parameter is usually set as 0.5 (Lebiere et al., 2000), therefore we will present results of the case when decay parameter is equal to 0.5. Parameters are set as $s = 0.0$, $L = 30$, $\alpha = 1.5$ where frequency of cooperative outcome is high and we observe cooperative, defective and mixed outcomes in distribution of



(a) (b)

Figure 51: Frequency of outcomes as game progresses when MODEL 3 plays against (a) ALLC strategy (b) ALLD strategy, decay = 0.5, noise = 0.0, L=30, α = 1.5.

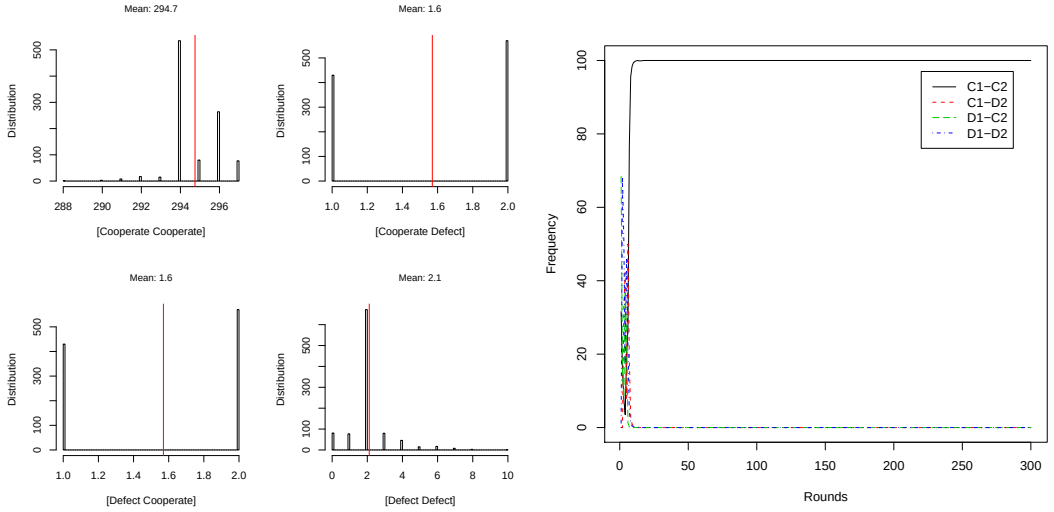


(a) (b)

Figure 52: Distribution of outcomes (a) and frequency of outcomes as game progresses (b) when MODEL 3 plays against Random (RAN) strategy, decay = 0.5, noise = 0.0, L=30, α= 1.5.

outcomes in the case when the model is playing with itself. Impact of increasing forgetting rate to 0.8 is also demonstrated in the section.

3.3.1 Model Behavior against Basic Strategies

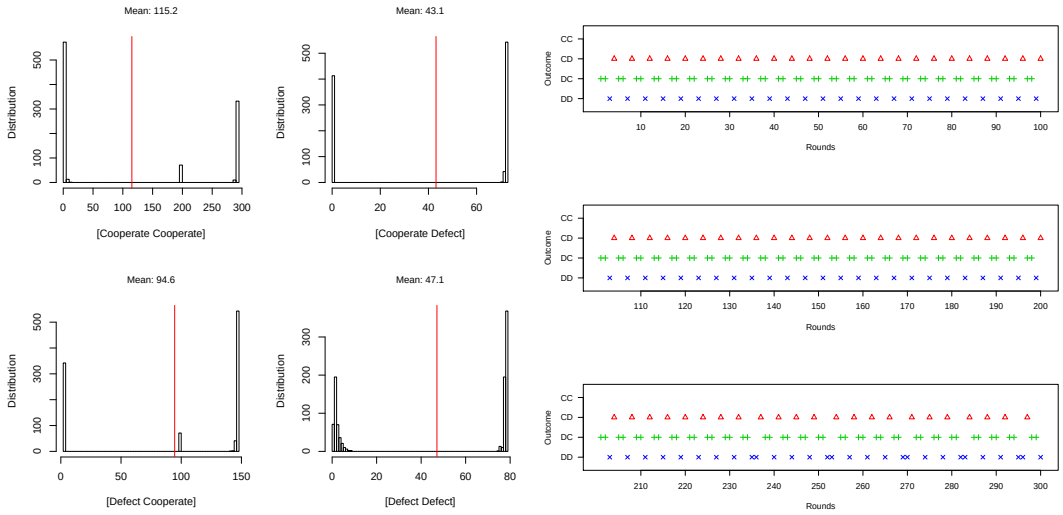


(a) (b)

Figure 53: Distribution of outcomes (a) and frequency of outcomes as game progresses (b) when MODEL 3 plays against Tit-for-Tat (TFT) strategy, decay = 0.5, noise = 0.0, L=30, α= 1.5.

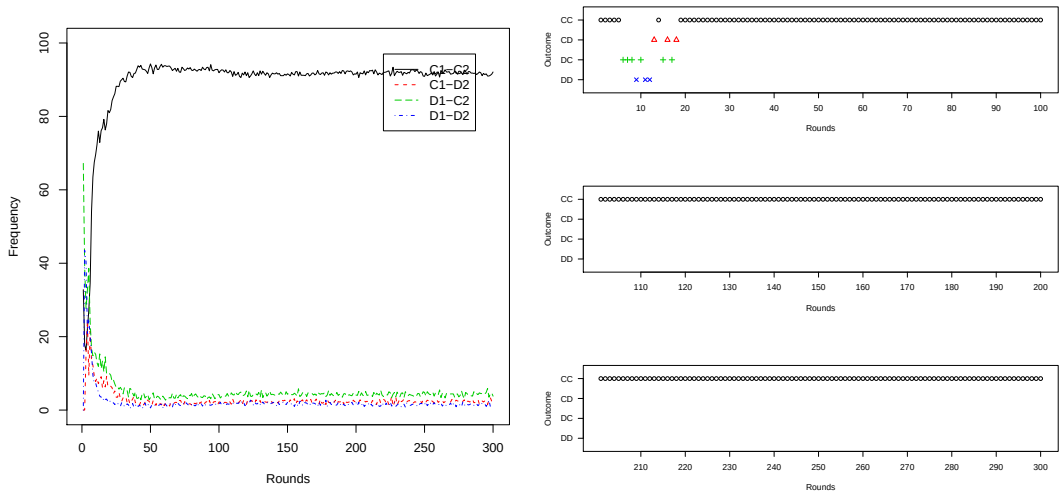
MODEL 3 detects the unconditional behavior pattern of ALLC strategy and shifts to defect move after a few rounds in the beginning of the game. Frequency of outcomes with respect to time course of the iterated game is represented in Figure 51 (a). Against All-Defect, performance of the model differs from first two models. Although MODEL 3 chooses to defect with a higher probability, frequency of selecting cooperate move is significantly higher than other models (Figure 51 (b)) . As a result, MODEL 3 performs worse compared to other models.

Defect move is the most favorable move against Random strategy since player may avoid disadvantageous [cooperate, defect] outcome and attain a high frequency of advantageous [defect, cooperate] outcome by playing defect against Random player. Similar to other models, MODEL 3 learns to play defect against Random strategy and stops cooperating after initial stages of the iterated game. Distribution of outcomes and change of frequencies throughout the iterated game are demonstrated in Figure 52.



(a) (b)

Figure 54: Distribution of outcomes (a) and frequency of outcomes as game progresses (b) when MODEL 3 plays against Tit-for-Two-Tats (TFTT) strategy, decay = 0.5, noise = 0.0, L=30, α= 1.5.

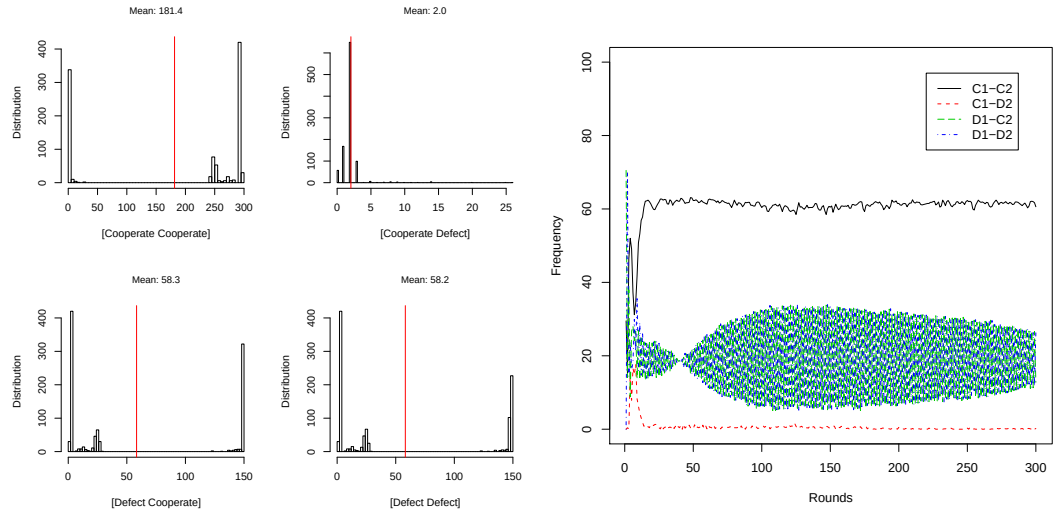


(a) (b)

Figure 55: Frequency of outcomes as game progresses (a) and outcome history of an exemplary game (b) when MODEL 3 plays against Forgiving Tit-for-Tat (TFTF) strategy, decay = 0.5, noise = 0.0, L=30, α= 1.5.

In contrast to first two models, MODEL 3 achieves cooperative equilibrium with Tit-for-Tat strategy. Model detects that cooperating with Tit-for-Tat strategy is profitable in the long run since Tit-for-Tat strategy punishes any divergence from cooperative move. [cooperate, cooperate] outcome becomes prevalent after initial rounds and the model does not engage in exploratory moves throughout the iterated game (Figure 53).

In around thirty percent of the games, MODEL 3 attains cooperative equilibrium with Tit-for-Two-Tats strategy. In other games, player develops a strategy which exploits the behavioral pattern of TFTT (Figure 54 (a)). After [defect, defect] outcome, MODEL 3 chooses to cooperate. TFTT strategy reciprocates cooperation in the next round. However, third model shifts back to defect after [cooperate, defect] is observed and continues to defect until [defect, defect] outcome is observed. This behavioral structure is illustrated in Figure 54 (b).

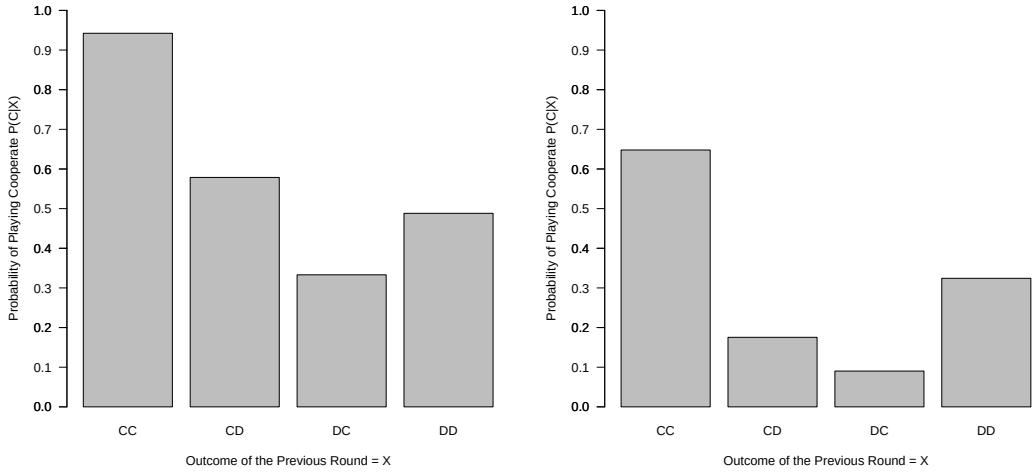


(a) (b)

Figure 56: Distribution of outcomes (a) and frequency of outcomes as game progresses (b) when MODEL 3 plays against Pavlovian (PAV) strategy, decay = 0.8, noise = 0.0, L=30, α= 1.5.

Against Forgiving Tit-for-Tat, behavior of MODEL 3 is different from first two models. First two models defects with a high frequency against TFTF strategy, whereas MODEL 3 attains a cooperative equilibrium against Forgiving Tit-for-Tat strategy. After initial rounds, third model learns to cooperate with Forgiving Tit-for-Tat strat-

egy, and frequencies of [defect, defect] outcome and asymmetric outcomes decreases quickly (Figure 55 (a)). Compared to first two models, third model fails to exploit forgiving behavior of TFTF strategy and does not achieve a high score against this strategy.



(a) (b)

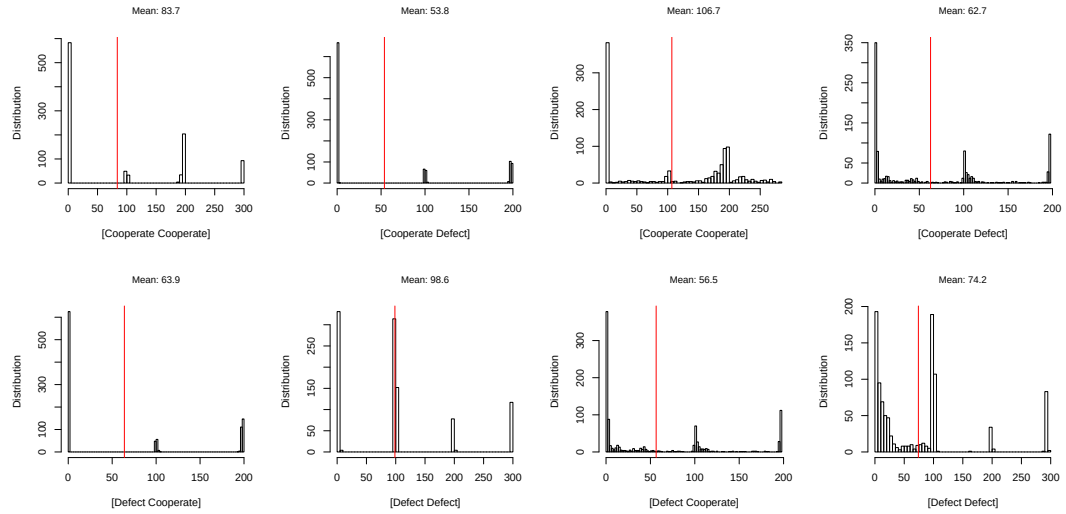
Figure 57: Mean of cooperation probabilities given the outcome of previous round when MODEL 3 plays against (a) Forgiving Tit-for-Tat (TFTF) strategy (b) Pavlovian (PAV) strategy, decay = 0.5, noise = 0.0, L=30, α = 1.5.

Distribution and frequency of outcomes are presented in Figure 56, when decay rate is 0.8 and MODEL 3 is playing against Pavlovian strategy. More than fifty percent of the games, MODEL 3 reaches cooperative equilibrium with Pavlovian strategy (Figure 56 (a)) . In other games, the model exhibited a behavioral pattern which resembles MODEL 2. Similar to second model, third model exploits Pavlovian strategy successfully. MODEL 3 learns the Pavlovian behavior of shifting to cooperation after [defect, defect] outcome and continues to defect. The advantageous [defect, cooperate] outcome follows defective outcome and both players select defect move, thus frequency of [cooperate, defect] and [cooperate, cooperate] is close to zero in these set of games (Figure 56 (a)).

Behavior of third model against Forgiving Tit-for-Tat and Pavlovian strategies can be analyzed according to mean conditional cooperation probabilities after each outcome (Figure 57). MODEL 3 presents a Pavlovian pattern against both teaching and

learning strategies. However, conditional cooperation probabilities after asymmetric outcomes are significantly higher against Forgiving Tit-for-Tat strategy.

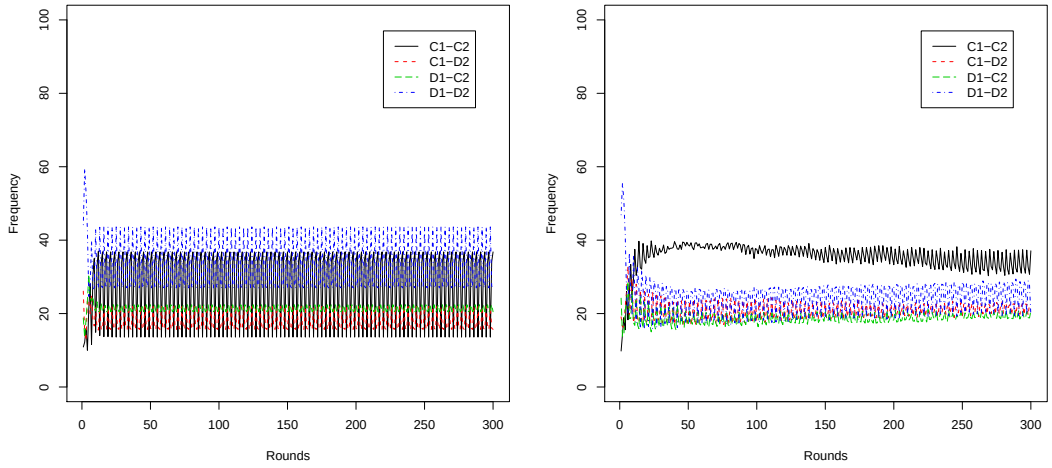
3.3.2 Model Behavior When MODEL 3 Plays with MODEL 3



(a) (b)

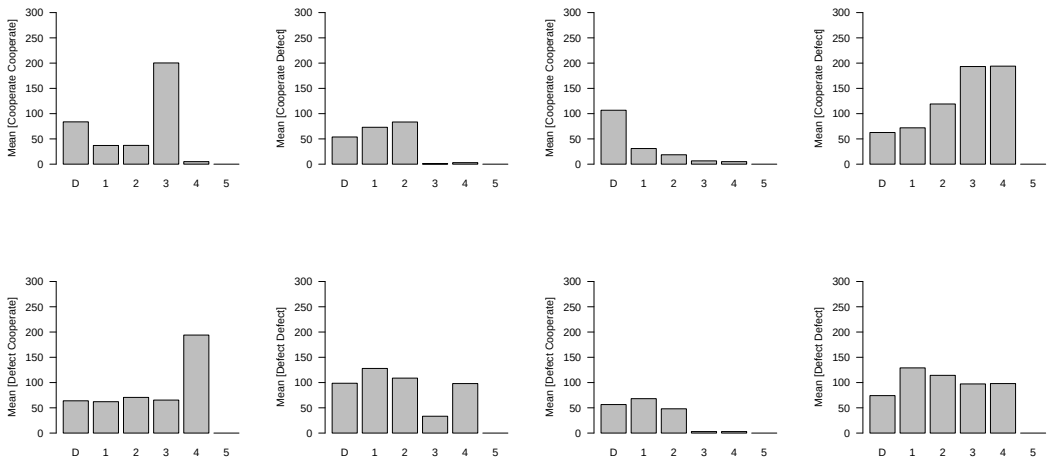
Figure 58: Distribution of outcomes when MODEL 3 plays against MODEL 3, (a) decay = 0.5, noise = 0.0, L=30, α= 1.5, (b) decay = 0.8, noise = 0.0, L=30, α= 1.5.

When third model plays against itself, we observe three different results in 1000 experiments. Around ten percent of experiments result in defective equilibrium where [defect, defect] becomes the dominating outcome of 300 rounds. Another ten percent of experiments converge to a cooperative equilibrium and [cooperate, cooperate] becomes the prevalent outcome in the iterated game. In other experiments, asymmetric outcomes become dominant and frequencies of symmetric outcomes are close to zero. In these experiments, players fail to coordinate their behaviors and asymmetric outcomes converge to values with a proportionality of one to two. One of the asymmetric outcomes becomes dominant outcome with a frequency of two times the frequency of the other asymmetric outcome. Consequently, one player reaches a high score than the other due to the difference in the frequency of asymmetric outcomes (Figure 58 (a)). Mean number of cooperative outcomes is low in cooperative equilibria compared to experiments with human subjects.



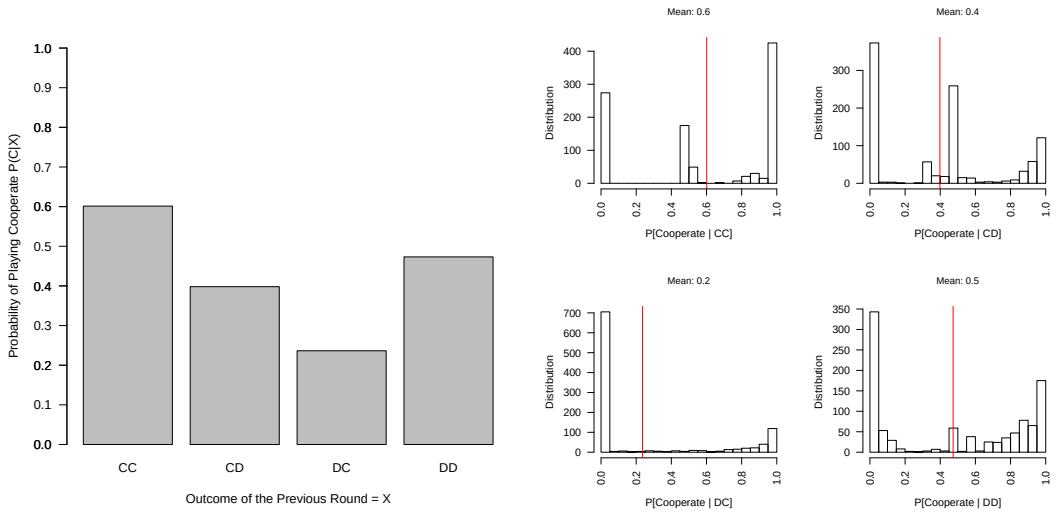
(a) (b)

Figure 59: Frequency of outcomes as game progresses when MODEL 3 plays against MODEL 3, (a) decay = 0.5, noise = 0.0, L=30, α= 1.5, (b) decay = 0.8, noise = 0.0, L=30, α= 1.5.



(a) (b)

Figure 60: Outcome frequencies with respect to cooperative outcomes in initial rounds when MODEL 3 plays against MODEL 1, (a) decay = 0.5, noise = 0.0, L=30, α= 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5



(a) (b)

Figure 61: Mean of cooperation probabilities (a) and distribution of cooperation probabilities (b) given the outcome of previous round when MODEL 3 plays against MODEL 3,decay = 0.8, noise = 0.0, L=30, α = 1.5.

When decay rate is higher and the level of forgetting is increased, number of cooperative equilibria increases significantly, whereas the number of defective equilibria decrease in 1000 experiments (Figure 58 (b)). When MODEL 3 is playing against itself, there are three possible learning patterns. Players either learn to defect together or cooperate together. In the third case, players fail to attain coordination and asymmetric outcomes are dominant throughout the game (Figure 59 (a)). Frequencies of asymmetric outcomes do not change significantly with respect to different forgetting levels. However, frequency of cooperative outcome increases as the decay rate increases (Figure 58 (b)). Change in frequency levels is illustrated in learning curves which depicts the change of mean frequencies as the game progresses.

Mean of conditional cooperation probabilities indicate a Pavlovian behavior which cooperates after cooperative and defective outcomes with probabilities of 0.8 and 0.5 respectively. This probability structure is similar to the first model. Player defects with a higher probability after asymmetric outcomes, though the model cooperates with a higher probability after [cooperate, defect] and [defect, cooperate] outcomes (Figure 61 (a)). However, distribution of conditional cooperation probabilities in 1000 iterated games indicate two distinct types of players. When their behaviors after [defect, de-

fect] are analyzed, first one exhibits a Pavlovian behavior and cooperates with a high probability, whereas the second type shows Tit-for-Tat characteristics and defects after [defect, defect] with a higher probability (Figure 61 (b)).

Similar to first two models, parameters and game history are instrumental in determining behavior of MODEL 3. Effects of cooperation in early stages of the game on outcome frequencies are depicted in Figure 60 according to two different forgetting rates. When forgetting rate is low, cooperation in early rounds has a decreasing impact on mean frequency of cooperative outcome. However, when [cooperate, cooperate] is observed in first three rounds of the game, more than fifty perfect of the iterated games resulted in cooperative outcome. When [cooperate, cooperate] is the result for the first four rounds of iterated game, frequency of cooperative outcome is lowest and at least one player defects unconditionally. When decay rate is higher, a decreasing cooperation trend is observed with respect to increase in early cooperation. Figure 60 points to a complex relation between model behavior and game history. As a result, additional simulation data and analysis tools are required for the investigation of model behavior in relation to game history.

3.4 Associative Memory Model of Iterated Prisoner's Dilemma Based on Outcome History

Decision making process depends on the outcome history and activation levels of outcome chunks for MODEL 4. Forth model resembles MODEL 1 in terms of memory items and decision making process. Similar to first model, MODEL 4 records outcome history by recreating outcome chunks. Model tries to predict the most likely outcome of making a specific move by retrieving outcome chunks from declarative memory module. After the retrieval process, procedural module evaluates defect and cooperate moves in terms of their expected payoffs and decides to exercise the move with the highest payoff. Retrieval of outcome chunks depends on the activation levels. As a result, L parameter which controls the initial number of references is effective in determining the activation levels of chunks. Moreover, s parameter which controls

noise level in the activation equation and *d* parameter which sets the decay rate of the activation levels are important parameters for the third model.

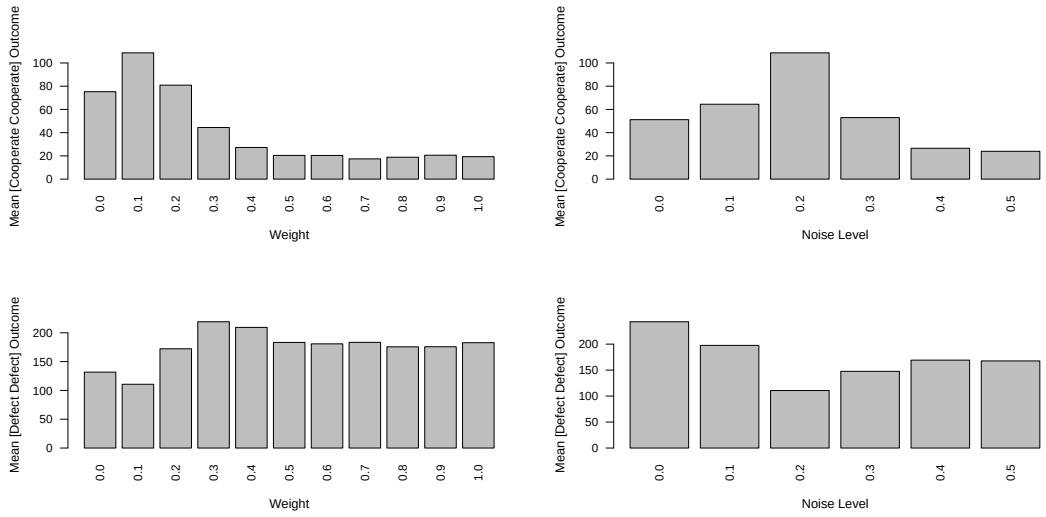


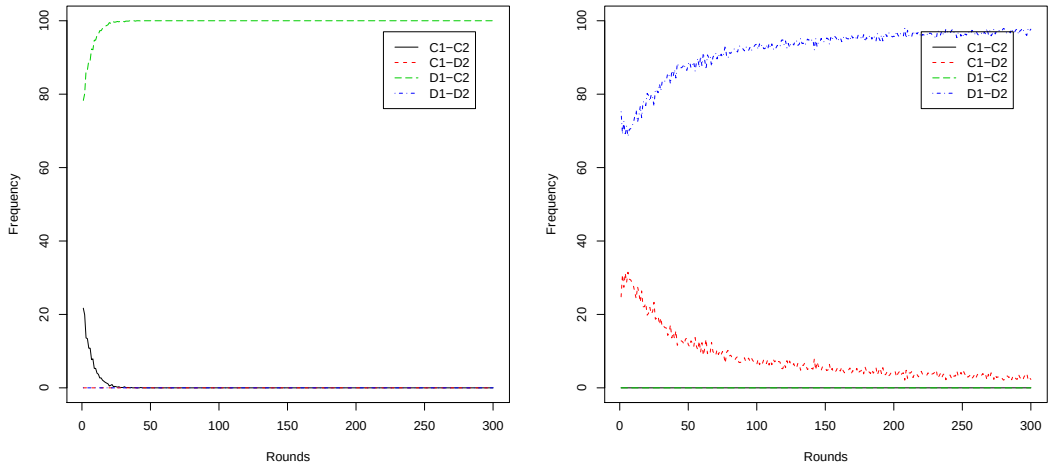
Figure 62: Mean of [cooperate, cooperate] and [defect, defect] outcomes when MODEL 4 plays against MODEL 4 and decay = 0.5, L = 30 (a) noise = 0.2 constant, weight is variable (b) w = 0.1 constant, noise is variable.

In addition to three parameters mentioned above, a forth parameter, *w* controls the impact of association strength between goal buffer and retrieval buffer on the retrieval of chunks. Activation levels of outcome chunks depend on the association between the two buffers. Associative strength between a goal chunk and memory chunk increases, when the chunk is retrieved from declarative memory. Presence of the same goal chunk increases retrieval probability for the memory chunk. Association mechanism between goal and memory modules is the main difference between first and forth models. As a result, weight parameter, *w* which controls the association rate is crucial for the decision making for the forth model.

Similar to other models, initial number of references to memory chunks, *L* is chosen as 30 for the forth model. Parameter space is explored for different values of *s* parameter from 0 to 0.5 with increments of 0.05, *w* parameter from 0 to 1 with 0.1 increments and *d* parameter between 0.4 and 0.9 with 0.1 increments. As Figure 62 illustrates, noise and weight parameters are effective in determining outcome frequencies.

Decay parameter is usually set to 0.5 in ACT-R models, therefore results for the $d = 0.5$ case will be presented in detail. However, effects of increasing forgetting are demonstrated with $d = 0.8$ case. Noise parameter s is chosen as 0.2 and weight parameter is set to 0.1. For these values, frequency of cooperative outcome is high and cooperative, defective and mixed equilibrium are present for different iterated games. Performance of MODEL 4 is explained in the next section.

3.4.1 Model Behavior against Basic Strategies

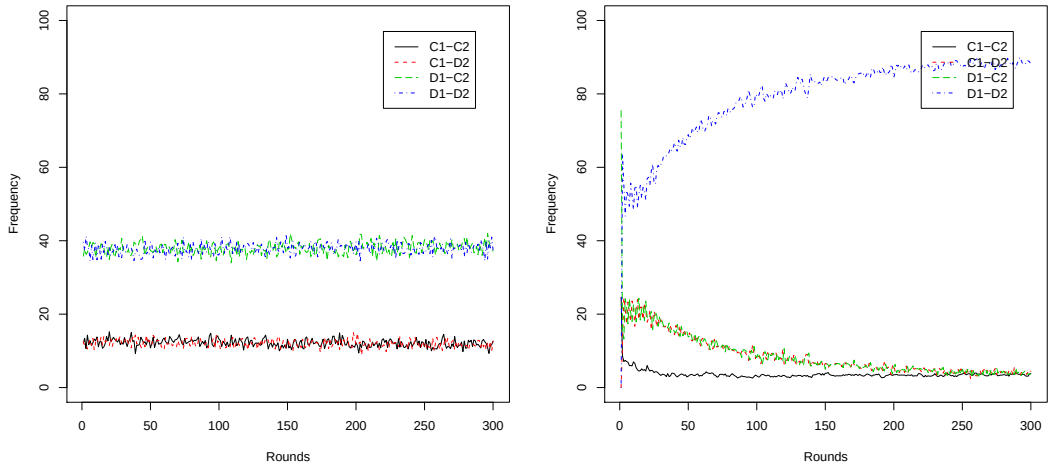


(a) (b)

Figure 63: Frequency of outcomes as game progresses when MODEL 4 plays against (a) ALLC strategy (b) ALLD strategy, decay = 0.5, noise = 0.2, L = 30, w = 0.1.

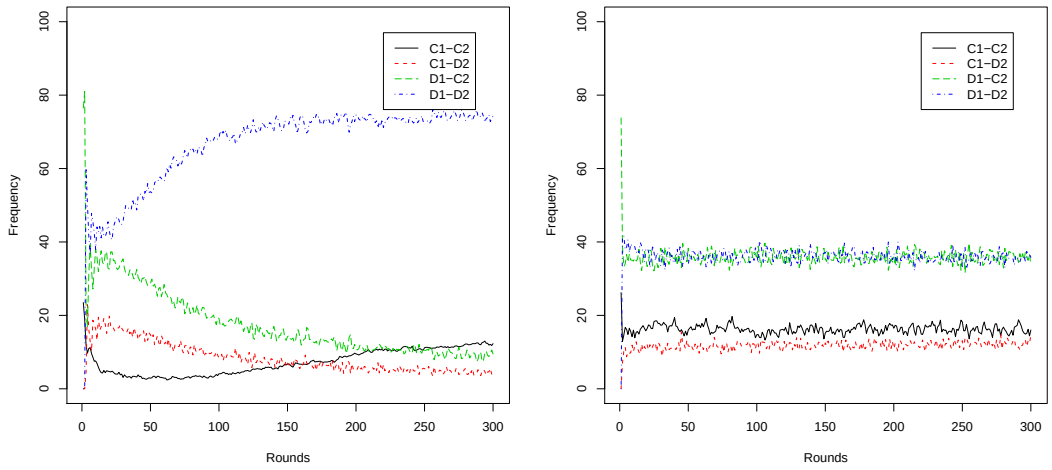
Against basic strategy, performance of MODEL 4 is similar to first model. MODEL 4 detects unconditionality in the behavior of All-Cooperate strategy and learns to play defect against this player. Although we observe [cooperate, cooperate] outcome at the early stages of the iterated game, [defect, cooperate] becomes prevalent as the game progresses. Due to the high payoff of the outcome, player continues to defect without exploring other behavior patterns (Figure 63 (a)).

Against All-Defect, forth model learns to defect with a high frequency in order to defend itself against unconditional defect strategy exercised by ALLD player. As a result [defect, defect] is the dominant outcome of the iterated game, this outcome is not favorable due to its negative payoff. Therefore forth model explores cooperate



(a) (b)

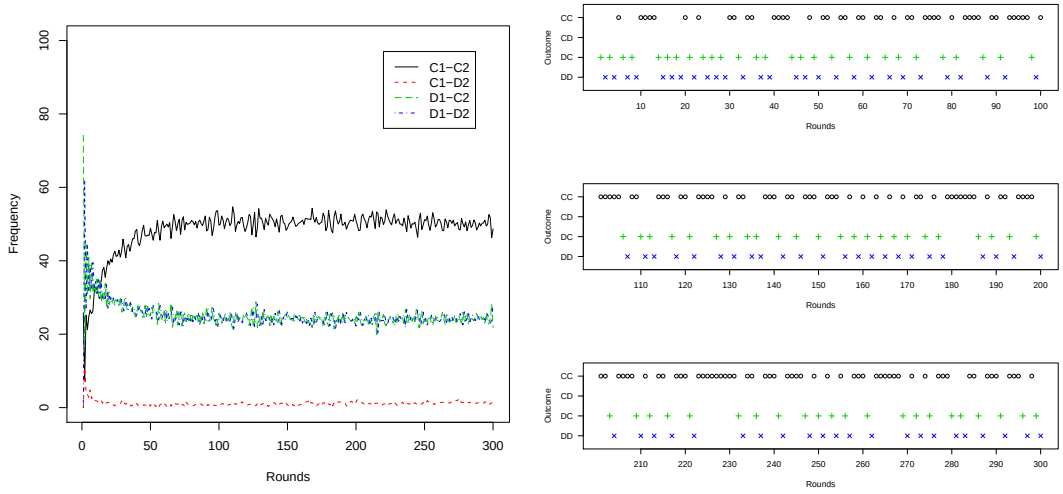
Figure 64: Frequency of outcomes as game progresses when MODEL 4 plays against (a) Random (RAN) strategy (b) Tit-for-Tat (TFT) strategy, decay = 0.5, noise = 0.2, L = 30, w = 0.1.



(a) (b)

Figure 65: Frequency of outcomes as game progresses when MODEL 4 plays against (a) Tit-for-Two-Tats (TFTT) strategy (b) Forgiving Tit-for-Tat (TFTF) strategy, decay = 0.5, noise = 0.2, L = 30, w = 0.1.

option with a small probability, but shifts back to defect since [cooperate, defect] is the least favorable outcome of the Prisoner's Dilemma game (Figure 63 (b)).



(a) (b)

Figure 66: Frequency of outcomes as game progresses (a) and outcome history (b) when MODEL 4 plays against Forgiving Tit-for-Tat (TFTF) strategy, decay = 0.5, noise = 0.14, L=30.

Similar to first three models, MODEL 4 chooses to defect with a high frequency against Random strategy. Playing defect is advantageous against Random player since cooperating may result in disadvantageous [cooperate, defect] outcome. By playing defect with a high probability, model can defend itself against defect moves. Although half of the defect moves are answered by defect by Random player leading to [defect, defect] outcome, player ensures gains through [defect, cooperate] outcome in other rounds. MODEL 4 learns to defect more frequently in the course of the iterated game and performs better than the Random strategy (Figure 64 (a)). Conditional cooperation probabilities against ALLC, ALLD are close to zero for MODEL 4. Like first model, forth model exhibits a Pavlovian type behavior against Random strategy, chooses to defect after asymmetric outcomes and cooperates with a probability close to 0.5 after symmetric outcomes (Appendix C).

Similar to first two models, MODEL 4 fails to attain cooperative behavior with teaching Tit-for-Tat strategy. TFT strategy never shifts to cooperation when the other player selected defect in the previous round. Due to this time lag in the behavior of

TFT strategy, forth model fails to recognize conditions of cooperation when playing with TFT strategy. Whenever MODEL 4 shifts to cooperate from defect move, the cooperative move is answered negatively by Tit-for-Tat strategy and disadvantageous [cooperate, defect] is realized. After observing the outcome, MODEL 4 decides going back to playing defect move, then advantageous [defect, cooperate] outcome is observed in the next round. The number of asymmetric outcomes are equal to each other since TFT strategy punishes every defect move of the other player. Therefore, MODEL 4 fails to attain a higher score than the TFT player. Both players do not succeed in coordinating their cooperate moves and receive negative payoffs as [defect, defect] becomes the dominant outcome in the iterated game (Figure 64 (b)).

When playing against Tit-for-Two-Tats strategy, model fails to attain cooperative equilibrium against Tit-for-Two-Tats strategy and [defect, defect] becomes the dominant outcome of the iterated game. Moreover, frequency of asymmetric outcomes decreases as game progresses. Unlike TFT case, frequency of asymmetric outcomes converge to different levels when MODEL 4 is playing against TFTT. Since frequency of [defect, cooperate] is higher than [cooperate, defect], model gets a higher score at the end of the game (Figure 65 (a)). Model exploits the reluctance of TFTT player in punishing defect moves, and chooses to defect with a higher probability throughout the game. In the second half of the game, the frequency of [cooperate, cooperate] outcome slightly increases. However, this trend is not observed when decay rate is set to 0.8 (Appendix F).

Compared to TFT, Forgiving Tit-for-Tat (TFTF) strategy is more forgiving towards defect moves. When playing with TFTF, MODEL 4 defects with a higher rate and receives a higher score at the end of the iterated game. Similar to first model, frequencies of [defect, cooperate] and [defect, defect] outcomes are almost forty percent and significantly higher than the frequencies of [cooperate, cooperate] and [cooperate, defect] outcomes throughout the iterated game (Figure 65 (b)). High frequency of [defect, cooperate] outcome ensures a high payoff for the model at the end of the game.

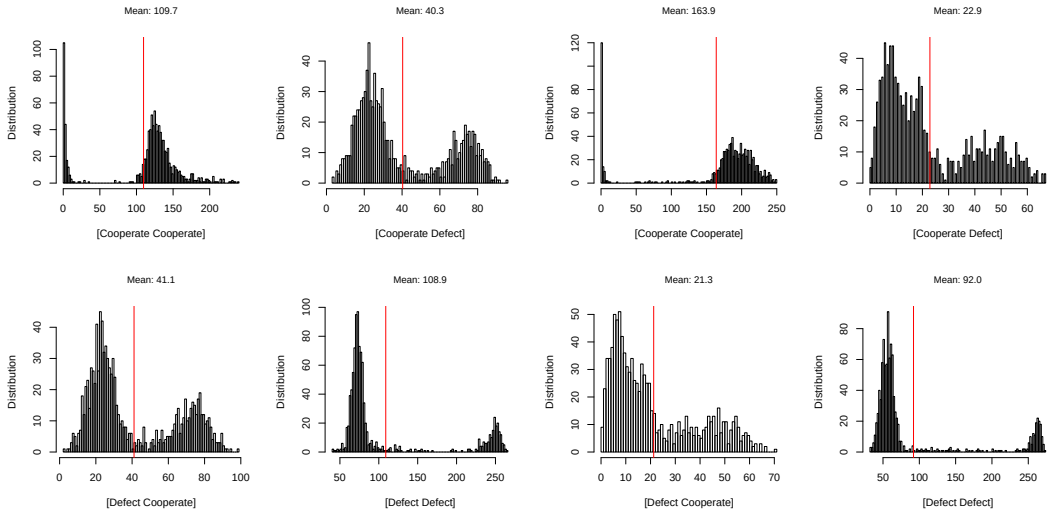
Pavlovian strategy cooperates after symmetric outcomes and defects after asym-

metric outcomes. MODEL 4 exhibits a similar performance to first model and is successful against Pavlovian strategy. Frequency of [cooperate, cooperate] is higher than [defect, defect] outcome throughout the game. Furthermore, disadvantageous [cooperate, defect] outcome is not experienced (Figure 66 (a)). Behavior of MODEL 4 player against Pavlovian strategy can be observed in Figure 66 (b). After cooperative outcome, the model either cooperates or defects which result in positive payoffs for the model. If the model chooses to defect and [defect, cooperate] is observed, model continues to defect while Pavlovian player also chooses to defect. As a result, [defect, defect] is observed which leads to a shift to cooperation for both players and the disadvantageous [cooperate, defect] outcome is almost never observed.

According to means of conditional cooperation probabilities MODEL 4 uses a Pavlovian type strategy against both Tit-for-Tat strategies and Pavlovian strategy. When playing with Forgiving Tit-for-Tat strategy, MODEL 4 defects after asymmetric outcomes and cooperates with a 1/2 probability after symmetric outcomes. Like MODEL 1, forth model employs a learning, Pavlovian type strategy against Pavlovian strategy. Conditional probabilities reveal that the model is more forgiving against Pavlovian player and cooperates with a higher probability after [defect, defect] is observed (Appendix C).

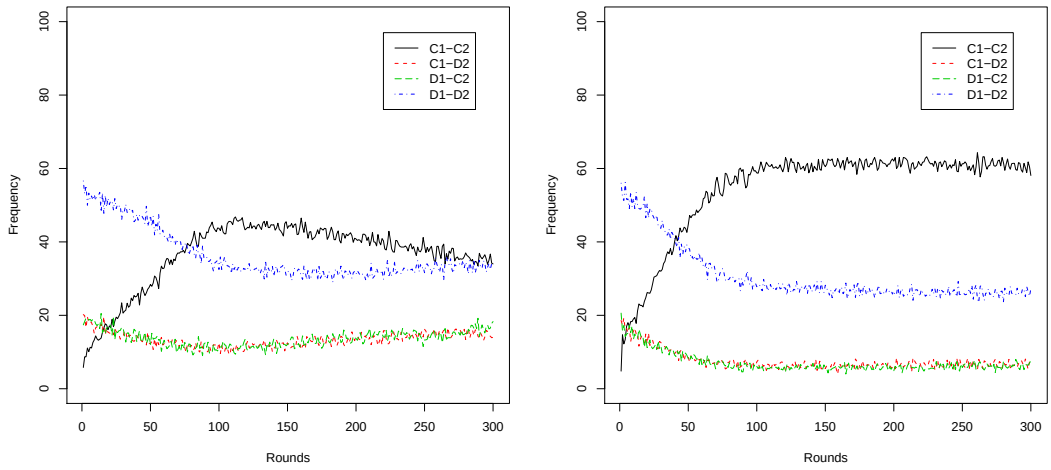
3.4.2 Model Behavior When MODEL 4 Plays with MODEL 4

MODEL 4 exhibits a performance similar to first model against basic strategies. Similarly, its behavioral characteristics against itself is similar to MODEL 1. Simulation results indicate two different types of results in a 300 round game. Iterated game results in either cooperative or defective equilibrium. Either [cooperate, cooperate] or [defect, defect] becomes the dominant strategy in iterated games. Moreover, frequency of asymmetric outcomes are very small compared to symmetric outcomes. A small number of iterated games have mixed outcomes (Figure 67 (a)). Similar to first model, mean number of cooperative outcomes is low in cooperative equilibria compared to experiments with human subjects. Simulation data can not be compared in terms of



(a) (b)

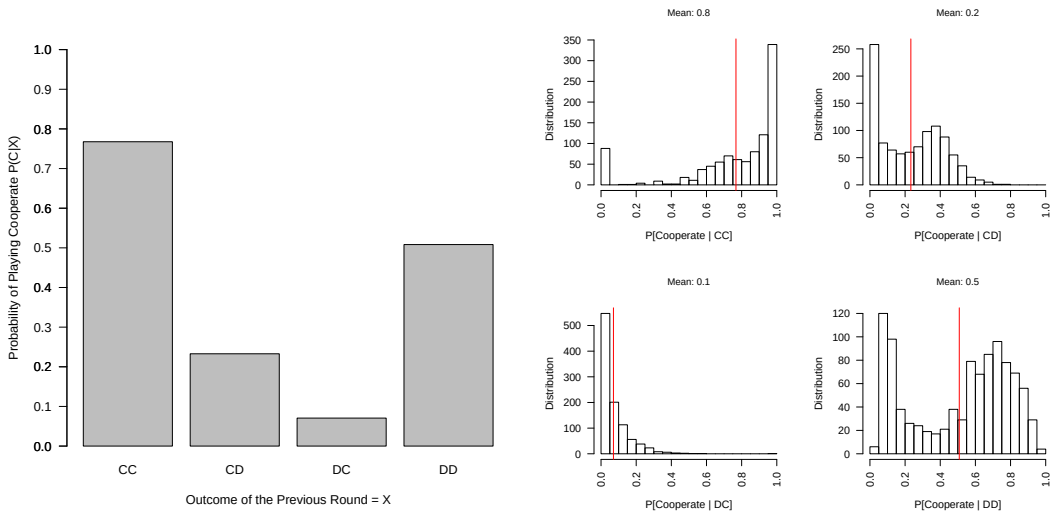
Figure 67: Distribution of outcomes when MODEL 4 plays against MODEL 4, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1.



(a) (b)

Figure 68: Frequency of outcomes as game progresses when MODEL 4 plays against MODEL 4, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1.

the distribution of mixed, cooperative and defect results since the experimental data is not extensive enough to provide information about distribution. When compared to first model, frequency of cooperative equilibrium is slightly higher for the forth model.

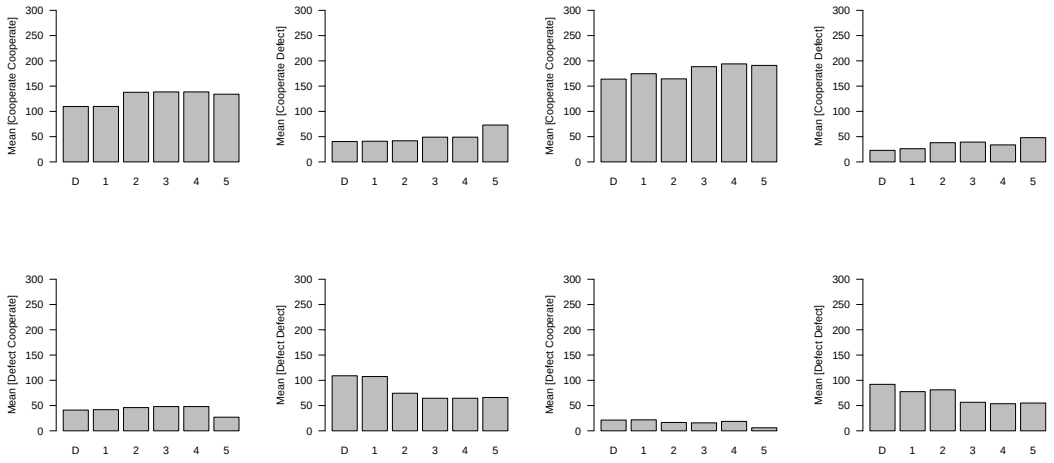


(a) (b)

Figure 69: Mean of cooperation probabilities (a) and distribution of cooperation probabilities (b) given the outcome of previous round when MODEL 4 plays against MODEL 4, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1.

When we increase the decay rate, performance of the model improves significantly. Both the frequency of cooperative equilibrium in 1000 simulations and the frequency of [cooperate, cooperate] outcomes in a cooperative equilibrium increase with respect to forgetting rate. Mean of [cooperate, cooperate] outcome in 300 rounds increases significantly from 109.7 to 163.9, meanwhile frequency of defective outcome decreases from 108.9 to 92. In addition to that, the frequencies of asymmetric outcomes decline significantly (Figure 67 (b)). This difference between two cases can be explained through the learning curves which depicts the change of mean frequencies as the game progresses.

At the beginning of the iterated game, frequency of [defect, defect] is the highest whereas the frequency of [cooperate, cooperate] is less than ten percent as expected. In later rounds, defection frequency decreases while the frequency of cooperative outcome increases. Although the frequencies of asymmetric outcomes decline for a while, they settle to their initial levels while the frequency of cooperative outcome starts to



(a) (b)

Figure 70: Outcome frequencies with respect to cooperative outcomes in initial rounds when MODEL 4 plays against MODEL 4, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1.

decrease after initial stages. Since decay rate is low in the first experiment, high increase in cooperative outcome frequency and decline in the frequencies of asymmetric outcomes are temporary. Similar to first model, model recalls asymmetric outcomes with a higher probability due to low forgetting rate which results in selecting defect move with a high probability (Figure 68 (a)). When decay rate is higher, retrieval probabilities of asymmetric outcomes decrease significantly as the player recalls recent outcomes better. Therefore, model recalls [cooperate, cooperate] and [defect, defect] as the most likely outcome of making cooperate and defect moves, respectively. As a result, model plays cooperate move with a higher probability (Figure 68 (b)).

Conditional cooperation probability structure for the forth model is similar to first model, and resembles a Pavlovian type of behavior which cooperates after cooperative and defect outcomes with probabilities of 0.8 and 0.5 respectively. Player defects with a higher probability after asymmetric outcomes, though it is more forgiving than the Pavlovian player after [cooperate, defect] outcome (Figure 69 (a)). Like first and third models, distribution of conditional cooperation probabilities in 1000 iterated game reveal that there are two distinct type of players, according to their behavior after [defect, defect] outcome. First type exercises a Pavlovian behavior and cooperates with a high

probability, whereas the second type exhibits Tit-for-Tat characteristics and defects after [defect, defect] with a higher probability (Figure 69 (b)).

In order to investigate the effects of game history and cooperation at initial rounds, an analysis of simulation data for MODEL 4 is provided in Figure 70. According to Figure 70, repeated cooperation in early rounds does not have an effect on outcome frequencies. Additional simulation data and other analysis tools are required fora more detailed investigation of the relationship between game history and outcome frequencies.

Simulation results for MODEL 4 reveal a close similarity between first and forth models in terms of model behavior and performance, despite the differences in model architecture and decision making processes. Association mechanism seem to be ineffective in decision making. One possible explanation is inadequacy of association w in modification of activation levels of outcome chunks. Therefore, additional exploration in parameter space is required. Another explanation is that associations between goal and memory chunks are not reinforced enough during iterated game. This possibility can be investigated through Iterated Prisoner's Dilemma simulations with different number of rounds.

3.5 Summary

Simulation results for memory models are presented in this chapter. Performance and behavioral characteristics depends on game history and model parameters such as decay rate and noise level. Therefore, parameter space for each model is systematically explored in order to determine the optimal parameter values.

After parameter setting, performance of memory models are evaluated according to simulated experiments. Each model plays Iterated Prisoner's Dilemma Game with basic strategies and against itself for 1000 simulations of 300 rounds of iterated game. The case where models play the iterated game against each other is not investigated. Behavioral characteristics, learning patterns and model performance of all models against basic strategies and against themselves are illustrated by detailed plots.

All simulations are conducted according to a specific payoff matrix which is also used in early ACT-R models and human experiments. Model performance and sensitivity with respect to different payoff values are not investigated in this study. Moreover, decision making processes depends on a qualitative comparison between payoff values. As a result quantitative analysis and decision processes are not investigated. Model evaluation is based on 1000 simulations of 300 rounds of iterated games in order to enable comparison with earlier studies. Alteration of iteration length may affect model performance and behavior patterns. Therefore, further research is required to examine sensitivity of model performance to payoff values and number of iterated games.

4 CONCLUSION

This study investigated four basic ACT-R based memory models. These models are built upon similar memory and procedural processes. However, they differ in terms of information extraction from game history. Moreover, there are differences in their decision making processes which enable the models to employ memory entries in different manners. Performance and behavioral characteristics of each model depends on decision processes, outcome history and model parameters. Parameter values are chosen according to resemblance to human behavior in terms of distribution of outcomes and mean outcome frequencies. After parameter setting, memory models are evaluated in terms of their behavior against basic Iterated Prisoner's Dilemma strategies and against themselves :

- All models were successful in detecting unconditionality in behaviors of All-Cooperate, All-Defect and Random strategies. Models learned to develop behavioral patterns in order to exploit All-Cooperate and Random strategies. Moreover, they were able to defend themselves against All-Defect strategy.

- Performance of the memory models are evaluated against basic conditional strategies. First, second and forth models were not able to attain cooperative equilibrium with Tit-for-Tat strategy. Third model, on the other hand, learned to cooperate with TFT strategy.

- Against Tit-for-Two-Tats strategy all models fail to reach cooperative equilibrium. However, apart from second model, all models learned to exploit the reluctance of Tit-for-Two-Tats strategy in punishing defect moves.

- Against Forgiving Tit-for-Tat strategy, only third model achieved cooperative equilibrium. Other models learned to exploit Forgiving Tit-for-Tat strategy and performed better than this strategy.

- When playing against Pavlovian player, third model managed to reach cooperative equilibrium in a significant portion of the games. First, second, forth mod-

els adapted their behavior in order to exploit the behavioral pattern exhibited by Pavlovian player and attained higher scores than Pavlovian player in almost all experiments.

- Although first, second and forth models presented learning behavior against teaching Tit-for-Tat strategies, the models failed to exhibit teaching patterns against learning Pavlovian strategy.

- All agents presented learning behavior against basic strategies.

After evaluating model performance against basic strategies, we investigated the behavior of models for the case where the model plays against itself:

- All four models were successful in reaching cooperative equilibrium in a significant portion of the games. However, first and forth models exhibited cooperative equilibrium with a higher frequency.

- Apart from second model, all models exhibited a learning pattern consistent with human subjects. Frequency of cooperative equilibrium increased as the iterated game progresses. Moreover, when first and forth models were playing against themselves, a decrease in the frequency of asymmetric outcomes is also observed. This finding is consistent with human behavior.

- First, third and forth models, agents can be classified into two groups according to their behavior after defective outcome. One class of agents exhibited a Forgiving Tit-for-Tat like behavior, whereas other agents adopted a more Pavlovian strategy.

- Finally, repeated cooperation at initial rounds affects mean frequency of outcomes for second and third models. However, relationship with early cooperation and outcome frequencies seems to be complex.

4.1 Future Work

Simulation study can be developed in several ways. First, model excludes emotional processing in decision making. Decision making in the model depends only on the relative ordering of the payoff values for all models except third model. Therefore, actual payoff values does not have any effect on the performance of these models. Models can be modified in certain ways in order to test the impact of different payoff matrices of Prisoner's Dilemma games.

Second, in this study number of iterated rounds are limited by three hundred. Model behavior with respect to number of iterated rounds can be investigated by additional simulations that can reveal if model behavior is stable or subject to change when game length for iterated game increases.

Third, model performance depends on game history and relationship between early game history and model behavior seems to be complex. Therefore additional simulations and analysis tools are required in order to investigate impact of early cooperation and game results on outcome distribution and model behavior.

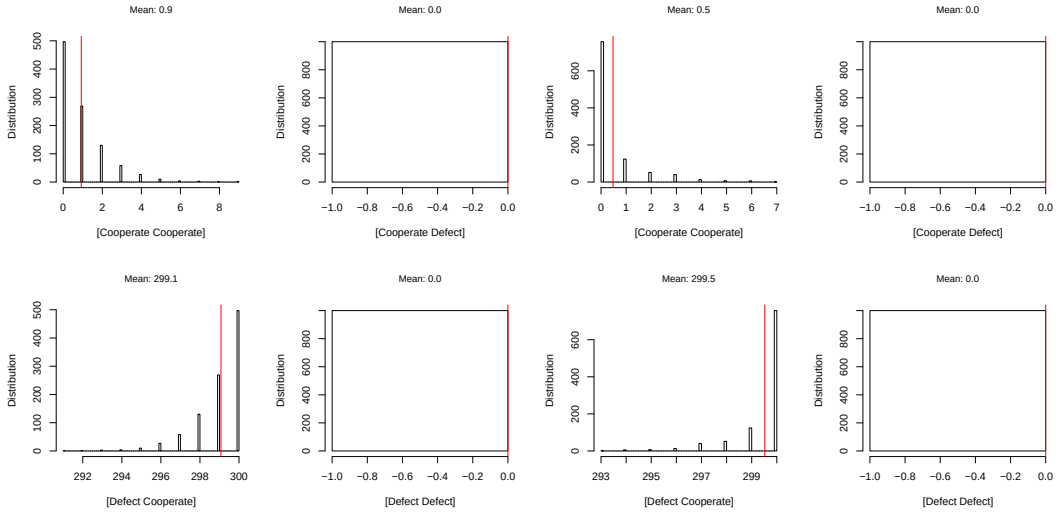
Fourth, ACT-R memory models can be tested in Multi Agent environments. Multi Agent settings can be developed in order to explore a wider parameter space for the models. Moreover, different evolving cognitive models and architectures can be implemented in order to evaluate model performances against each other.

Fifth, different memory models can be tested against human players. Memory models may be valuable in order to investigate complex human behavioral patterns and assessing human performance against learning models.

Finally, ACT-R Memory models can easily be modified for other 2x2 games such as Snowdrift and Zero-Sum games. Moreover, models can be used in order to investigate learning for more complex games such as Mixed-Strategy games. Basic decision making mechanism and model structure may also adopted for more complex Prisoner's Dilemma games with more than two players and with multiple action space.

A Distribution of Outcomes

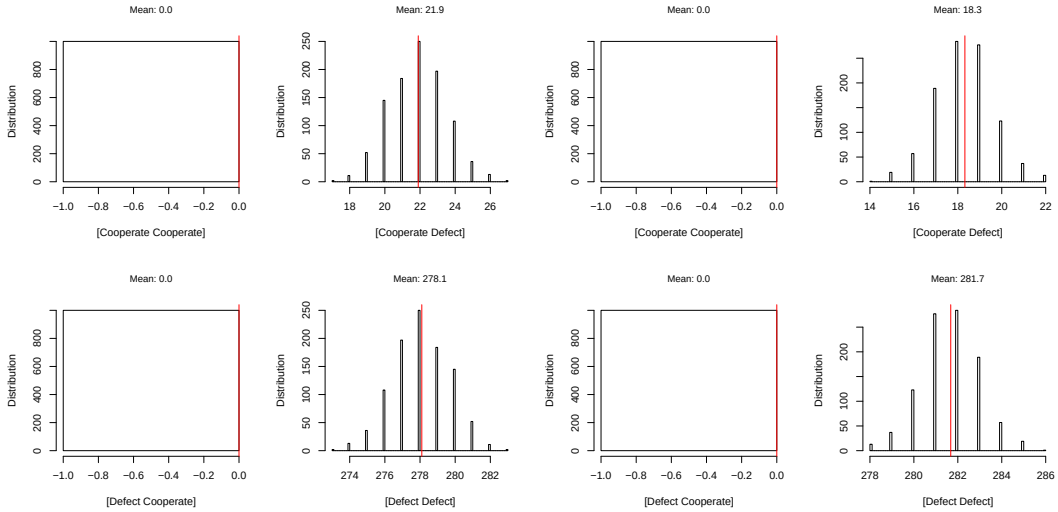
MODEL 1 vs. ALLC



(a) (b)

Distribution of outcomes when MODEL 1 plays against ALLC strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

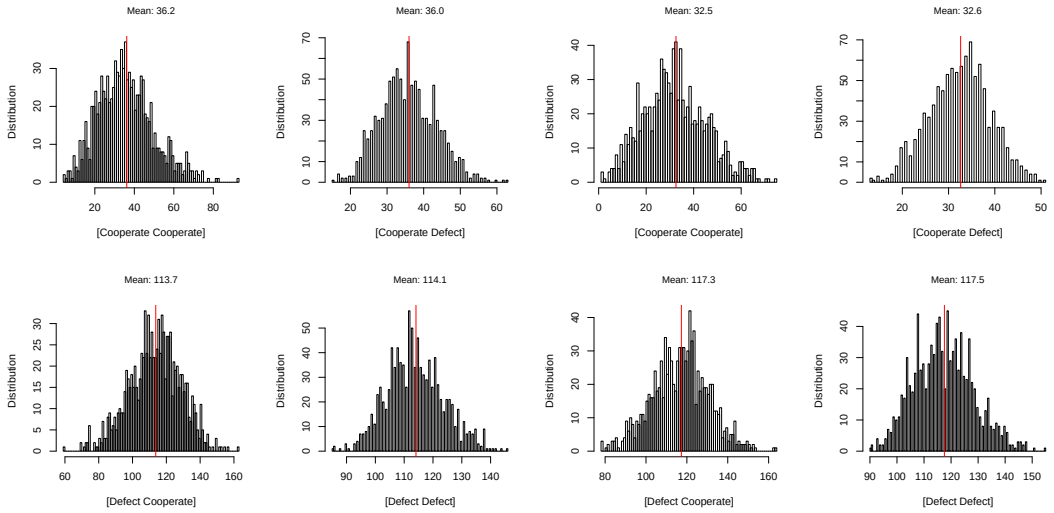
MODEL 1 vs. ALLD



(a) (b)

Distribution of outcomes when MODEL 1 plays against ALLD strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

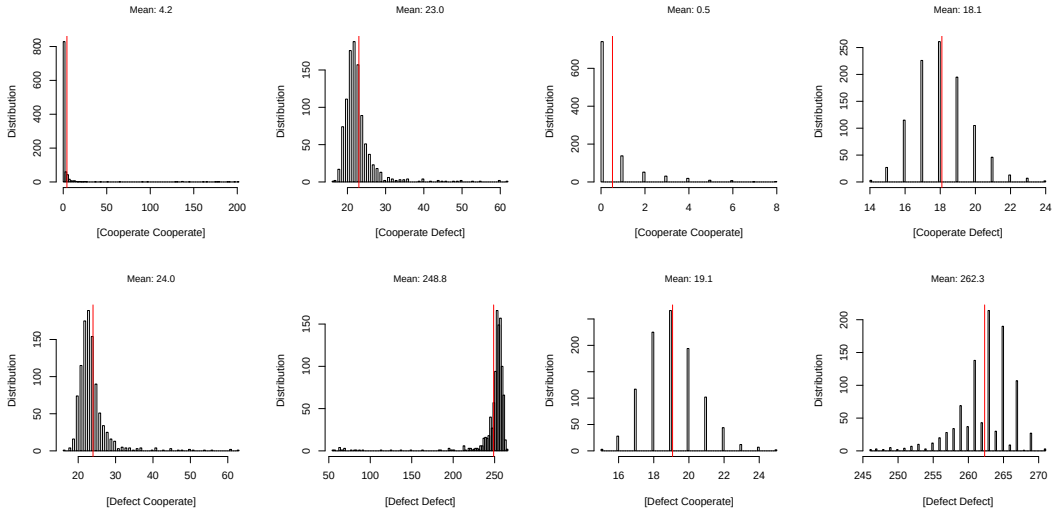
MODEL 1 vs. RAN



(a) (b)

Distribution of outcomes when MODEL 1 plays against RAN strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

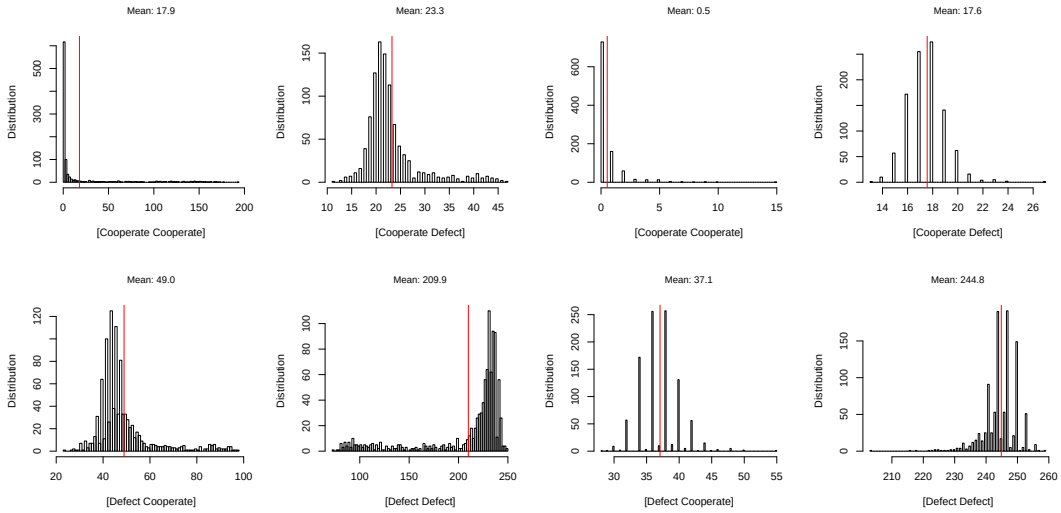
MODEL 1 vs. TFT



(a) (b)

Distribution of outcomes when MODEL 1 plays against TFT strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

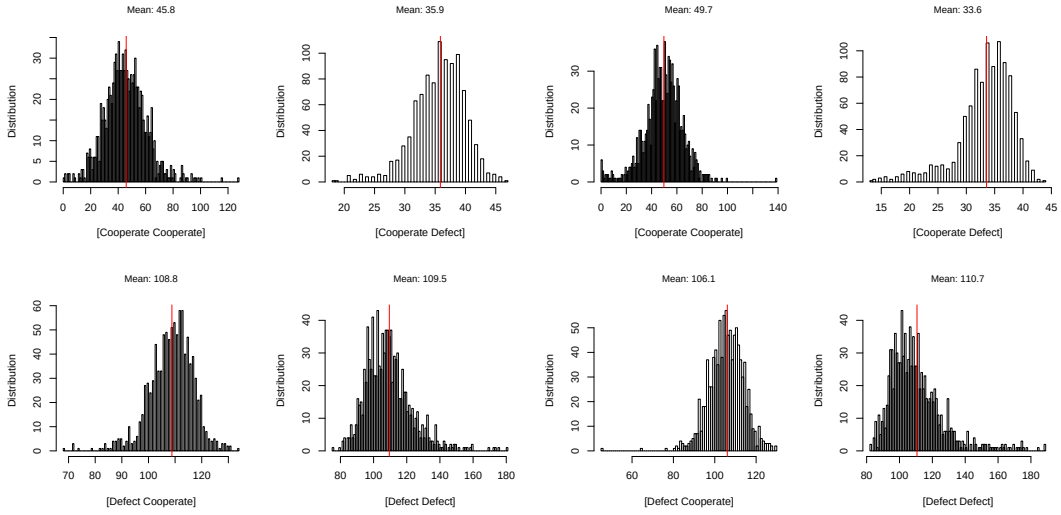
MODEL 1 vs. TFTT



(a) (b)

Distribution of outcomes when MODEL 1 plays against TFTT strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

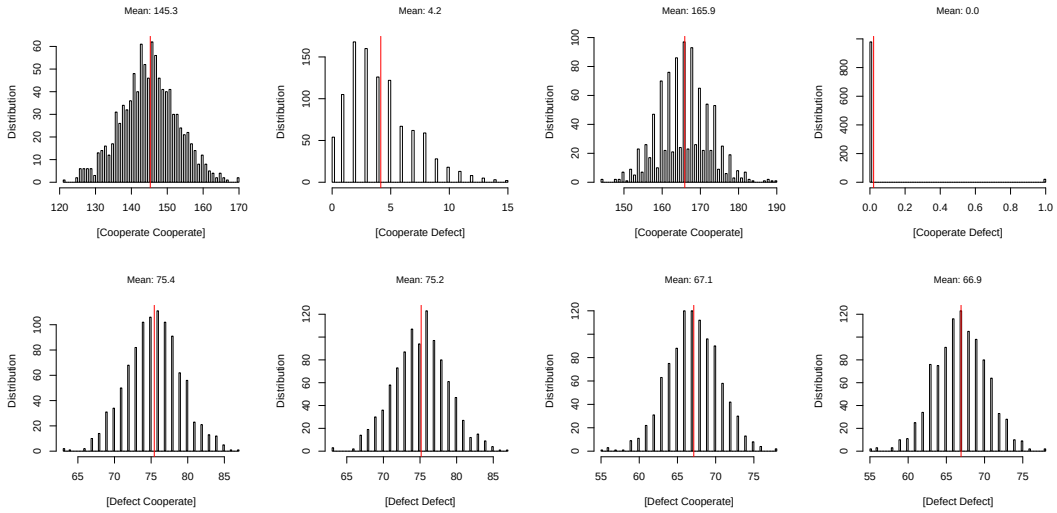
MODEL 1 vs. TFTF



(a) (b)

Distribution of outcomes when MODEL 1 plays against TFTF strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

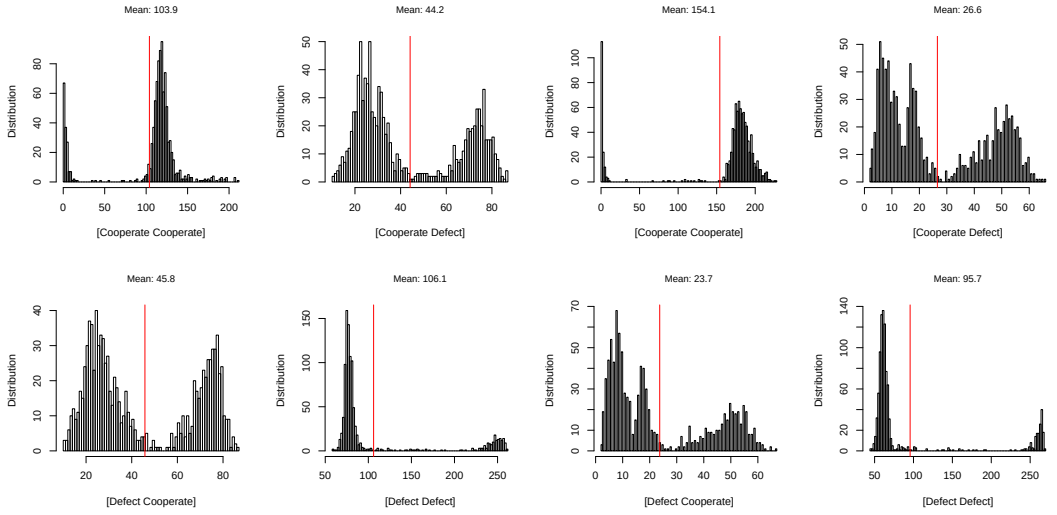
MODEL 1 vs. PAV



(a) (b)

Distribution of outcomes when MODEL 1 plays against PAV strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

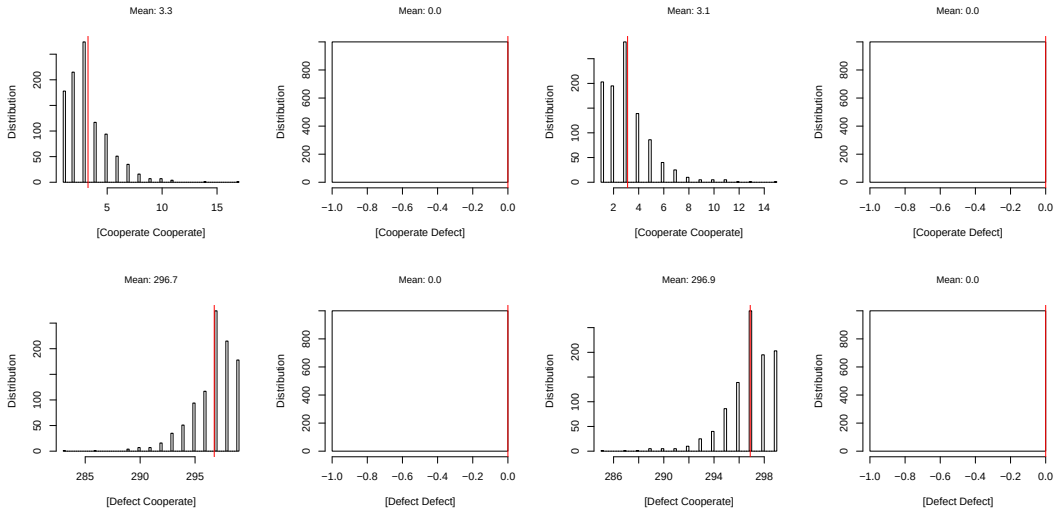
MODEL 1 vs. MODEL 1



(a) (b)

Distribution of outcomes when MODEL 1 plays against MODEL 1, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

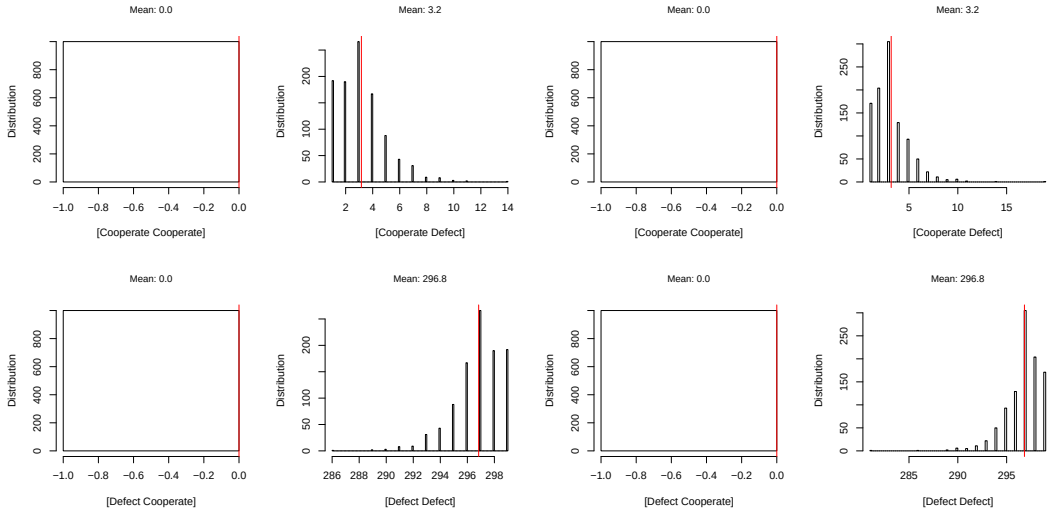
MODEL 2 vs. ALLC



(a) (b)

Distribution of outcomes when MODEL 2 plays against ALLC strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

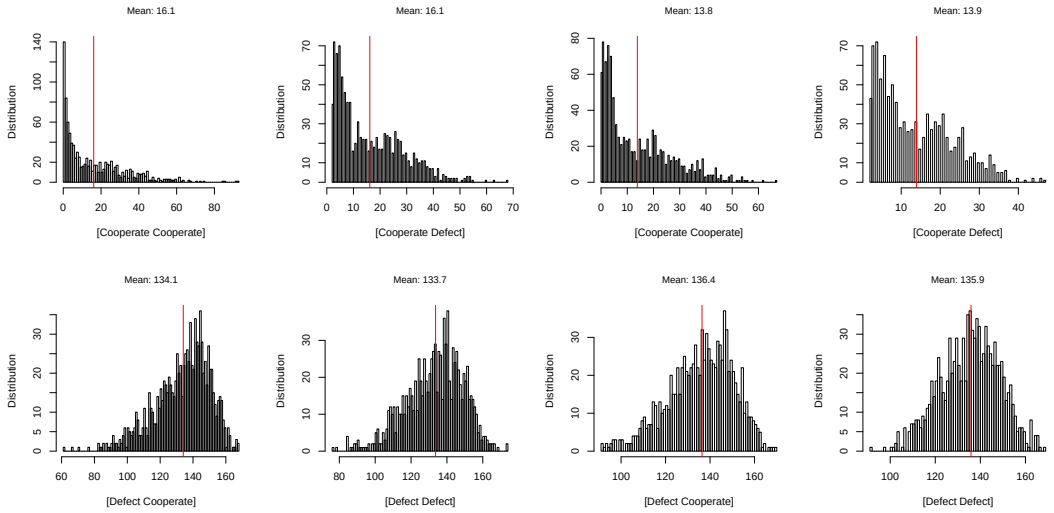
MODEL 2 vs. ALLD



(a) (b)

Distribution of outcomes when MODEL 2 plays against ALLD strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

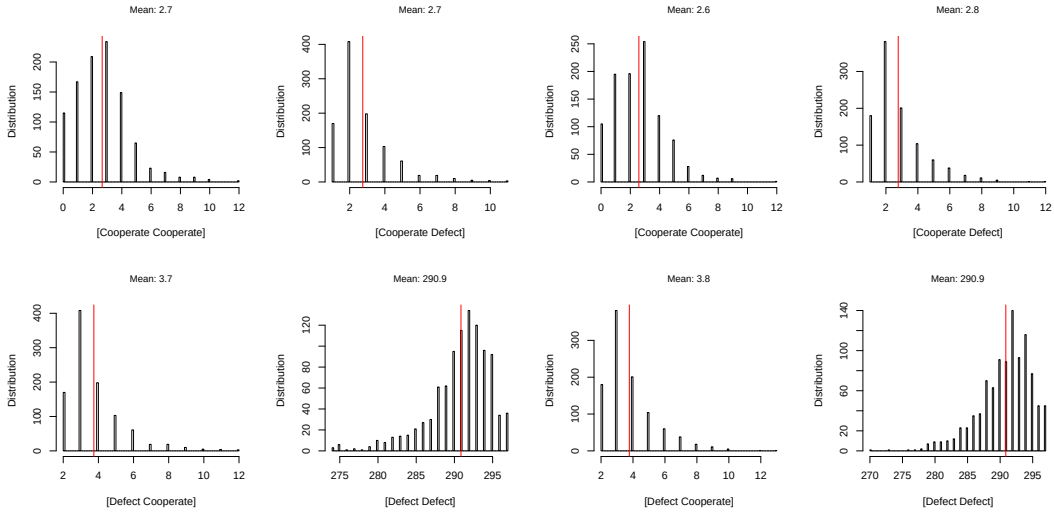
MODEL 2 vs. RAN



(a) (b)

Distribution of outcomes when MODEL 2 plays against RAN strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

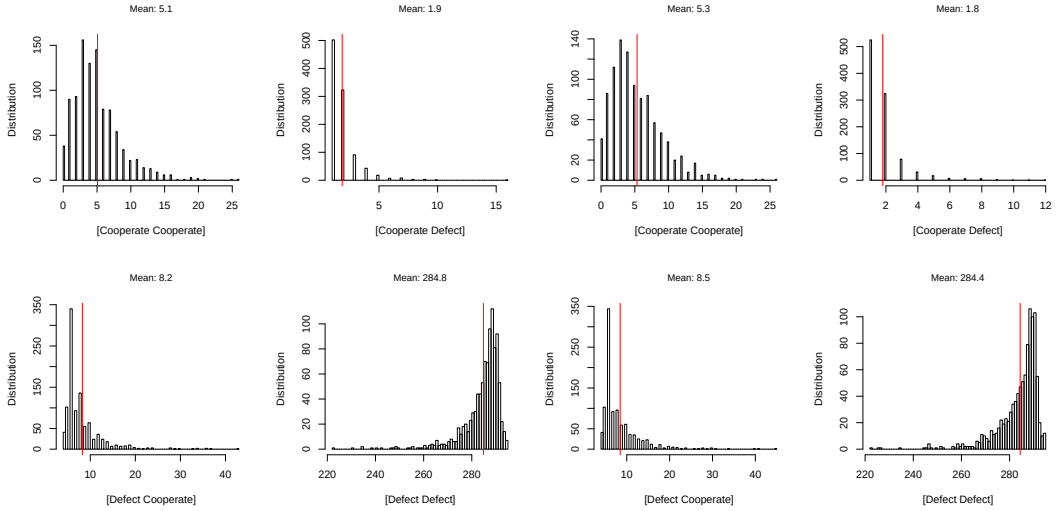
MODEL 2 vs. TFT



(a) (b)

Distribution of outcomes when MODEL 2 plays against TFT strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

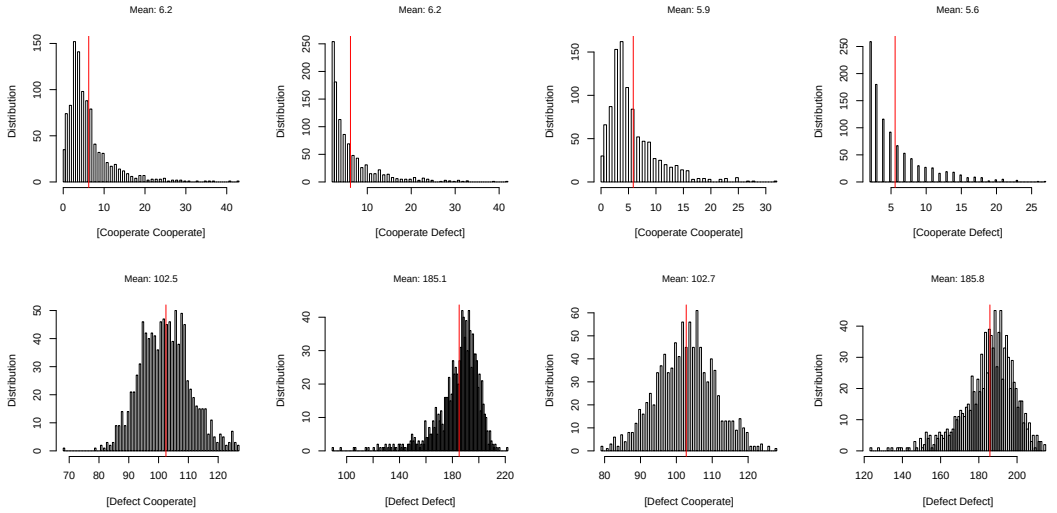
MODEL 2 vs. TFTT



(a) (b)

Distribution of outcomes when MODEL 2 plays against TFTT strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

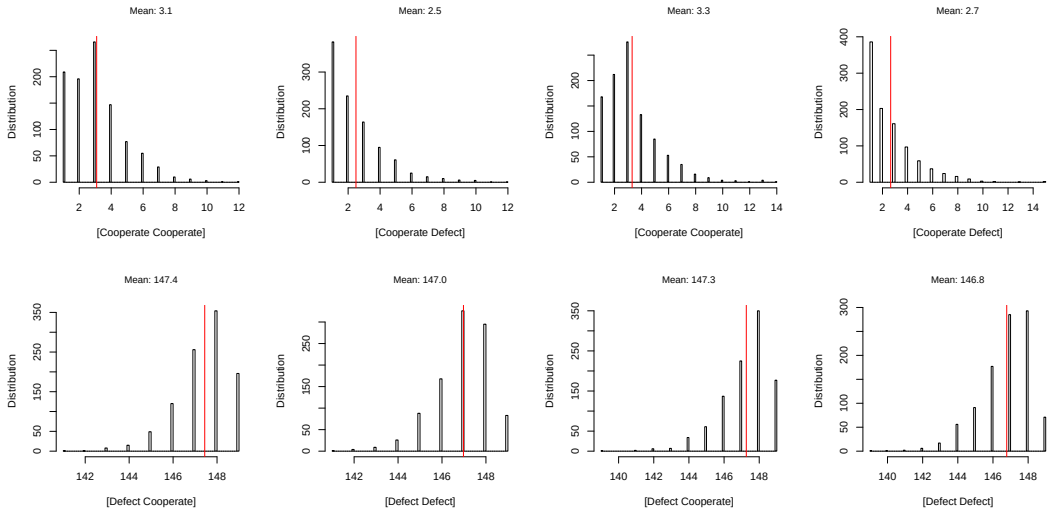
MODEL 2 vs. TFTF



(a) (b)

Distribution of outcomes when MODEL 2 plays against TFTF strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

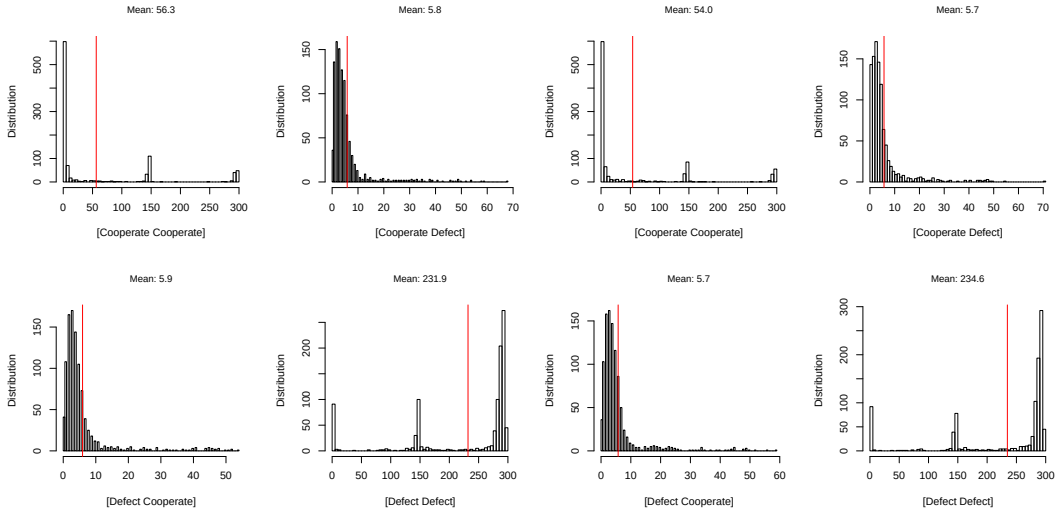
MODEL 2 vs. PAV



(a) (b)

Distribution of outcomes when MODEL 2 plays against PAV strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

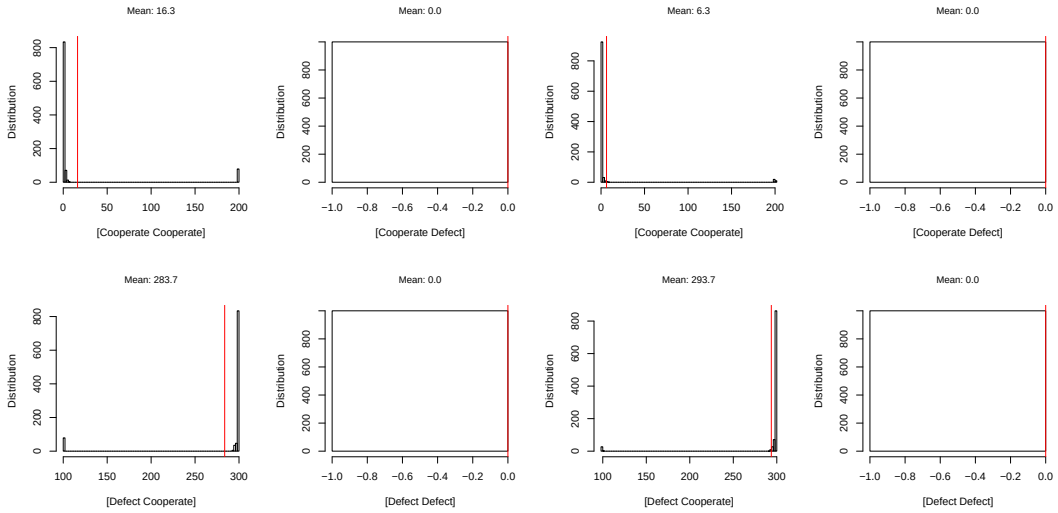
MODEL 2 vs. MODEL 2



(a) (b)

Distribution of outcomes when MODEL 2 plays against MODEL 2, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

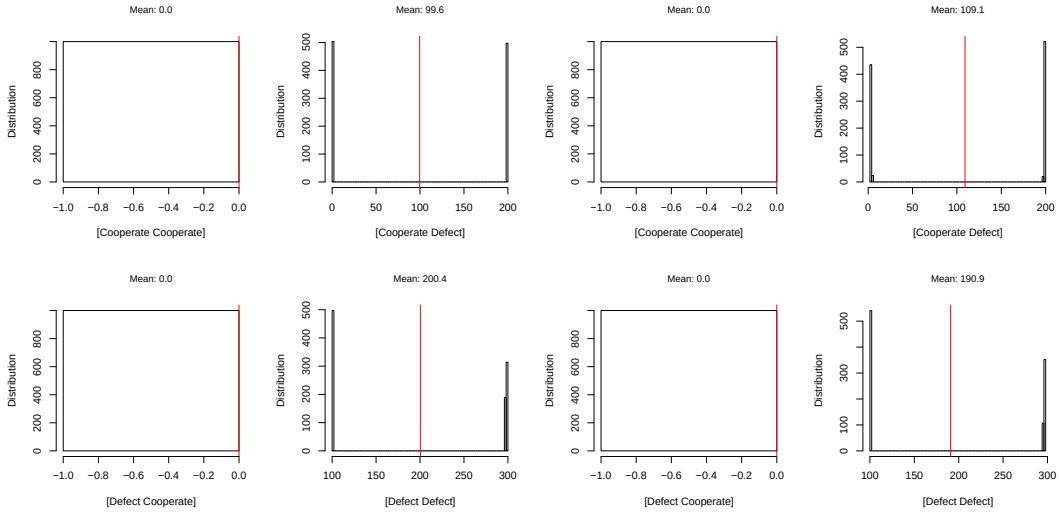
MODEL 3 vs. ALLC



(a) (b)

Distribution of outcomes when MODEL 3 plays against ALLC strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

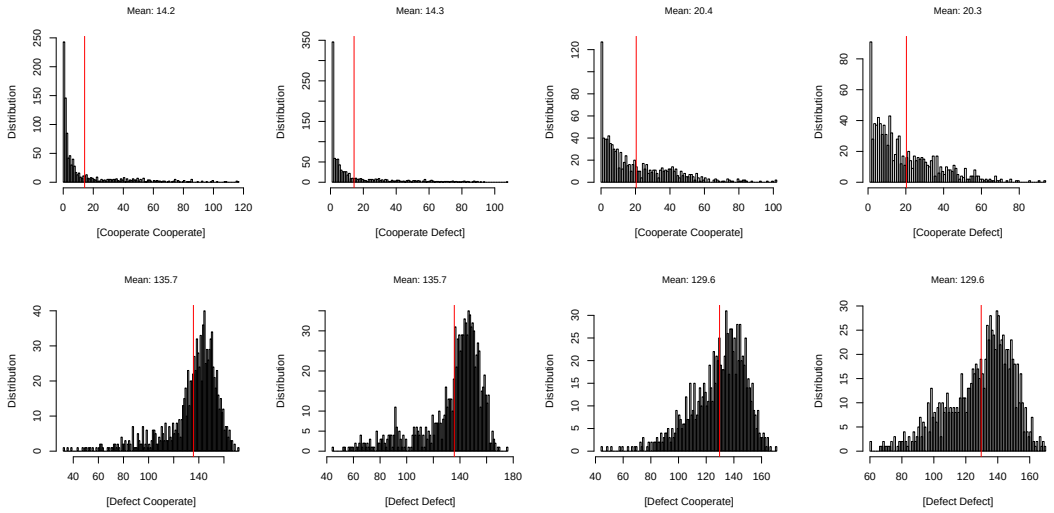
MODEL 3 vs. ALLD



(a) (b)

Distribution of outcomes when MODEL 3 plays against ALLD strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

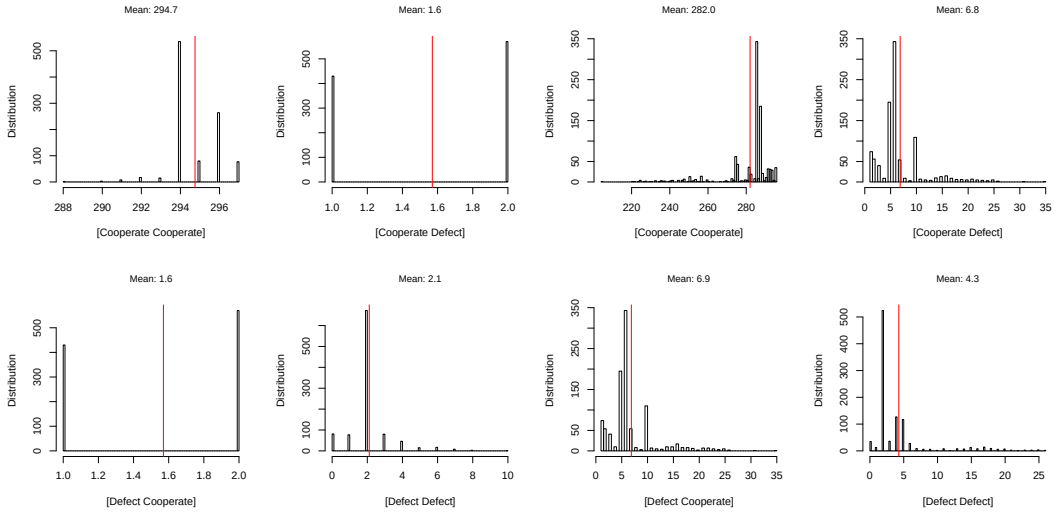
MODEL 3 vs. RAN



(a) (b)

Distribution of outcomes when MODEL 3 plays against RAN strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

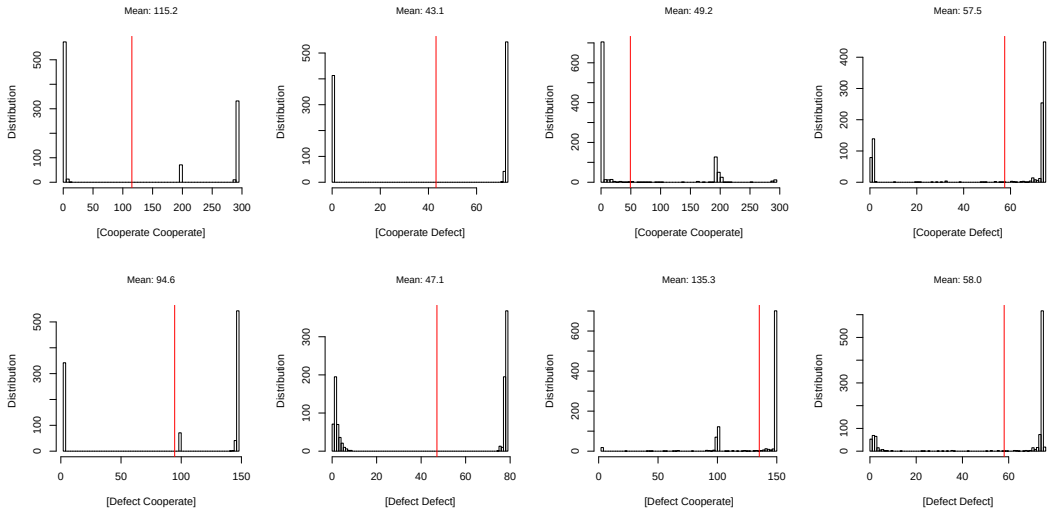
MODEL 3 vs. TFT



(a) (b)

Distribution of outcomes when MODEL 3 plays against TFT strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5 (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

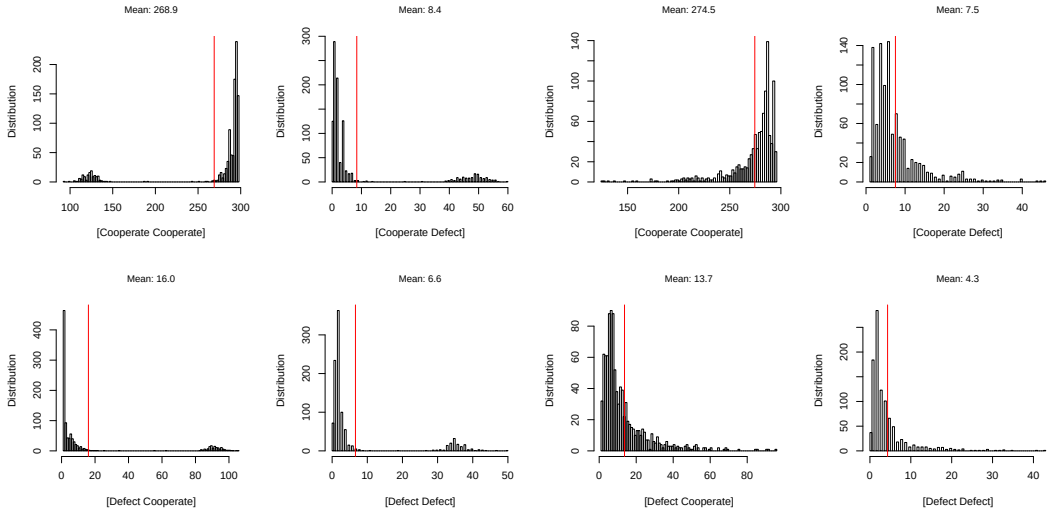
MODEL 3 vs. TFTT



(a) (b)

Distribution of outcomes when MODEL 3 plays against TFTT strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

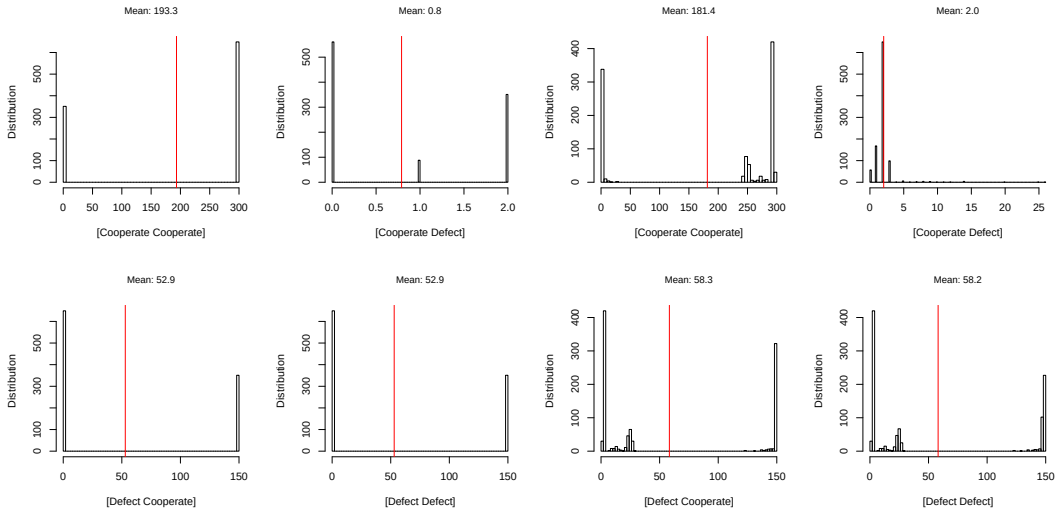
MODEL 3 vs. TFTF



(a) (b)

Distribution of outcomes when MODEL 3 plays against TFTF strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

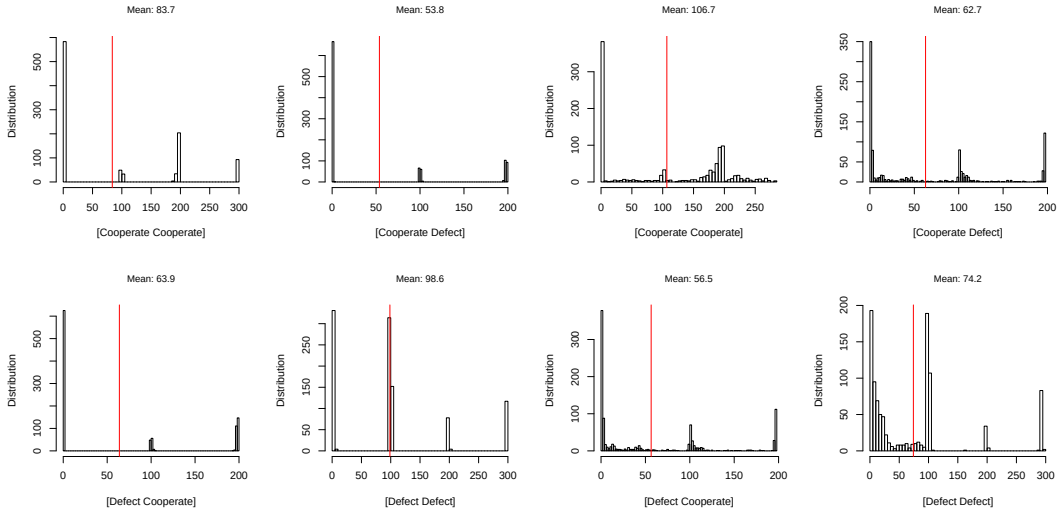
MODEL 3 vs. PAV



(a) (b)

Distribution of outcomes when MODEL 3 plays against PAV strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

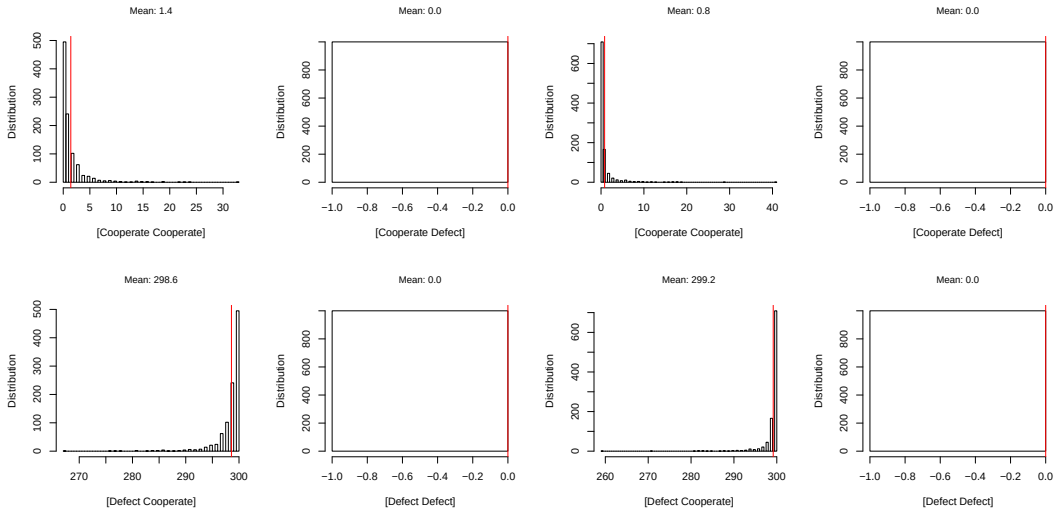
MODEL 3 vs. MODEL 3



(a) (b)

Distribution of outcomes when MODEL 3 plays against MODEL 3, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

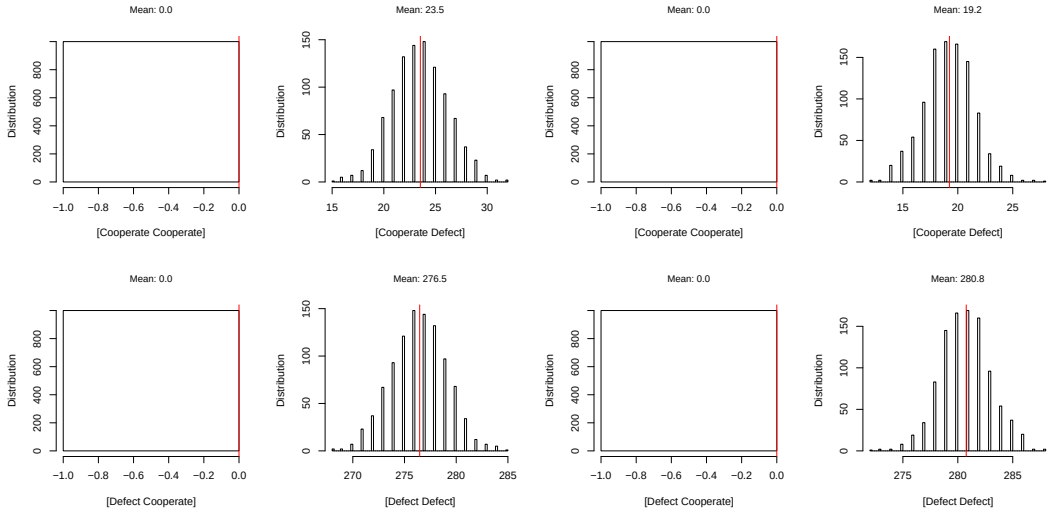
MODEL 4 vs. ALLC



(a) (b)

Distribution of outcomes when MODEL 4 plays against ALLC strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

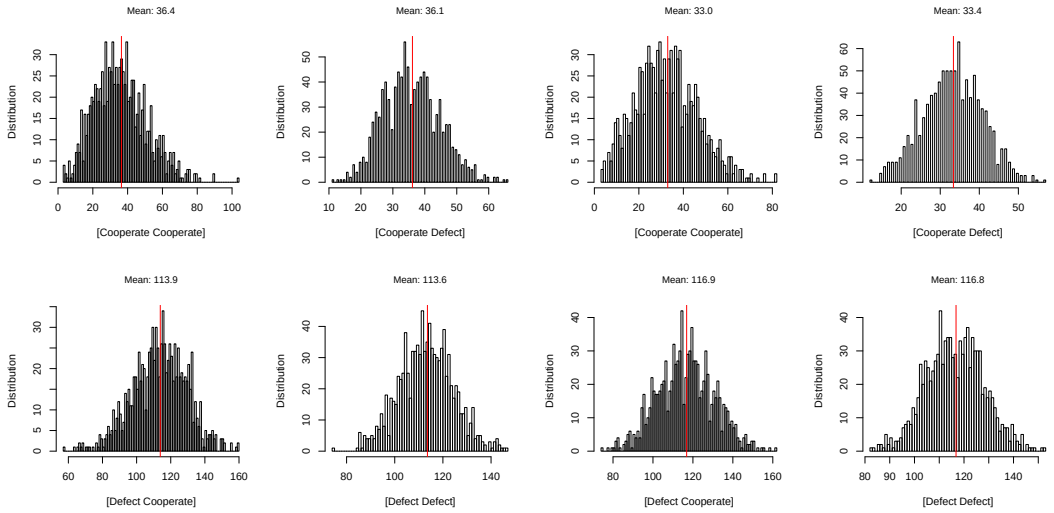
MODEL 4 vs. ALLD



(a) (b)

Distribution of outcomes when MODEL 4 plays against ALLD strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w= 0.1

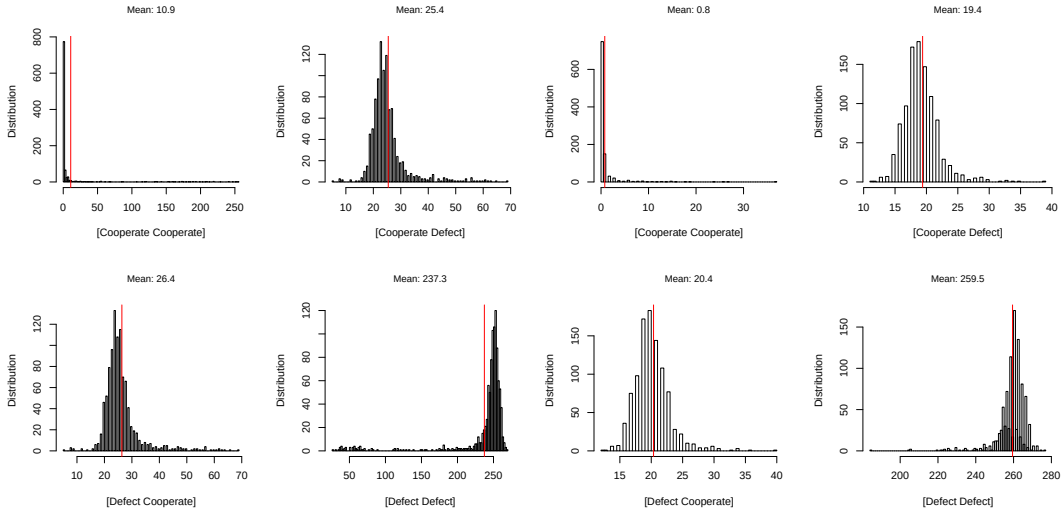
MODEL 4 vs. RAN



(a) (b)

Distribution of outcomes when MODEL 4 plays against RAN strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

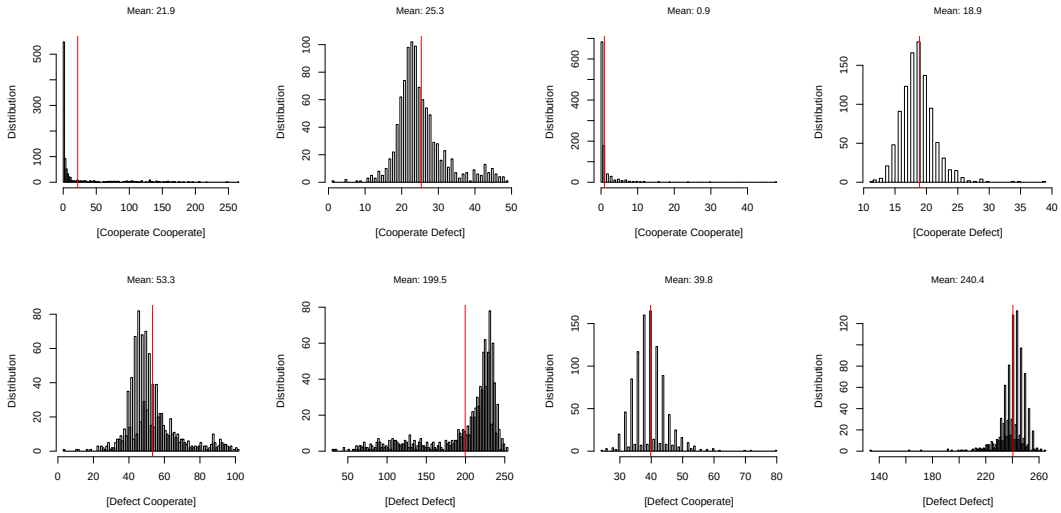
MODEL 4 vs. TFT



(a) (b)

Distribution of outcomes when MODEL 4 plays against TFT strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

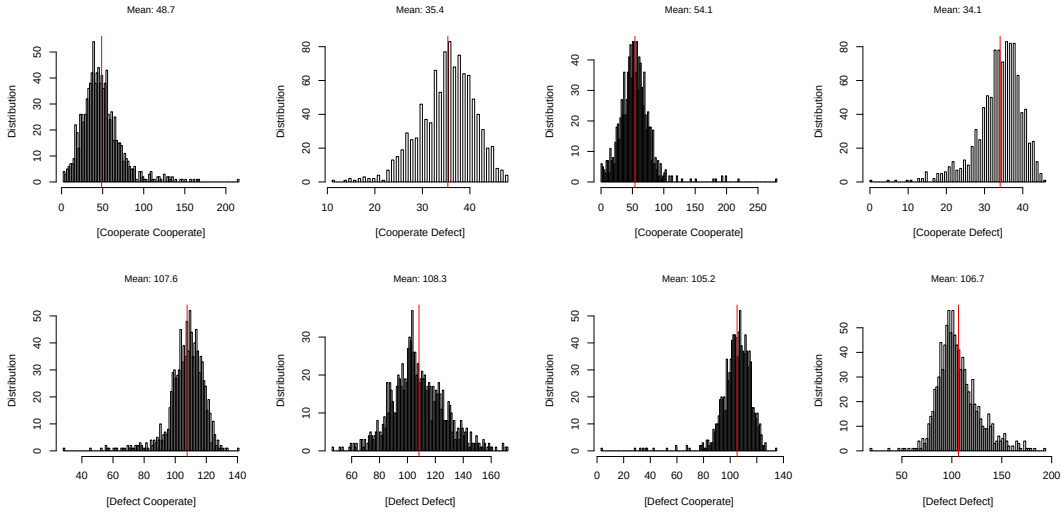
MODEL 4 vs. TFTT



(a) (b)

Distribution of outcomes when MODEL 4 plays against TFTT strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

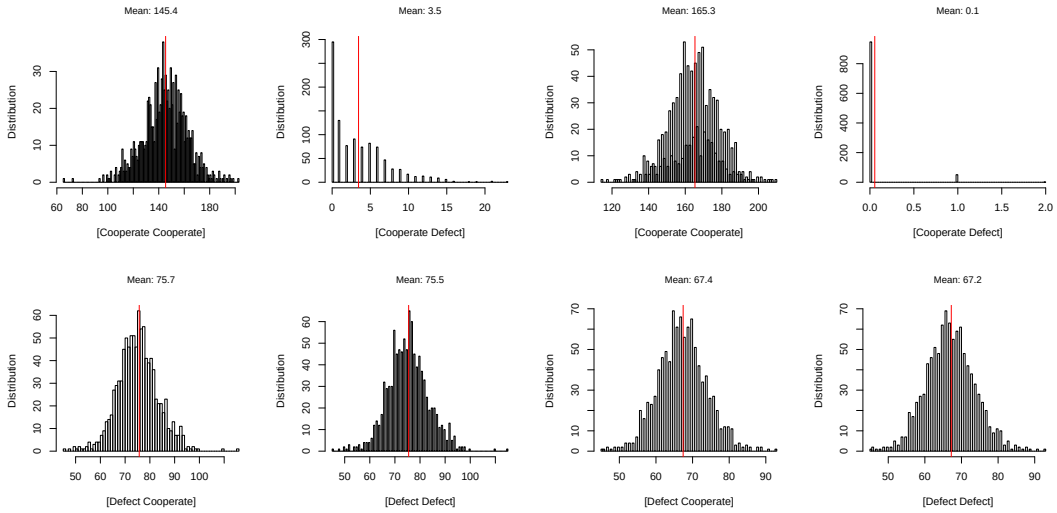
MODEL 4 vs. TFTF



(a) (b)

Distribution of outcomes when MODEL 4 plays against TFTF strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

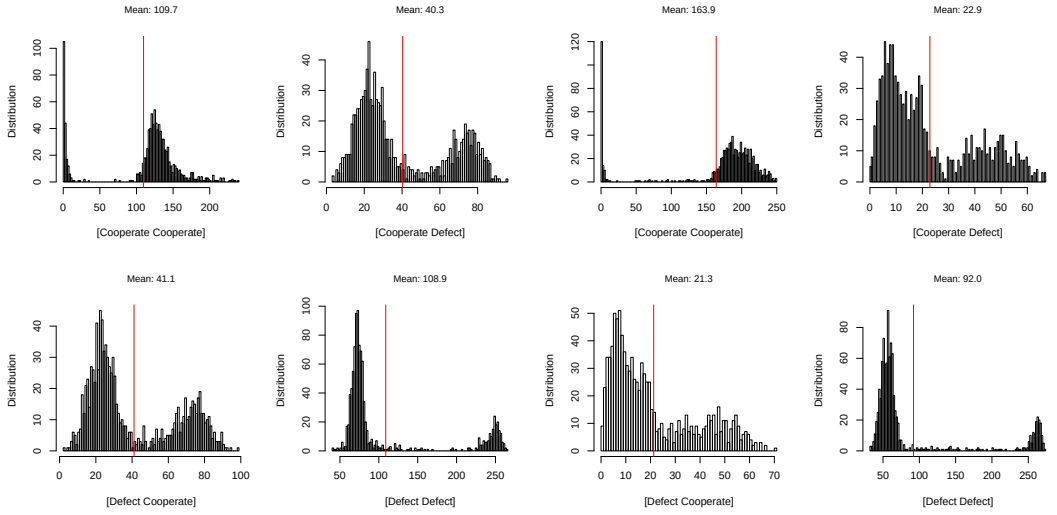
MODEL 4 vs. PAV



(a) (b)

Distribution of outcomes when MODEL 4 plays against PAV strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

MODEL 4 vs. MODEL 4

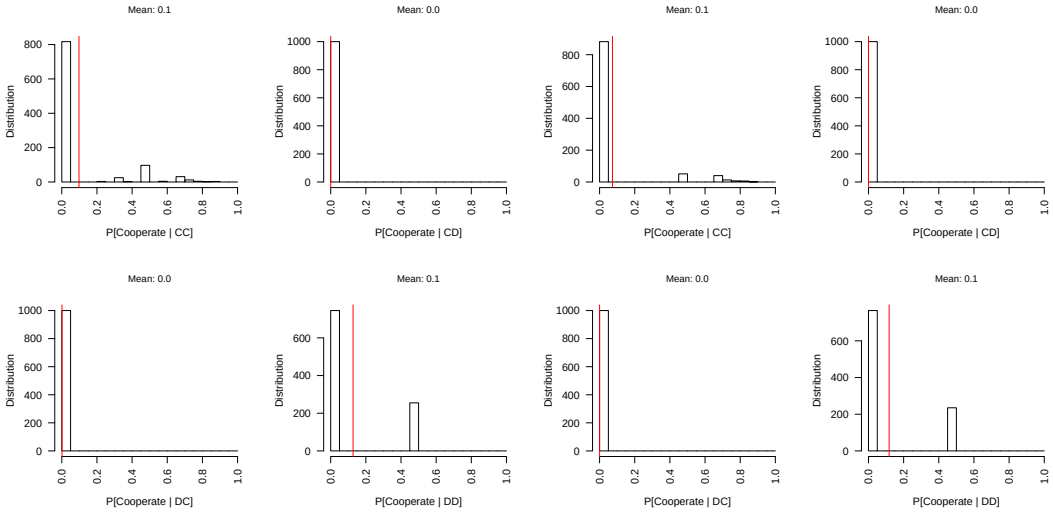


(a) (b)

Distribution of outcomes when MODEL 4 plays against MODEL 4, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

B Distribution of Conditional Cooperation Probabilities

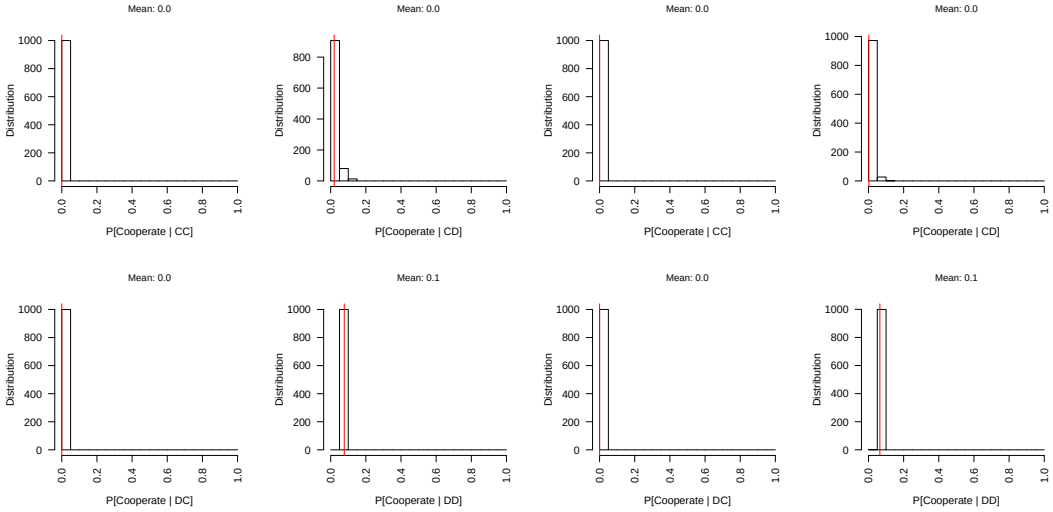
MODEL 1 vs. ALLC



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 1 plays against ALLC strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

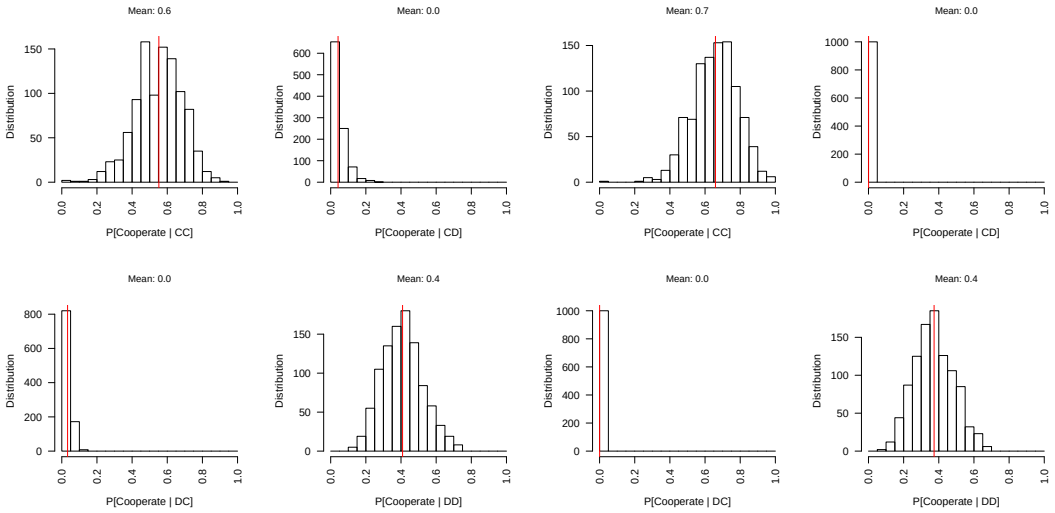
MODEL 1 vs. ALLD



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 1 plays against ALLD strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

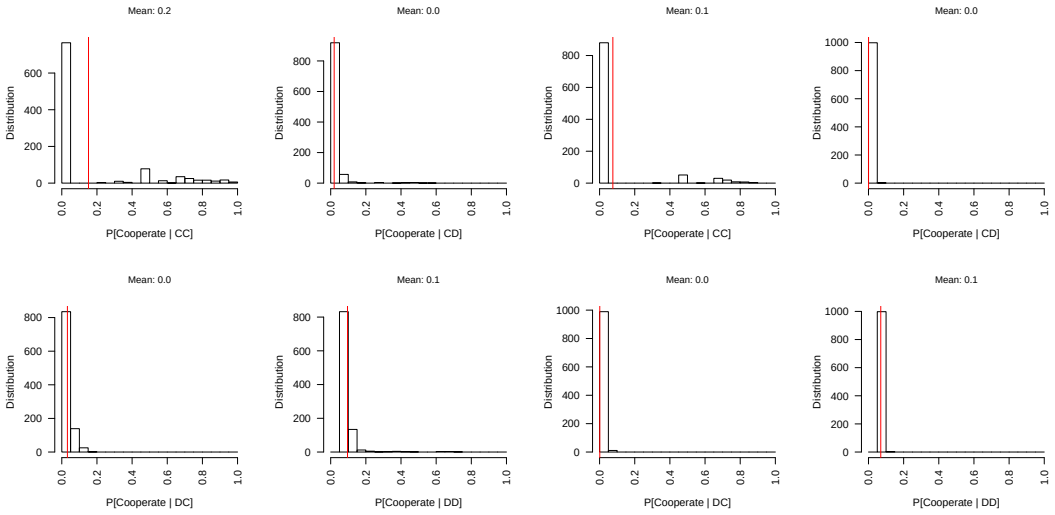
MODEL 1 vs. RAN



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 1 plays against RAN strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

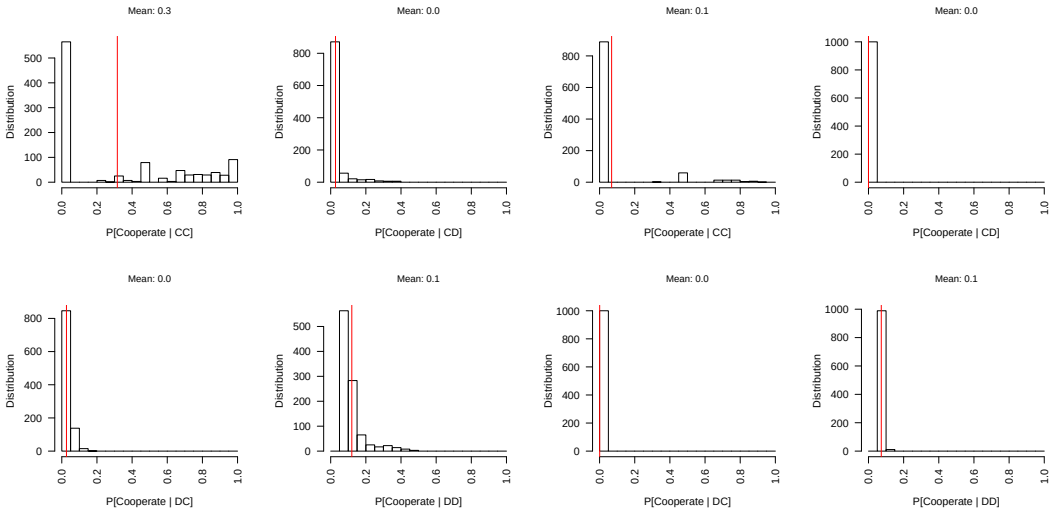
MODEL 1 vs. TFT



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 1 plays against TFT strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

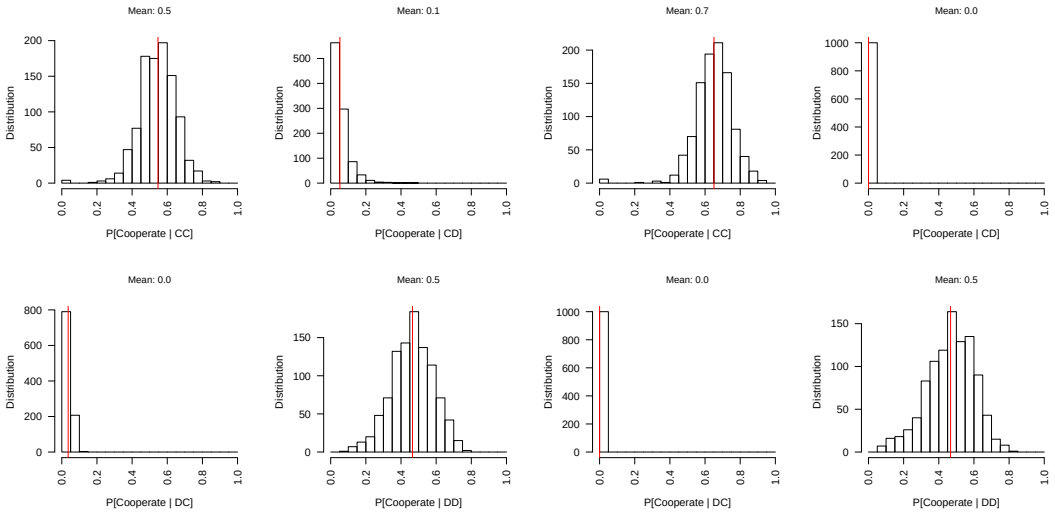
MODEL 1 vs. TFTT



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 1 plays against TFTT strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

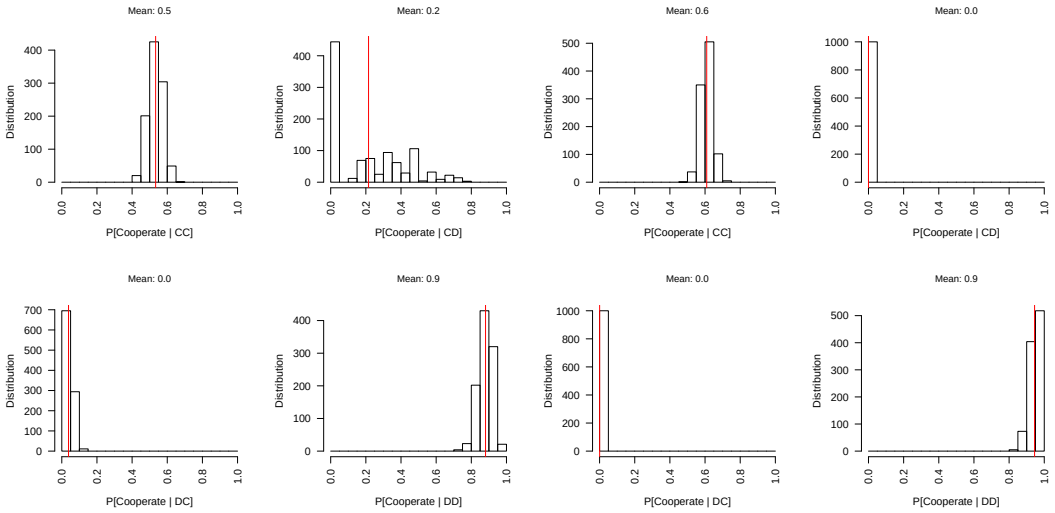
MODEL 1 vs. TFTF



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 1 plays against TFTF strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

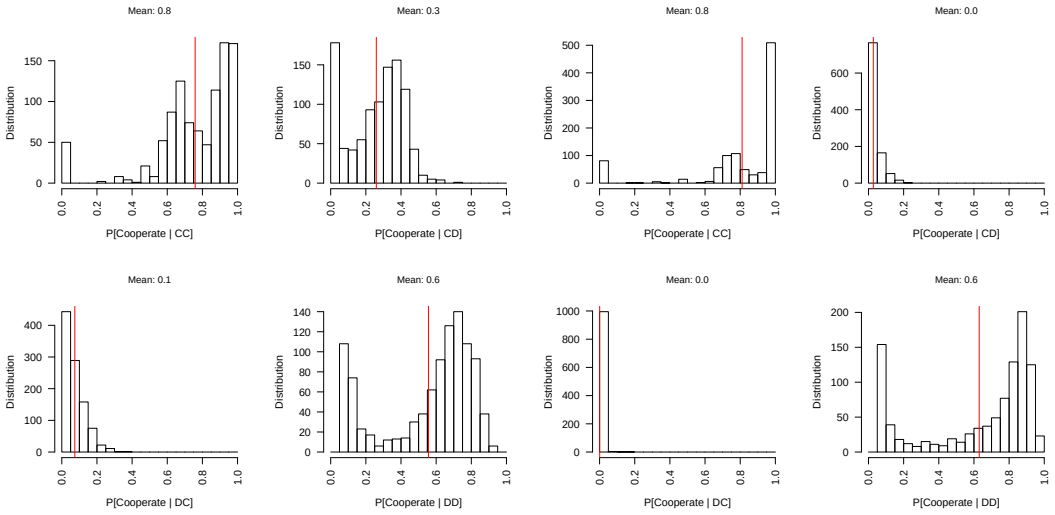
MODEL 1 vs. PAV



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 1 plays against PAV strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

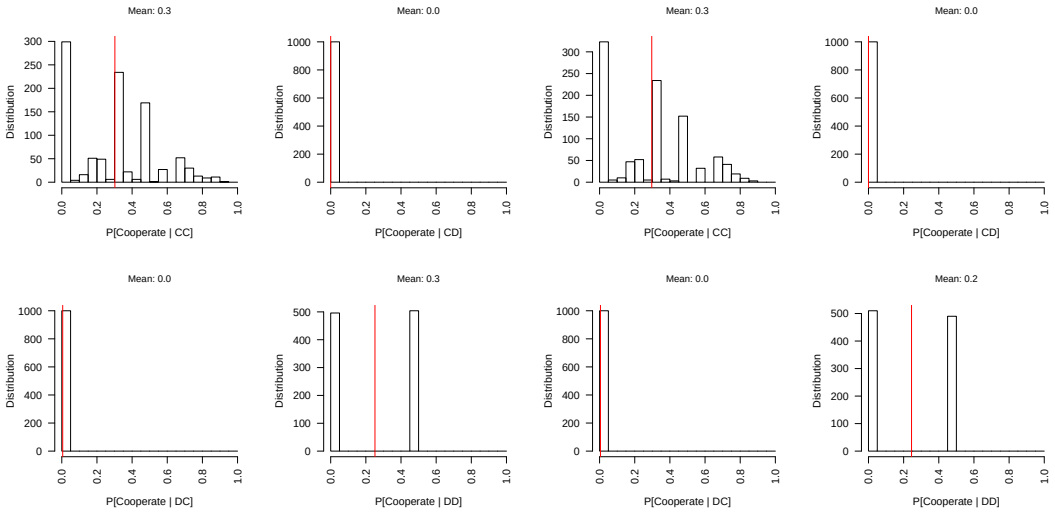
MODEL 1 vs. MODEL 1



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 1 plays against MODEL 1, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

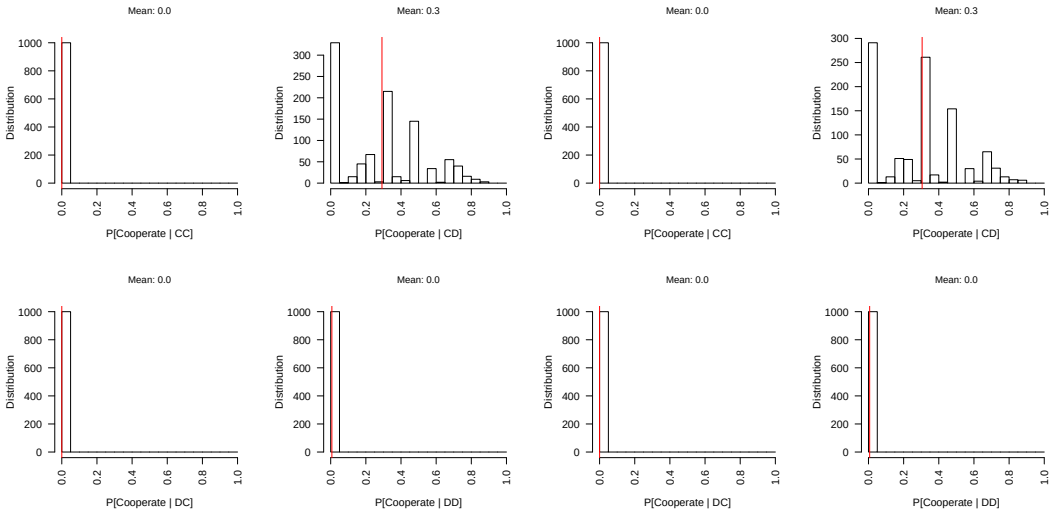
MODEL 2 vs. ALLC



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 2 plays against ALLC strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

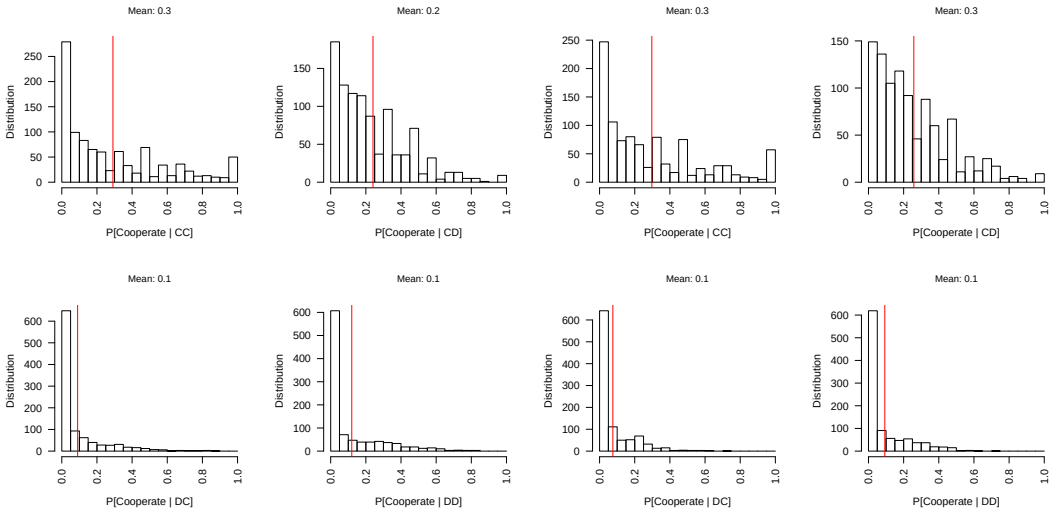
MODEL 2 vs. ALLD



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 2 plays against ALLD strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

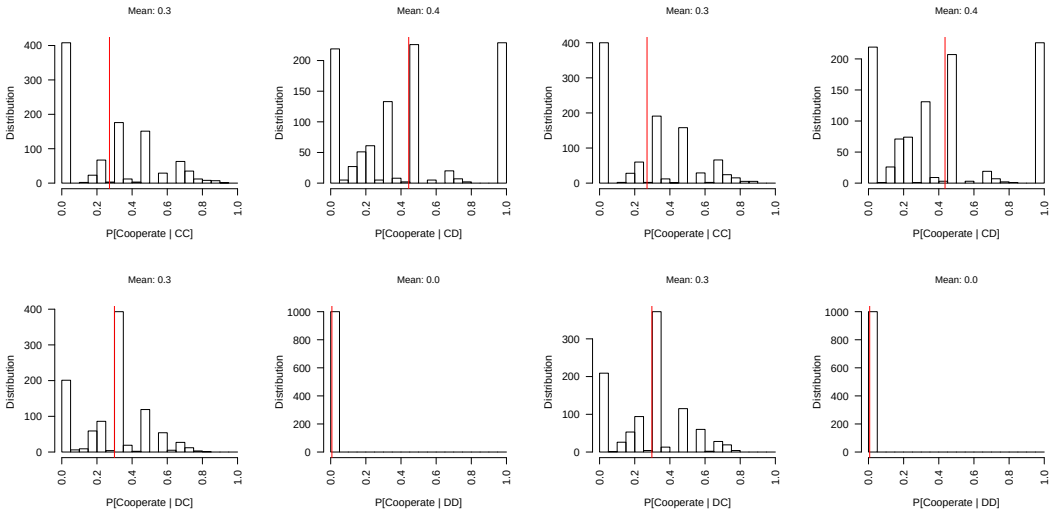
MODEL 2 vs. RAN



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 2 plays against RAN strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

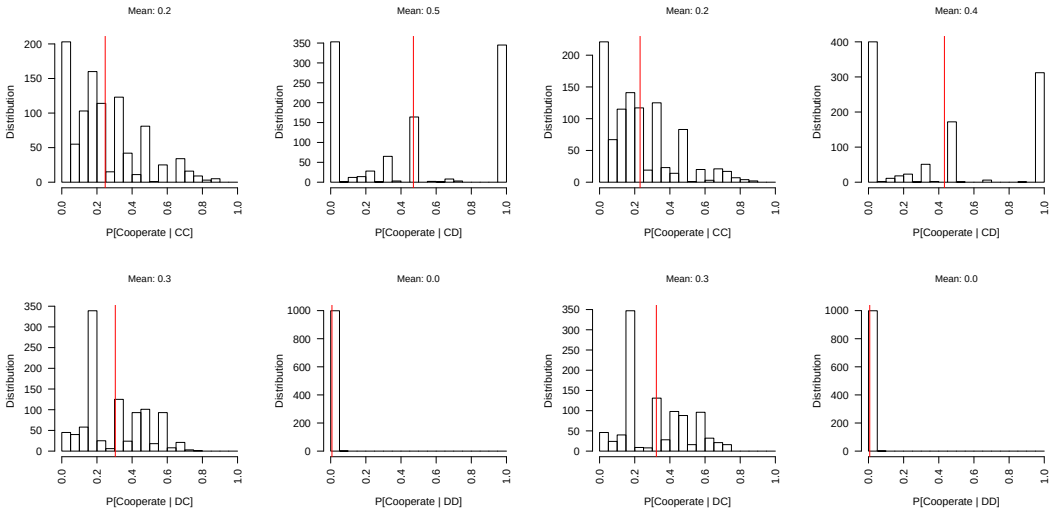
MODEL 2 vs. TFT



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 2 plays against TFT strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

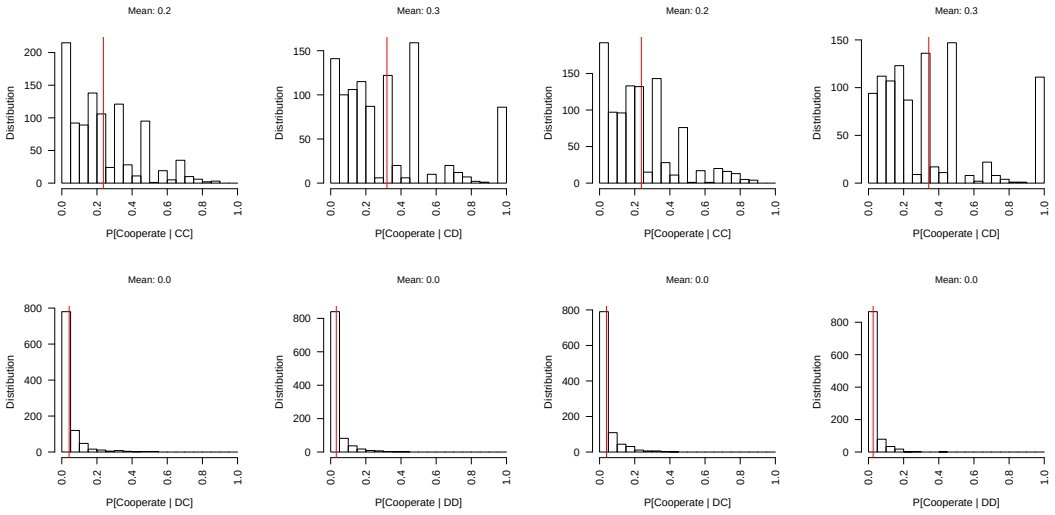
MODEL 2 vs. TFTT



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 2 plays against TFTT strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

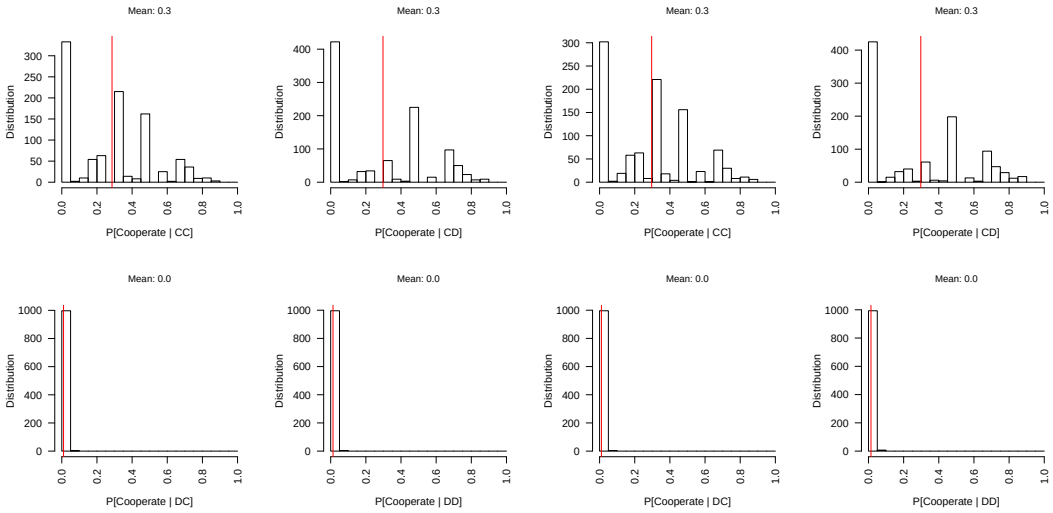
MODEL 2 vs. TFTF



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 2 plays against TFTF strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

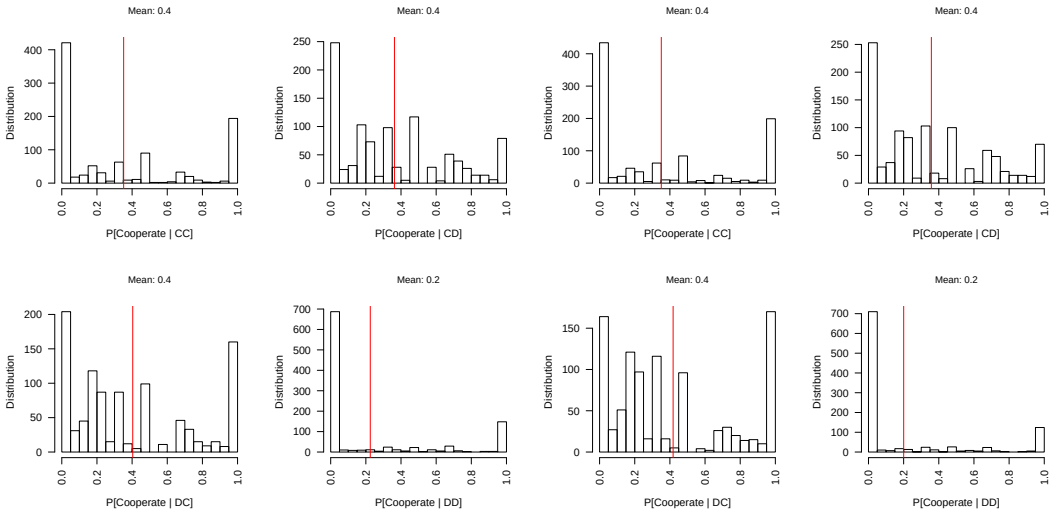
MODEL 2 vs. PAV



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 2 plays against PAV strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

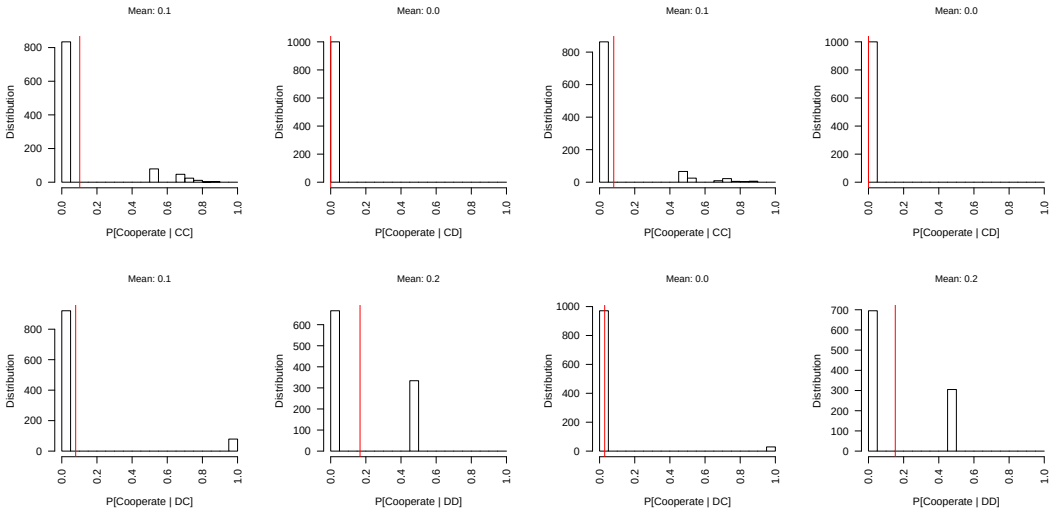
MODEL 2 vs. MODEL 2



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 2 plays against MODEL 2, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

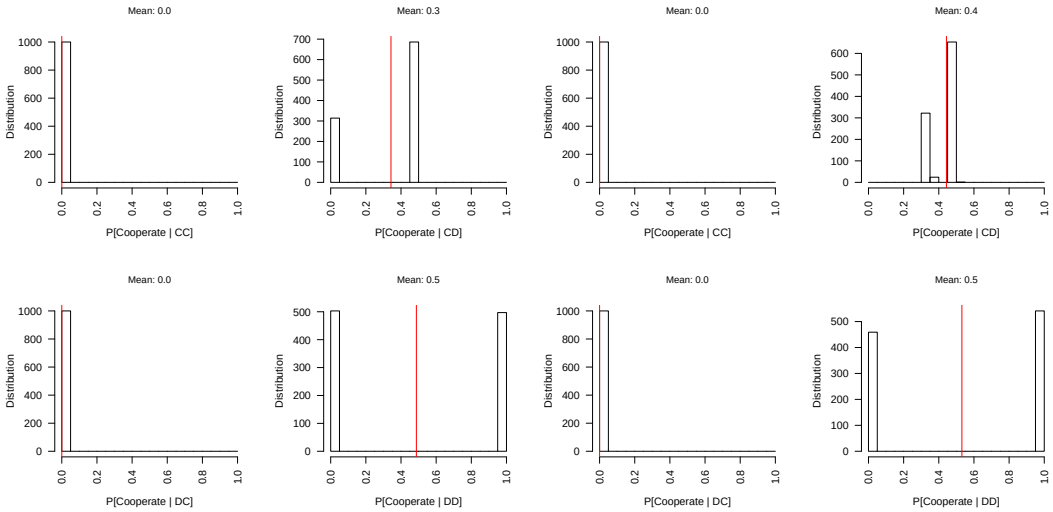
MODEL 3 vs. ALLC



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 3 plays against ALLC strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

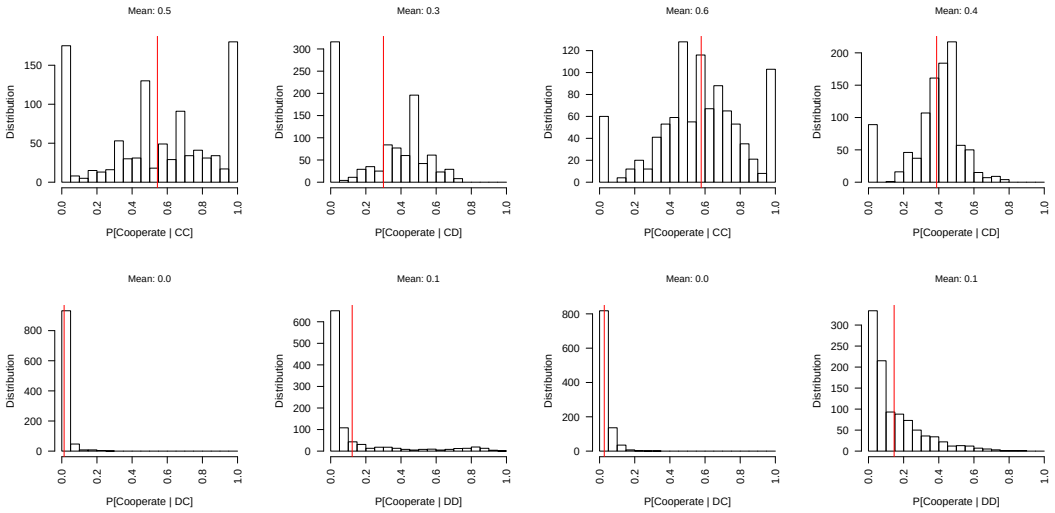
MODEL 3 vs. ALLD



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 3 plays against ALLD strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

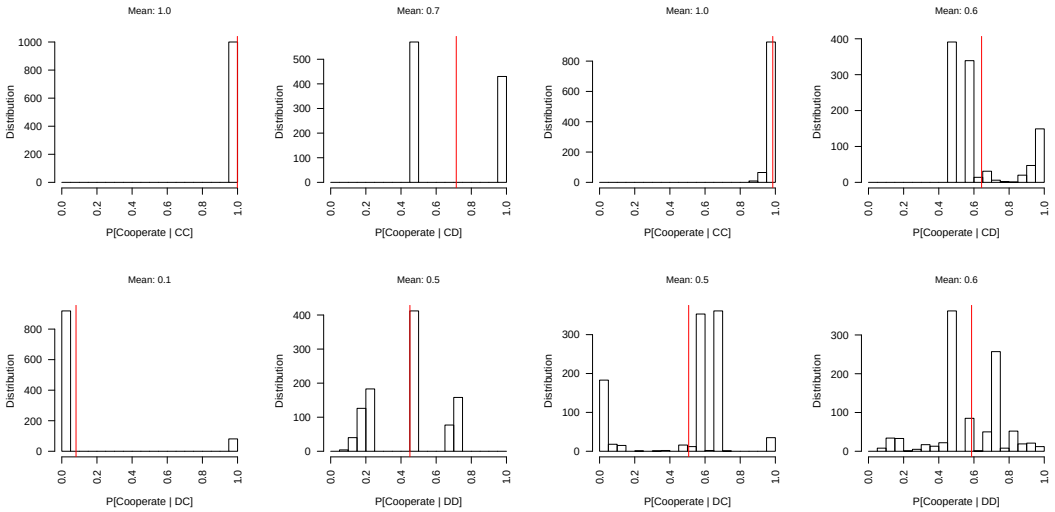
MODEL 3 vs. RAN



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 3 plays against RAN strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

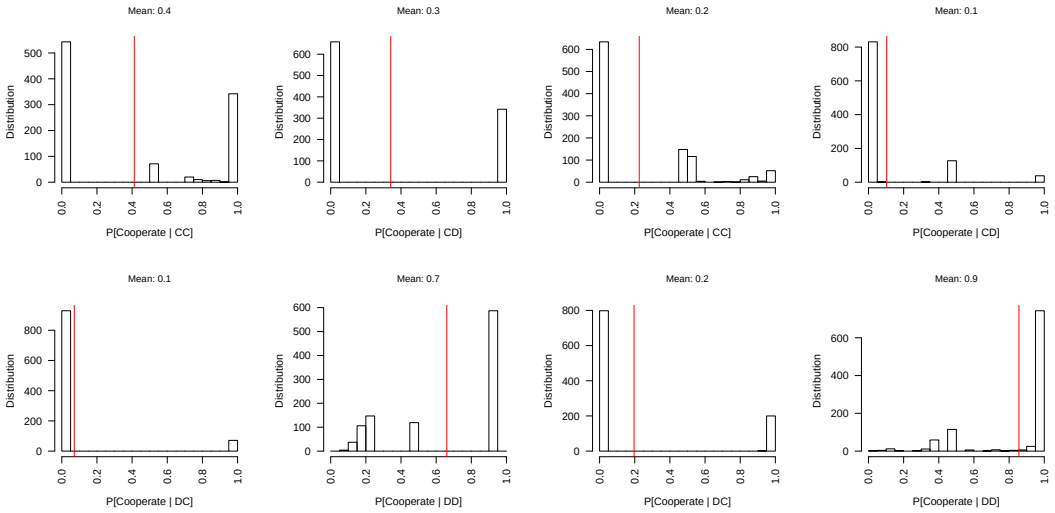
MODEL 3 vs. TFT



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 3 plays against TFT strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5 (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

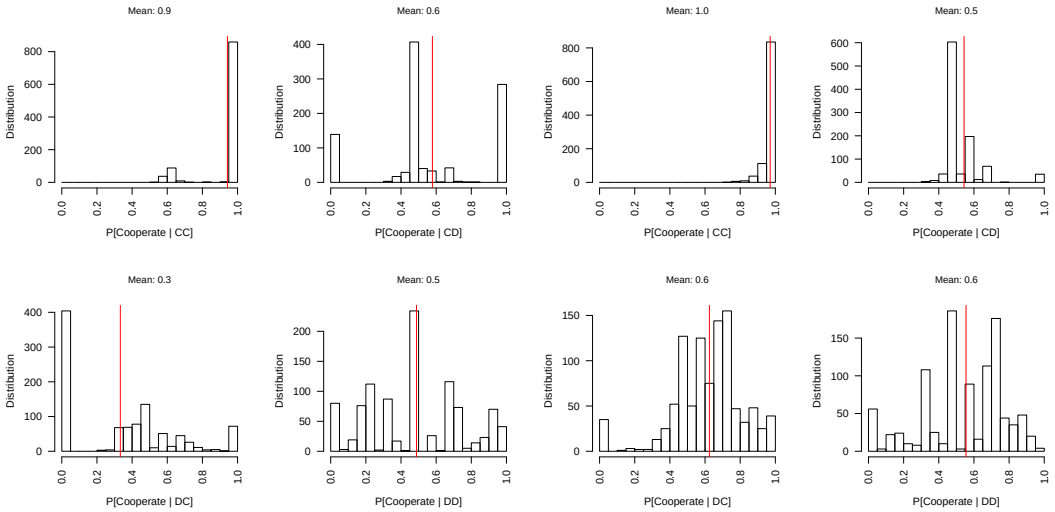
MODEL 3 vs. TFTT



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 3 plays against TFTT strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

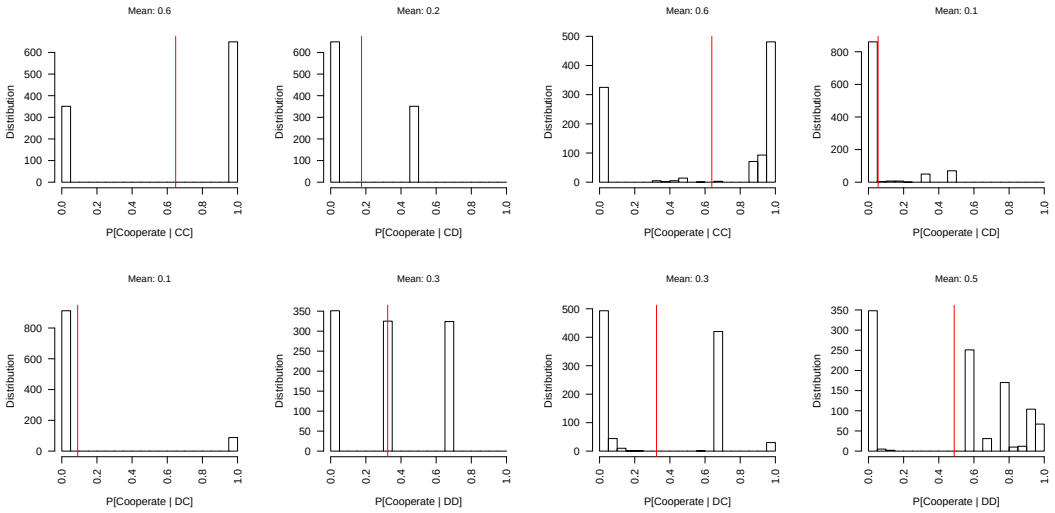
MODEL 3 vs. TFTF



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 3 plays against TFTF strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

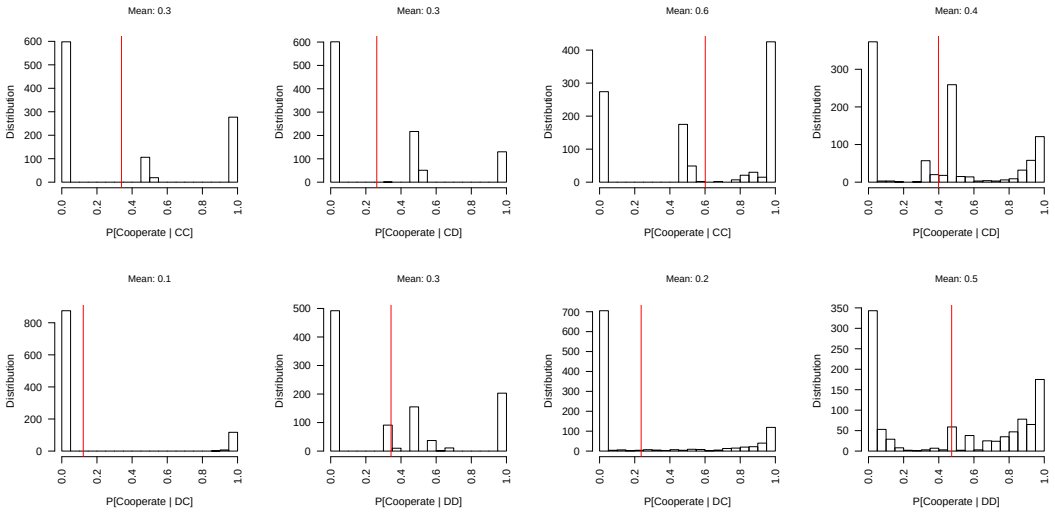
MODEL 3 vs. PAV



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 3 plays against PAV strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

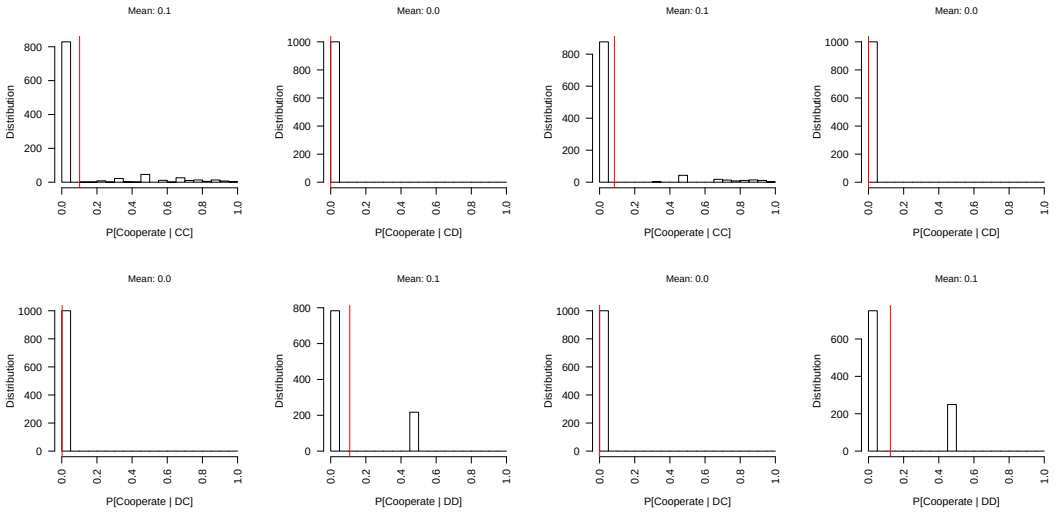
MODEL 3 vs. MODEL 3



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 3 plays against MODEL 1, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

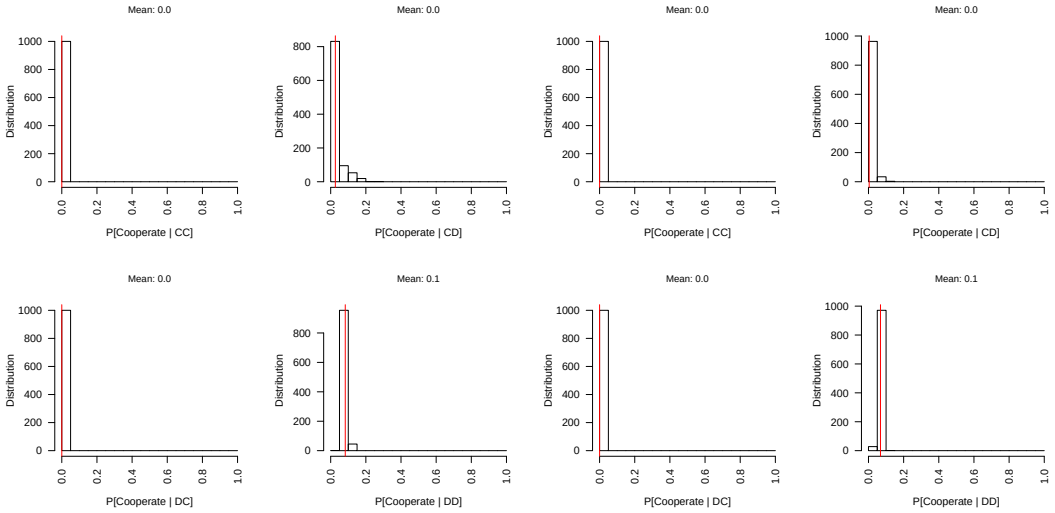
MODEL 4 vs. ALLC



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 4 plays against ALLC strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

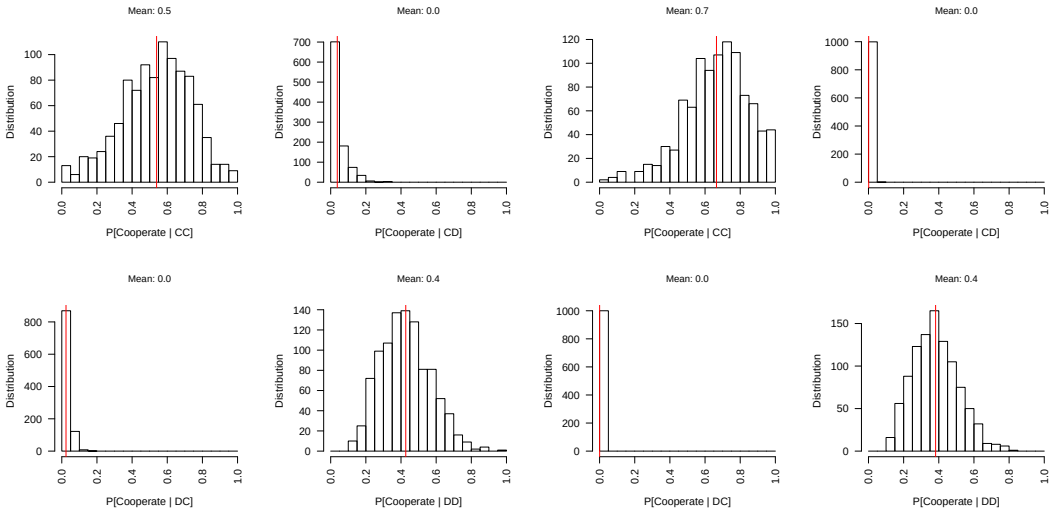
MODEL 4 vs. ALLD



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 4 plays against ALLD strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

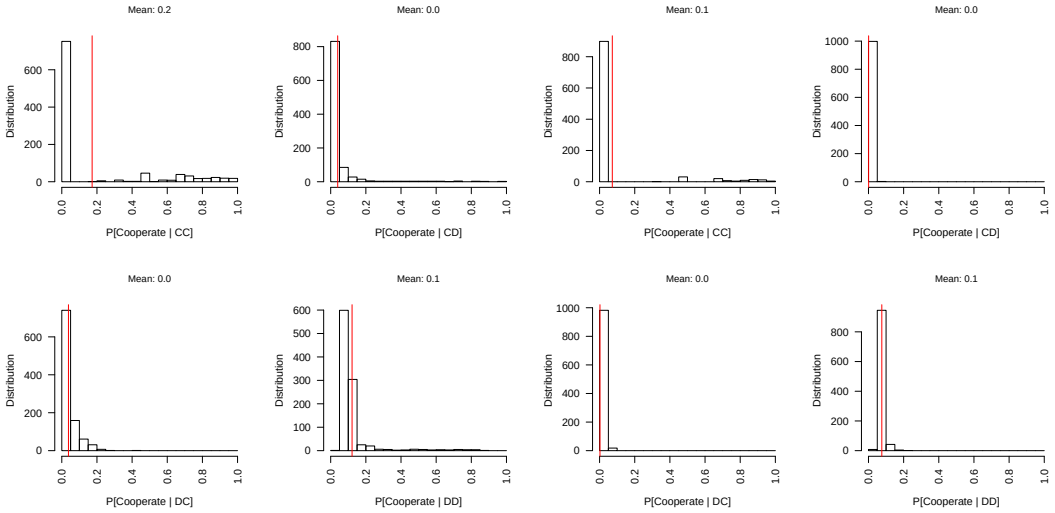
MODEL 4 vs. RAN



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 4 plays against RAN strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

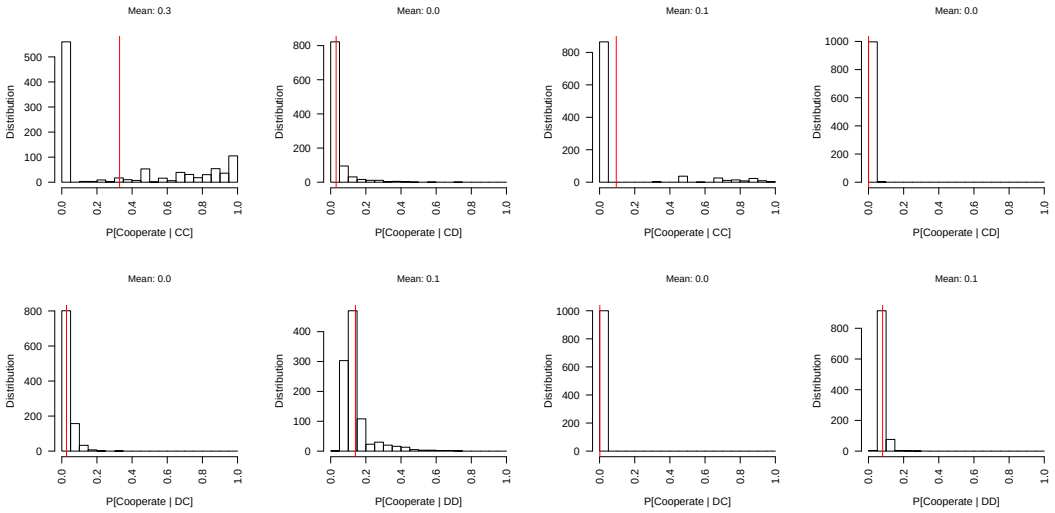
MODEL 4 vs. TFT



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 4 plays against TFT strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

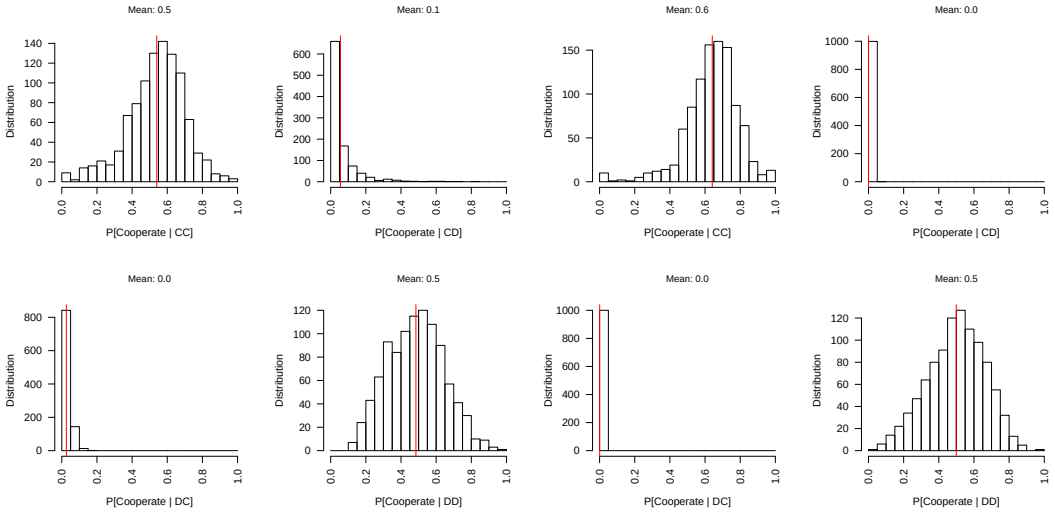
MODEL 4 vs. TFTT



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 4 plays against TFTT strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

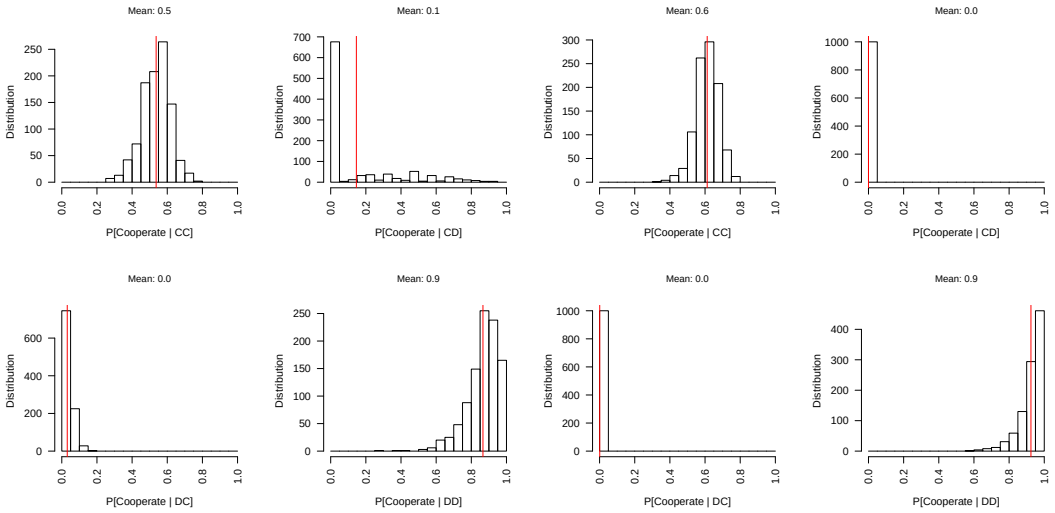
MODEL 4 vs. TFTF



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 4 plays against TFTF strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

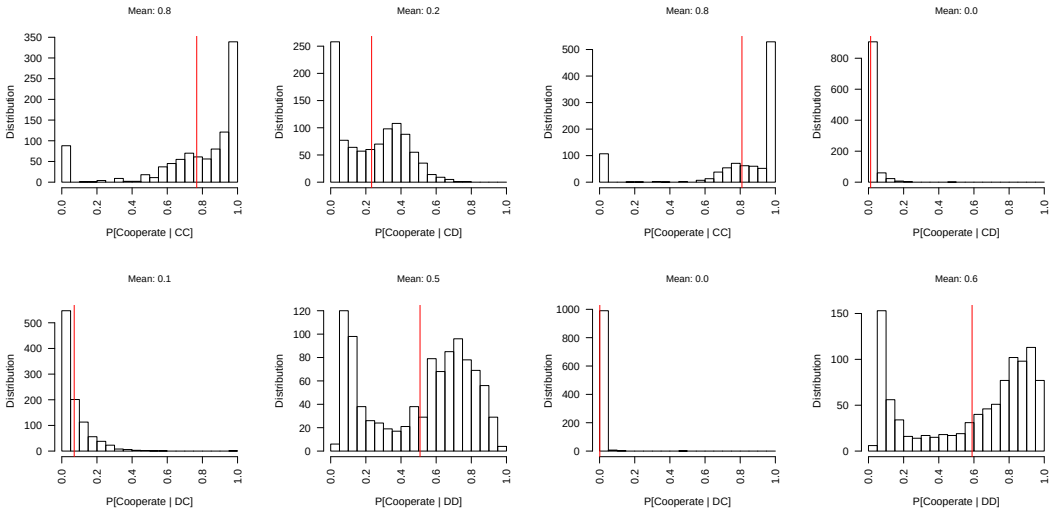
MODEL 4 vs. PAV



(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 4 plays against PAV strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

MODEL 4 vs. MODEL 4

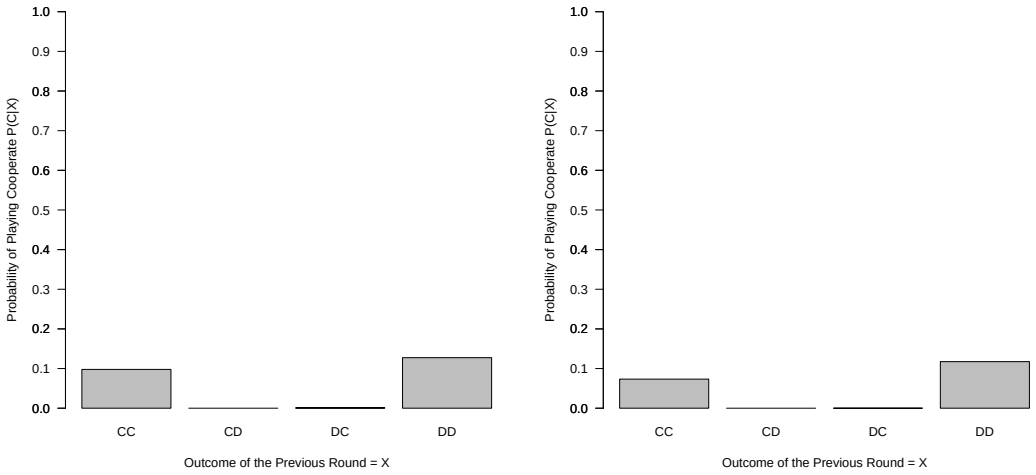


(a) (b)

Distribution of cooperation probabilities given the outcome of previous round when MODEL 4 plays against MODEL 4, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

C Mean of Conditional Cooperation Probabilities

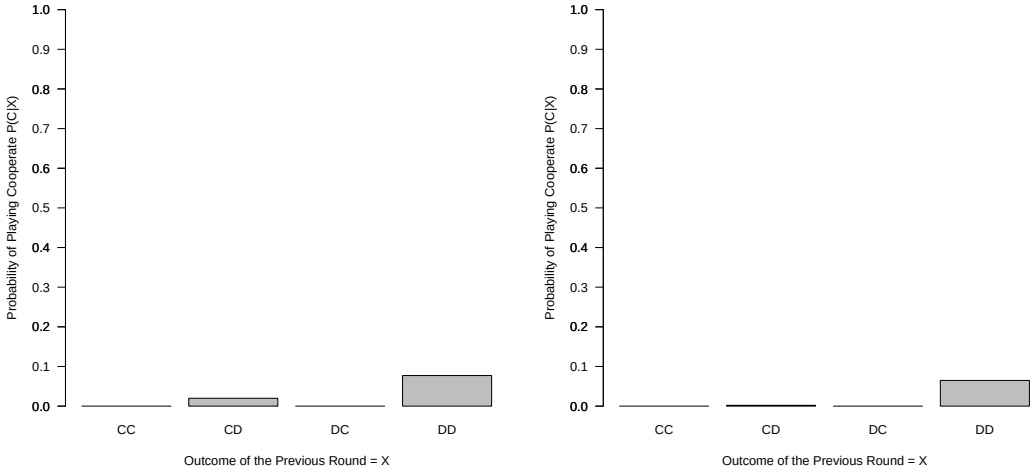
MODEL 1 vs. ALLC



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 1 plays against ALLC strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

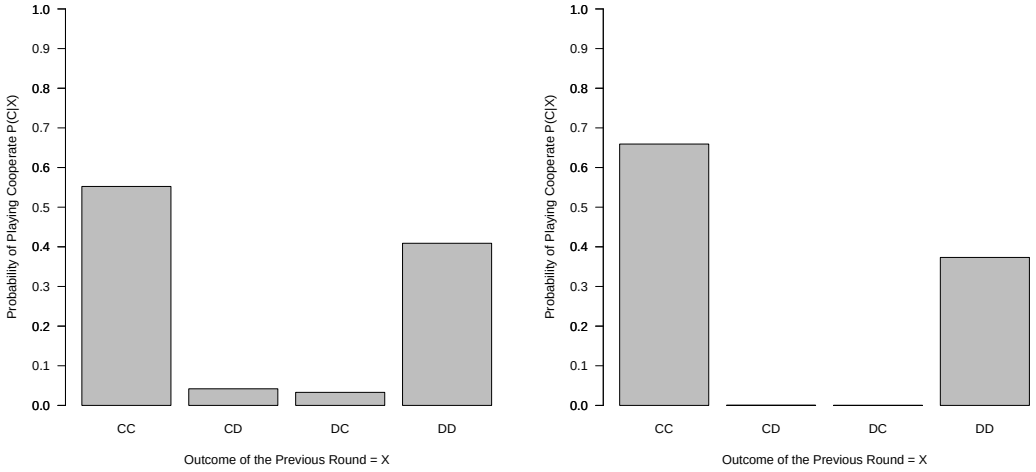
MODEL 1 vs. ALLD



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 1 plays against ALLD strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

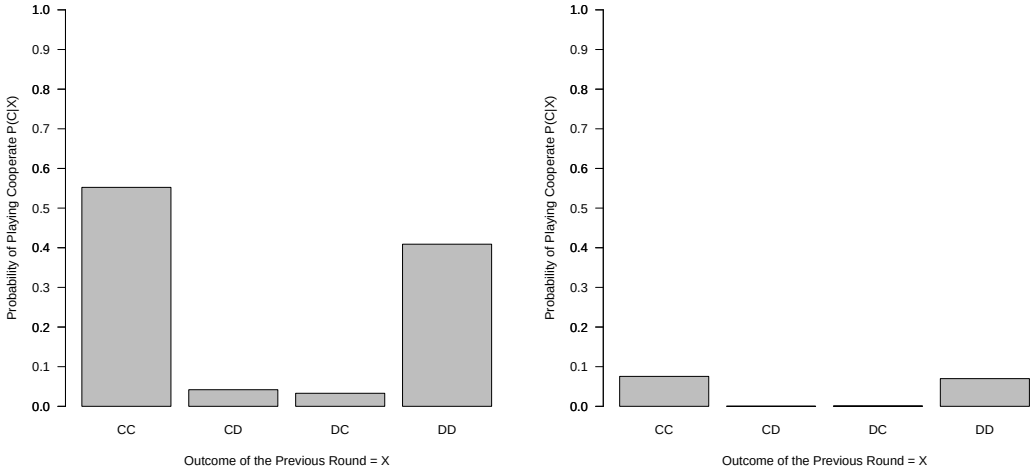
MODEL 1 vs. RAN



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 1 plays against RAN strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

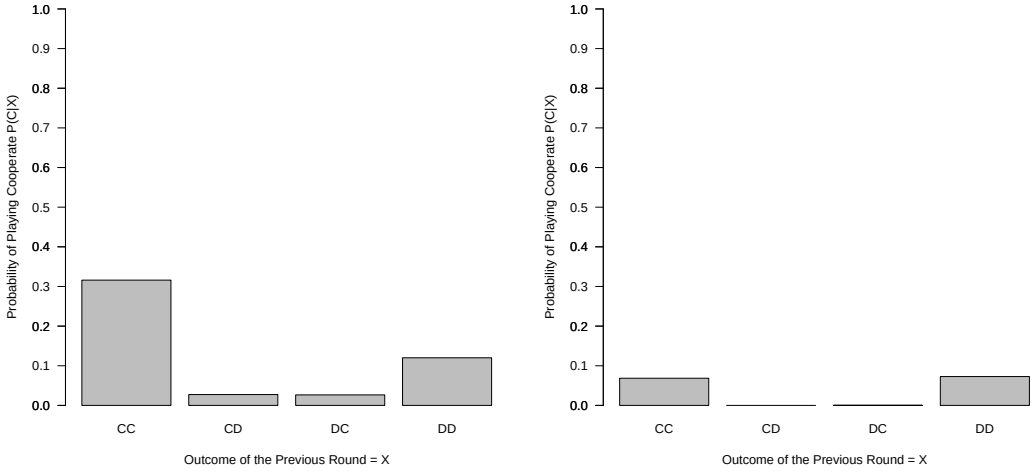
MODEL 1 vs. TFT



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 1 plays against TFT strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

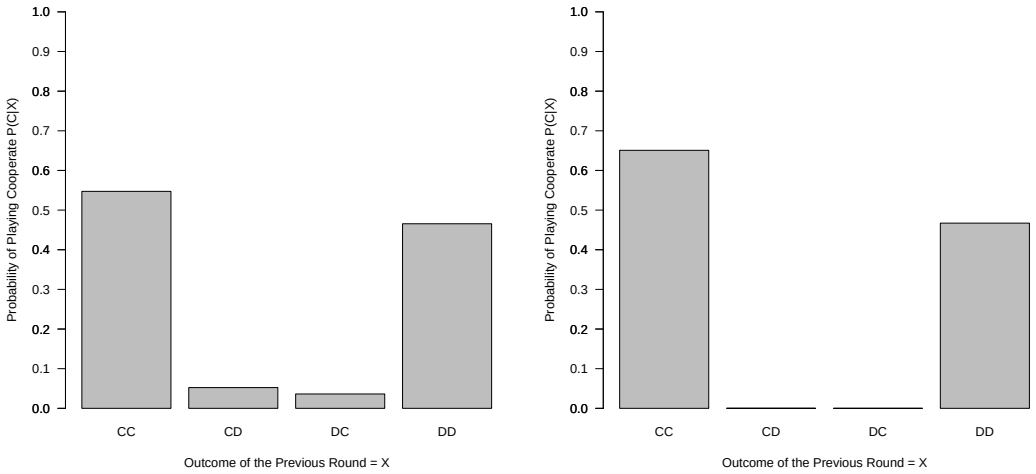
MODEL 1 vs. TFTT



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 1 plays against TFTT strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

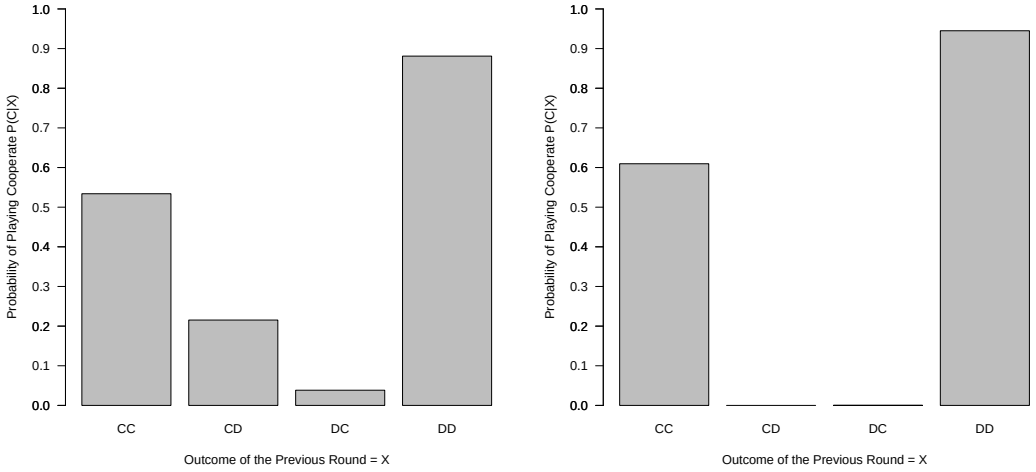
MODEL 1 vs. TFTF



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 1 plays against TFTF strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

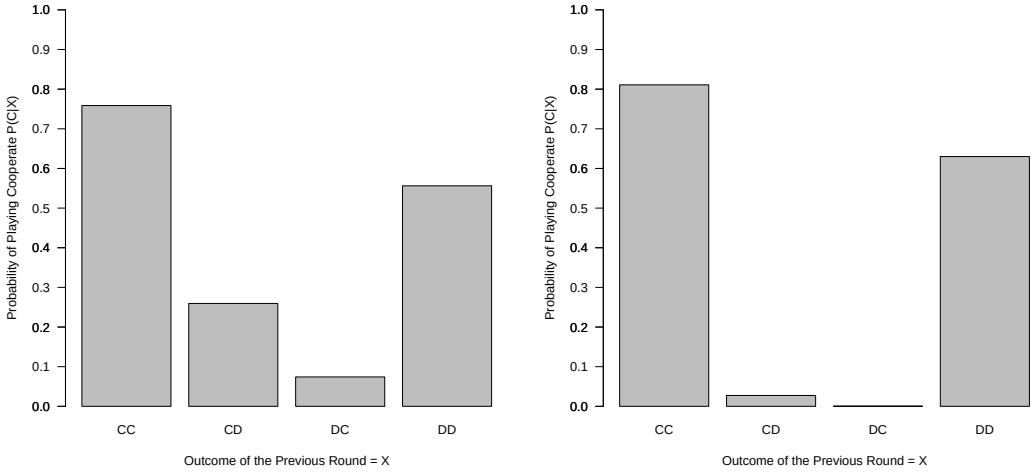
MODEL 1 vs. PAV



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 1 plays against PAV strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

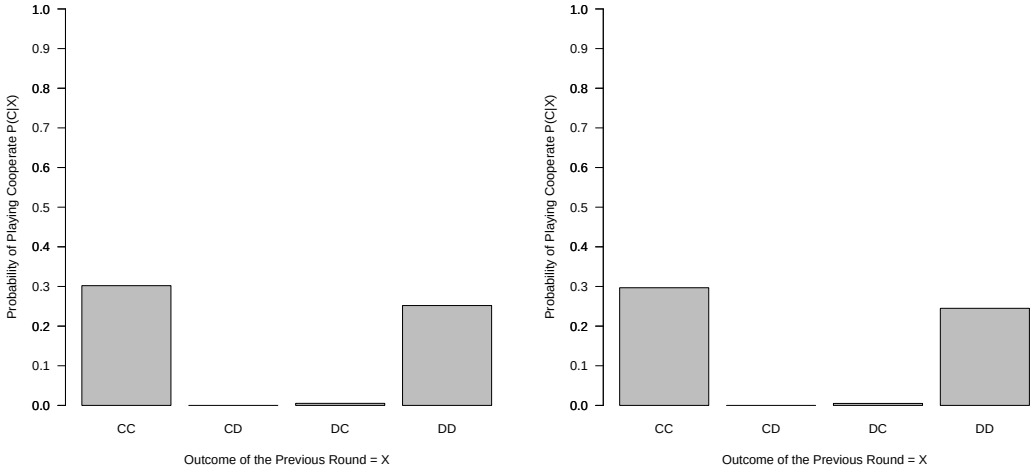
MODEL 1 vs. MODEL 1



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 1 plays against MODEL 1, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

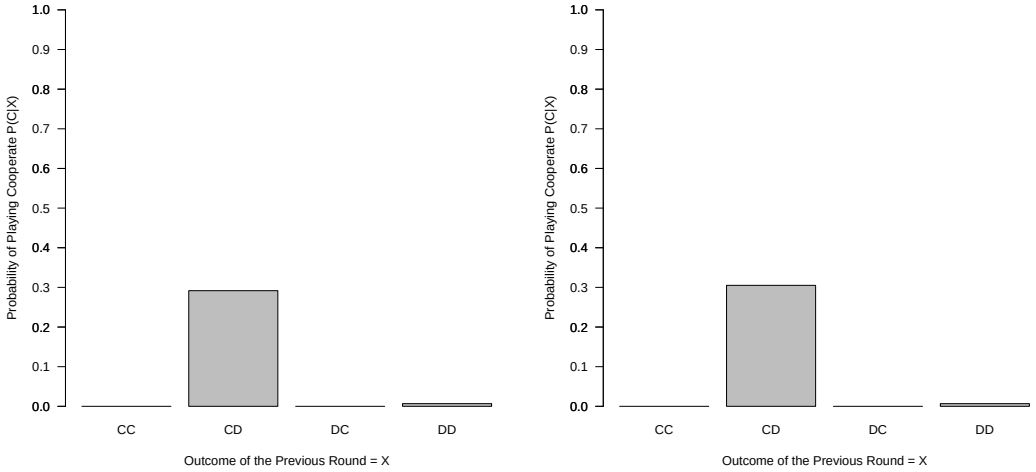
MODEL 2 vs. ALLC



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 2 plays against ALLC strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

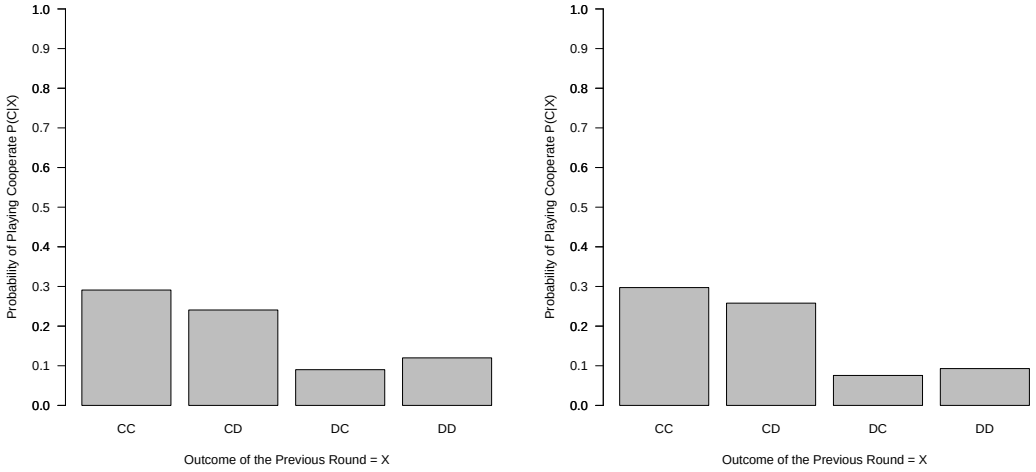
MODEL 2 vs. ALLD



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 2 plays against ALLD strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

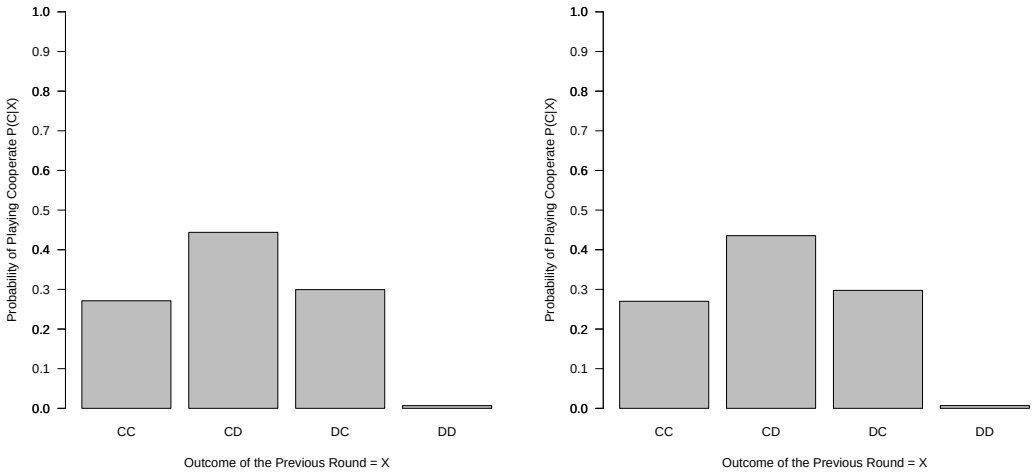
MODEL 2 vs. RAN



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 2 plays against RAN strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

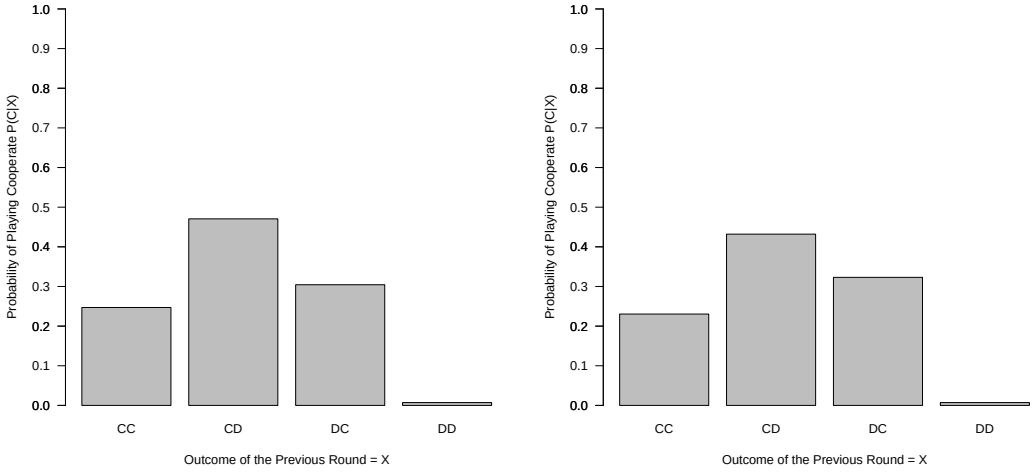
MODEL 2 vs. TFT



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 2 plays against TFT strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

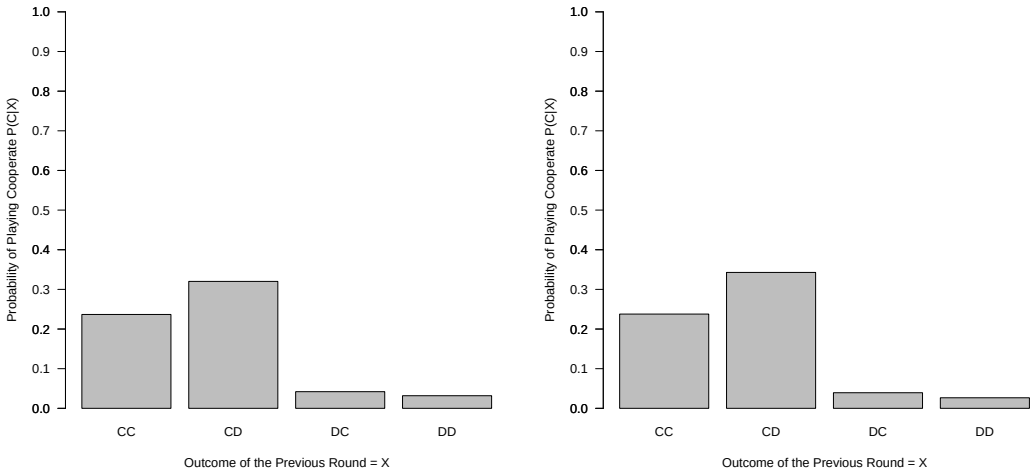
MODEL 2 vs. TFTT



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 2 plays against TFTT strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

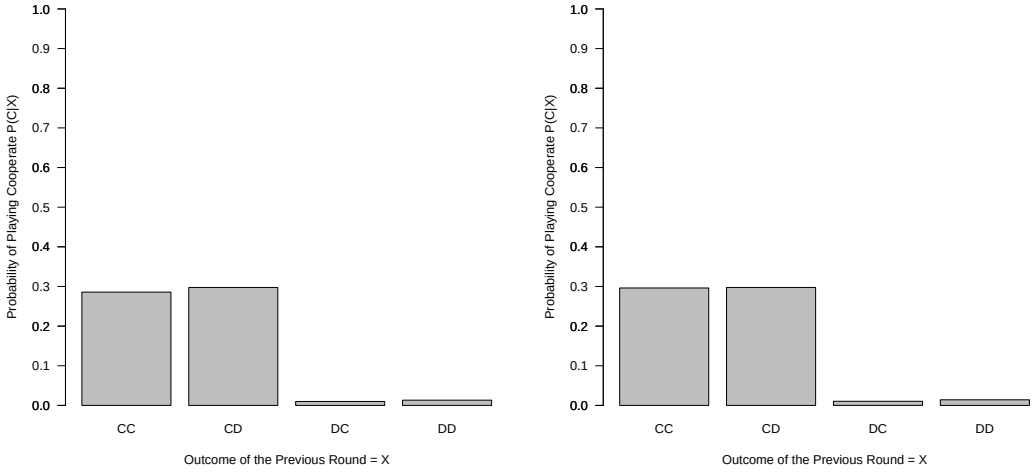
MODEL 2 vs. TFTF



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 2 plays against TFTF strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

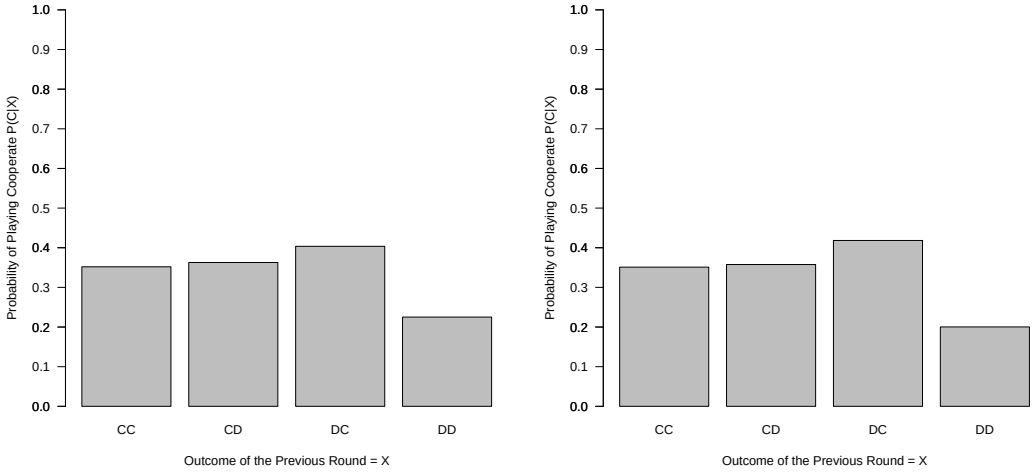
MODEL 2 vs. PAV



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 2 plays against PAV strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

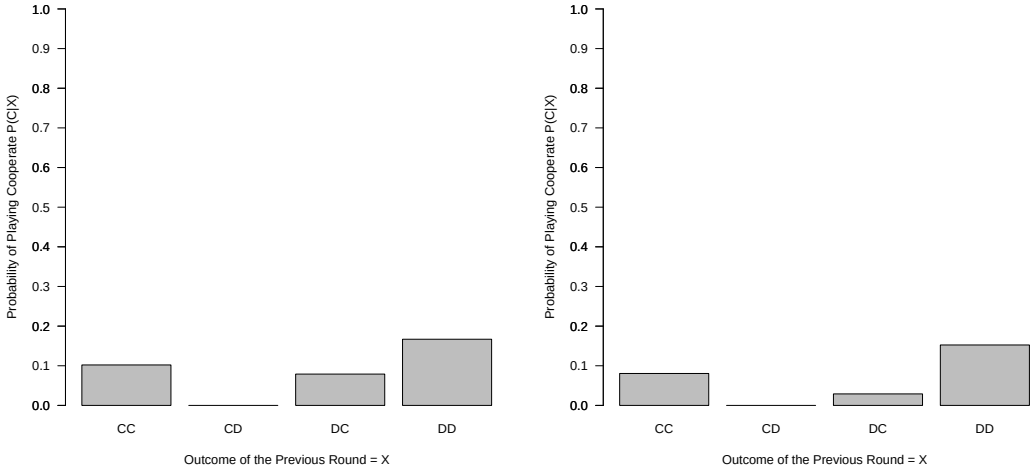
MODEL 2 vs. MODEL 2



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 2 plays against MODEL 2, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

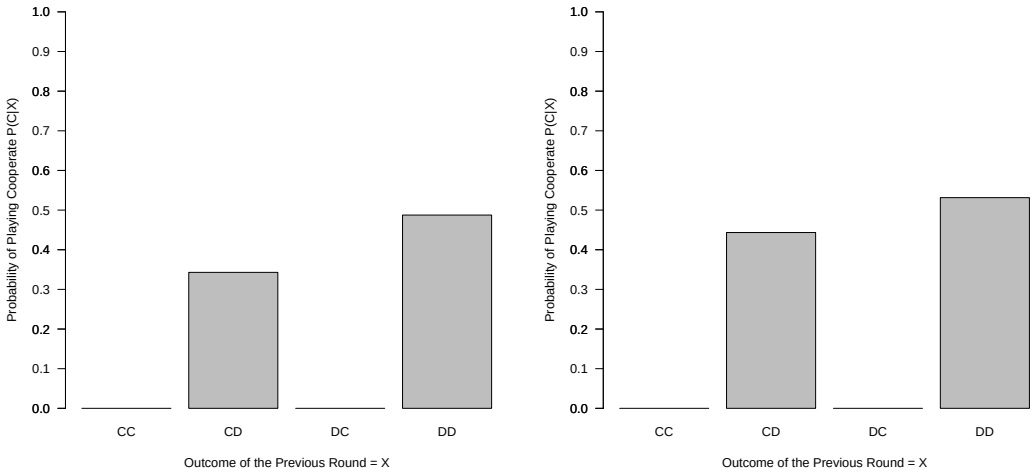
MODEL 3 vs. ALLC



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 3 plays against ALLC strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

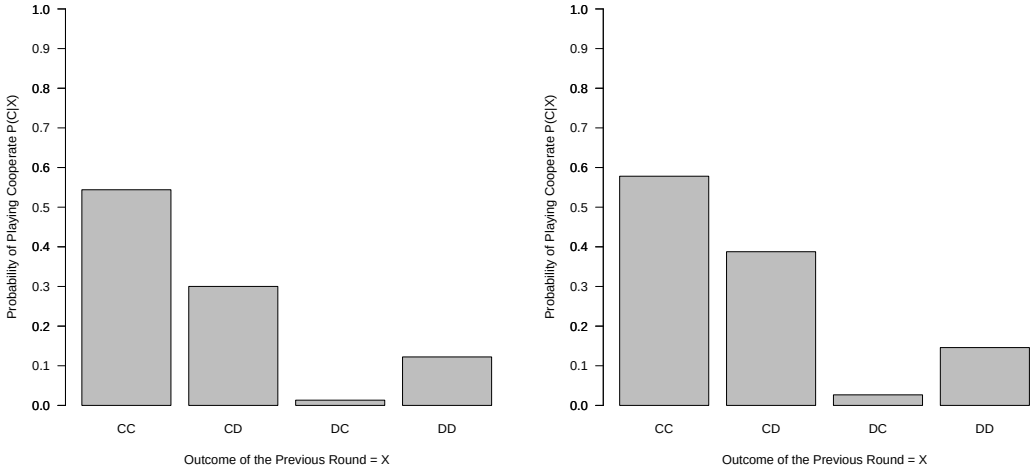
MODEL 3 vs. ALLD



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 3 plays against ALLD strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

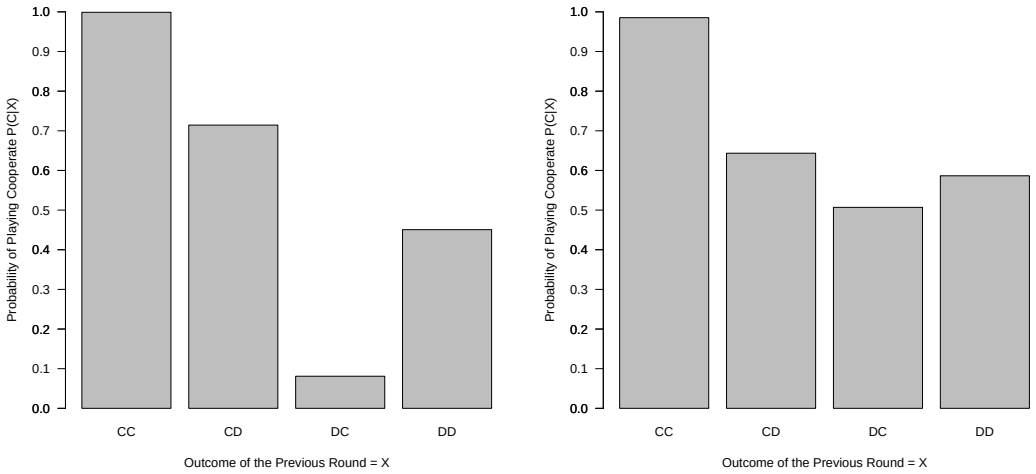
MODEL 3 vs. RAN



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 3 plays against RAN strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

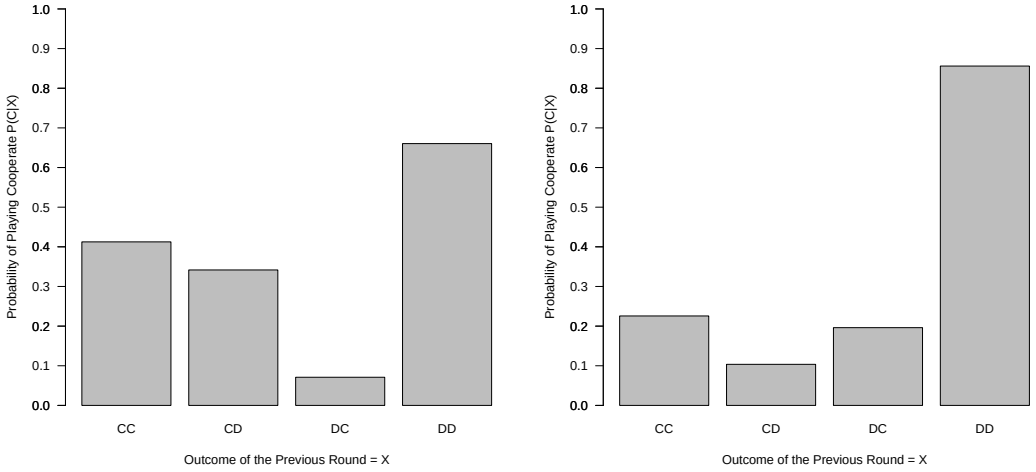
MODEL 3 vs. TFT



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 3 plays against TFT strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5 (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

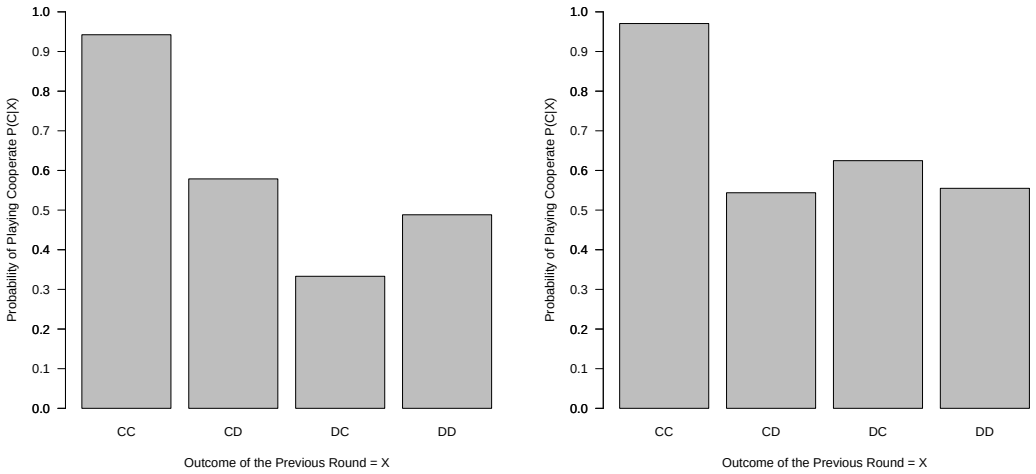
MODEL 3 vs. TFTT



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 3 plays against TFTT strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

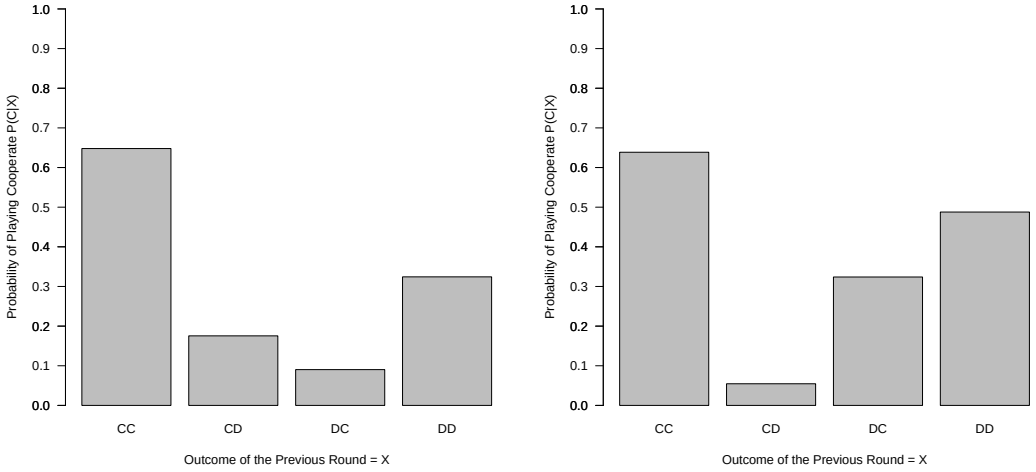
MODEL 3 vs. TFTF



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 3 plays against TFTF strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

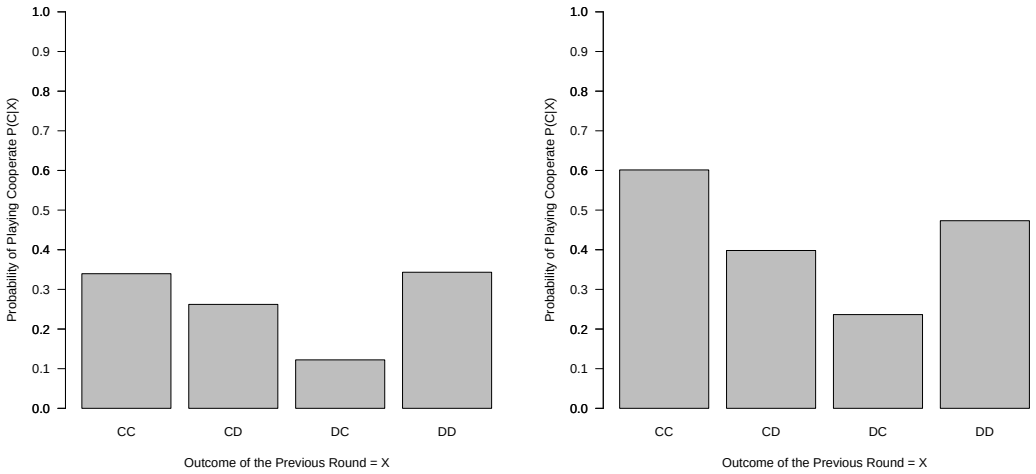
MODEL 3 vs. PAV



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 3 plays against PAV strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

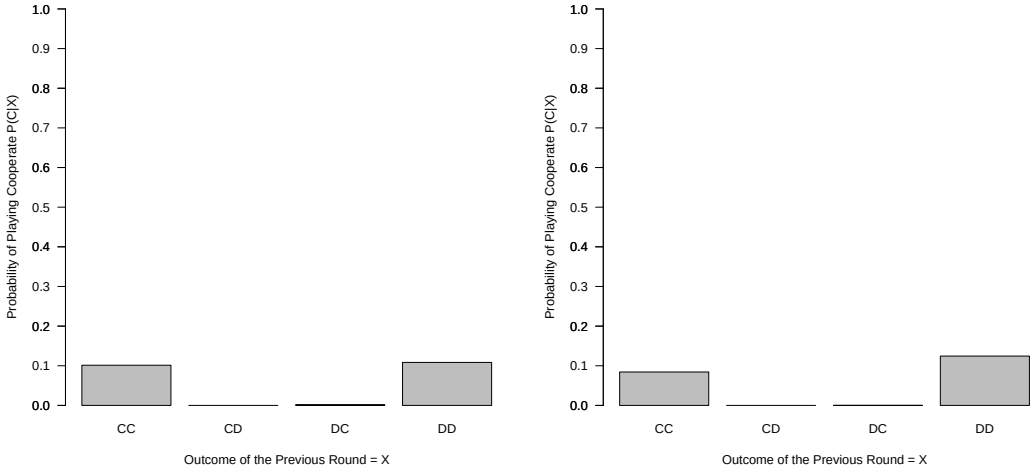
MODEL 3 vs. MODEL 3



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 3 plays against MODEL 1, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

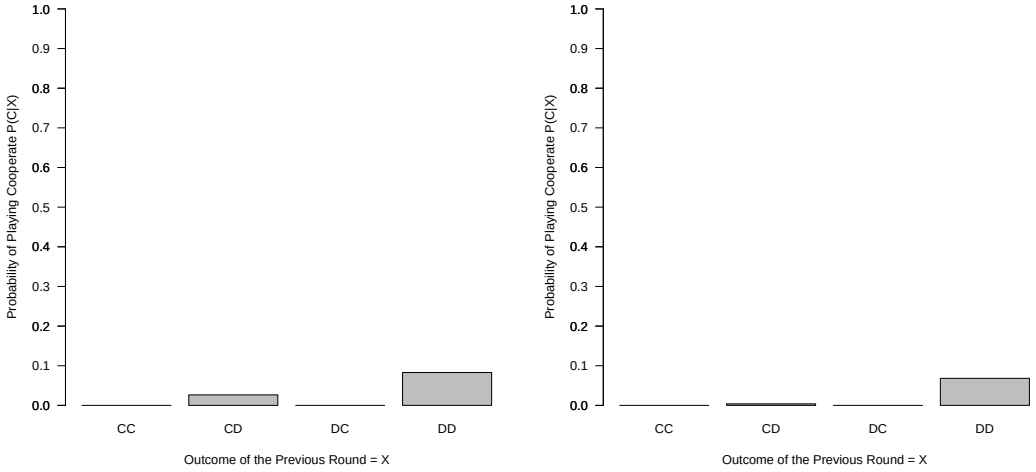
MODEL 4 vs. ALLC



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 4 plays against ALLC strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

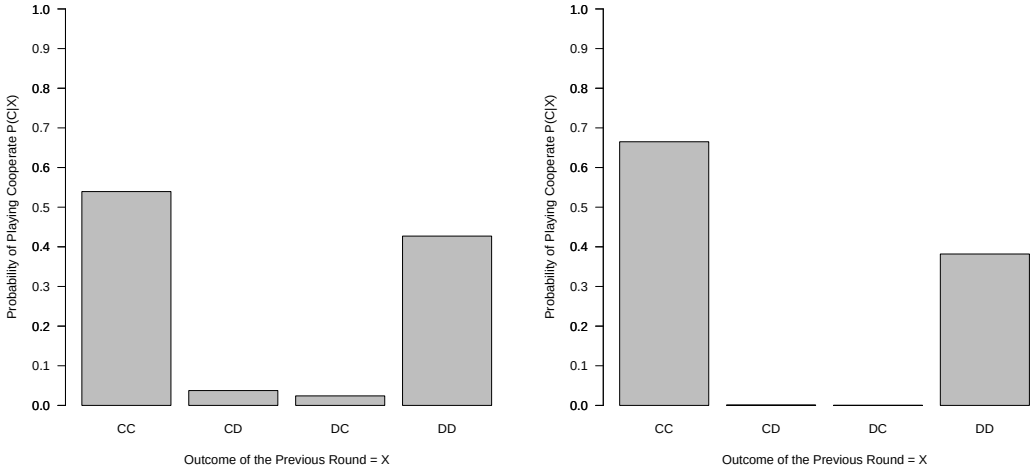
MODEL 4 vs. ALLD



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 4 plays against ALLD strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

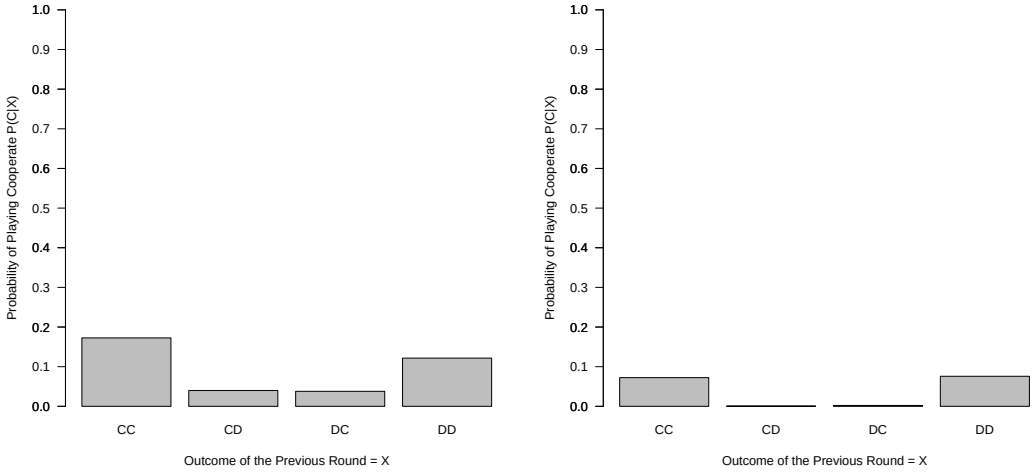
MODEL 4 vs. RAN



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 4 plays against RAN strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

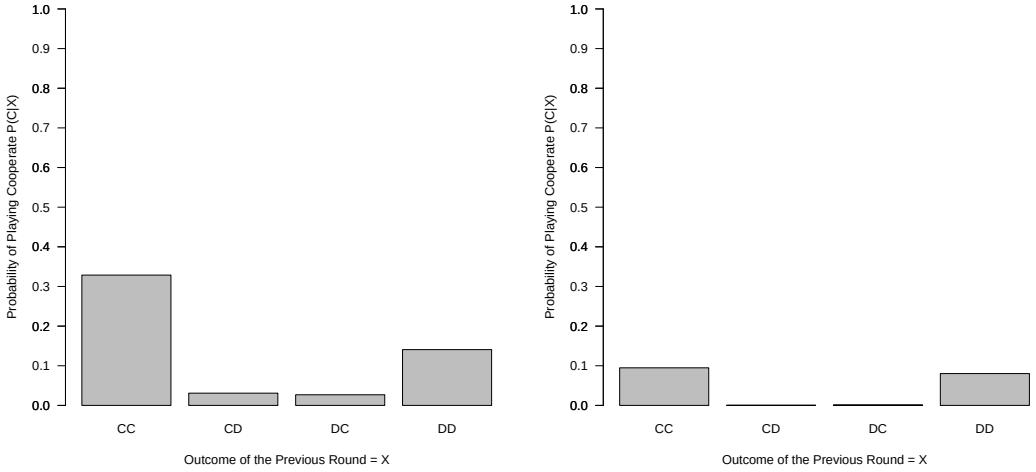
MODEL 4 vs. TFT



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 4 plays against TFT strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

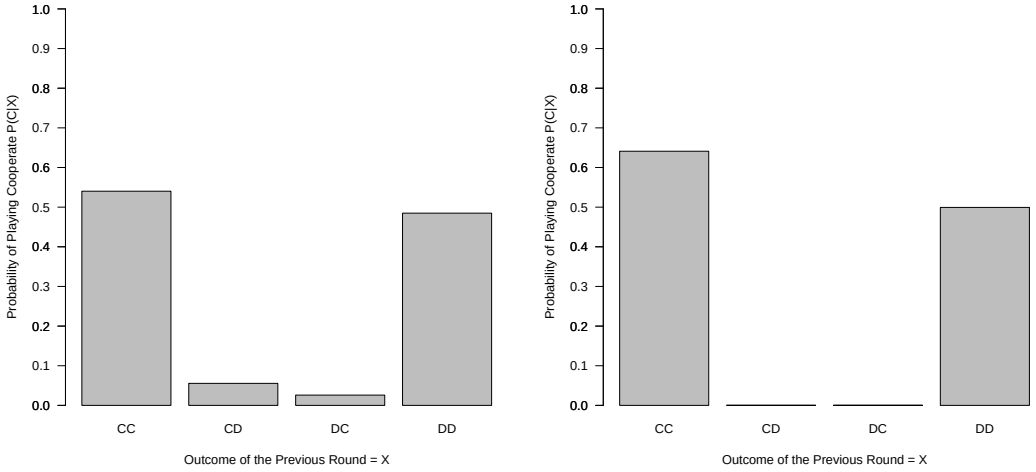
MODEL 4 vs. TFTT



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 4 plays against TFTT strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

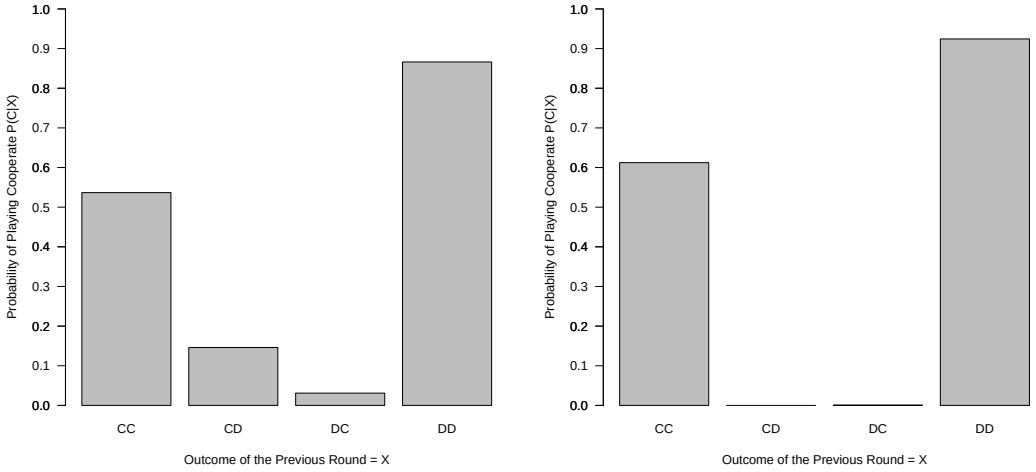
MODEL 4 vs. TFTF



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 4 plays against TFTF strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

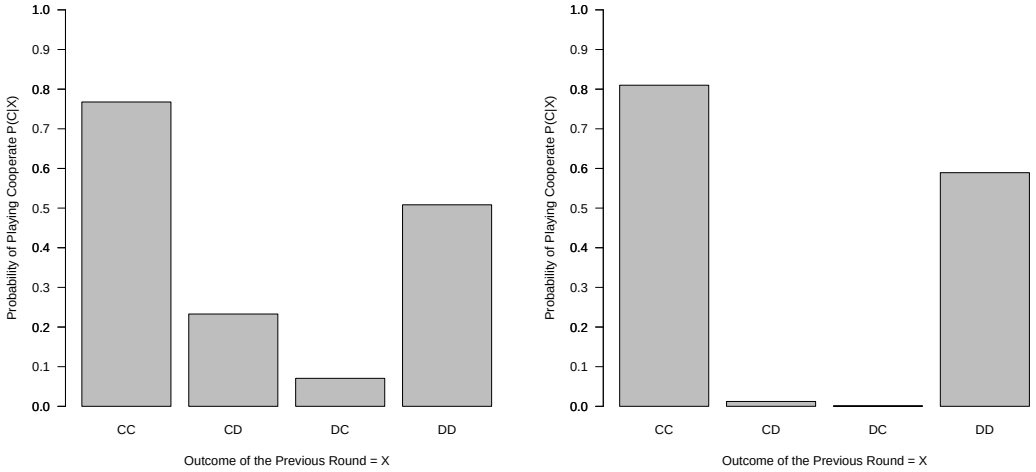
MODEL 4 vs. PAV



(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 4 plays against PAV strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

MODEL 4 vs. MODEL 4

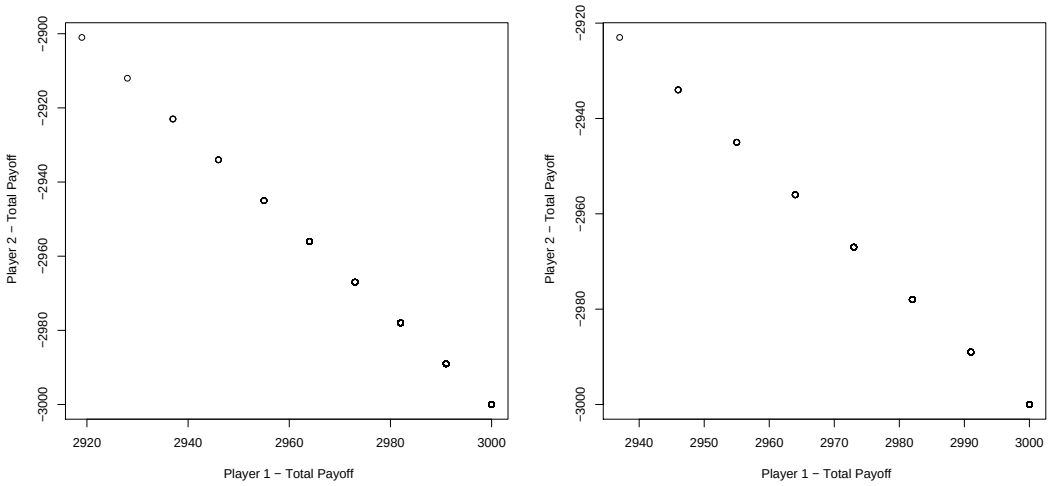


(a) (b)

Mean of cooperation probabilities given the outcome of previous round when MODEL 4 plays against MODEL 4, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

D Distribution of Player Scores

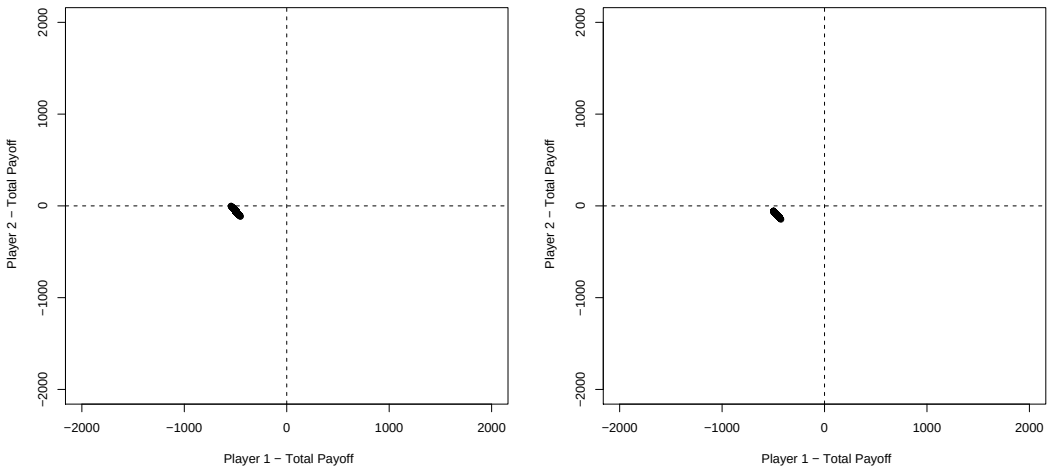
MODEL 1 vs. ALLC



(a) (b)

Distribution of player scores when MODEL 1 plays against ALLC strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

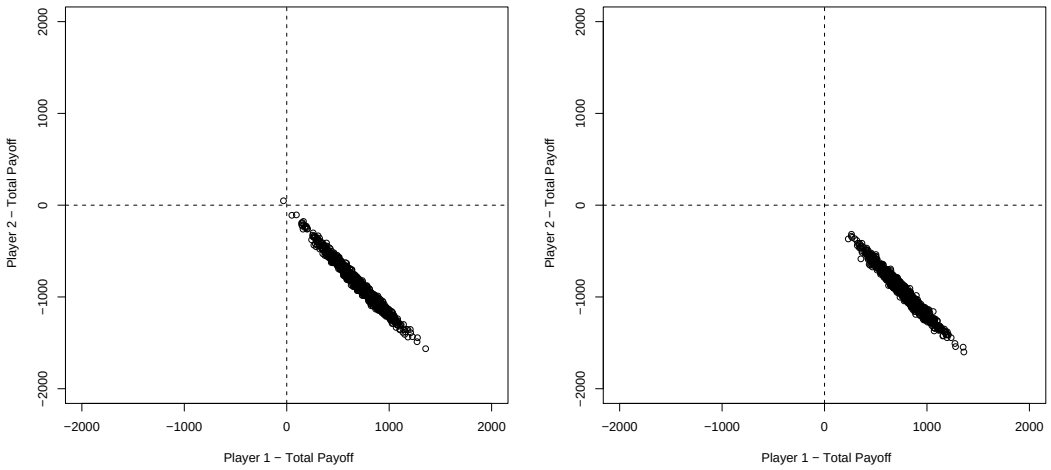
MODEL 1 vs. ALLD



(a) (b)

Distribution of player scores when MODEL 1 plays against ALLD strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

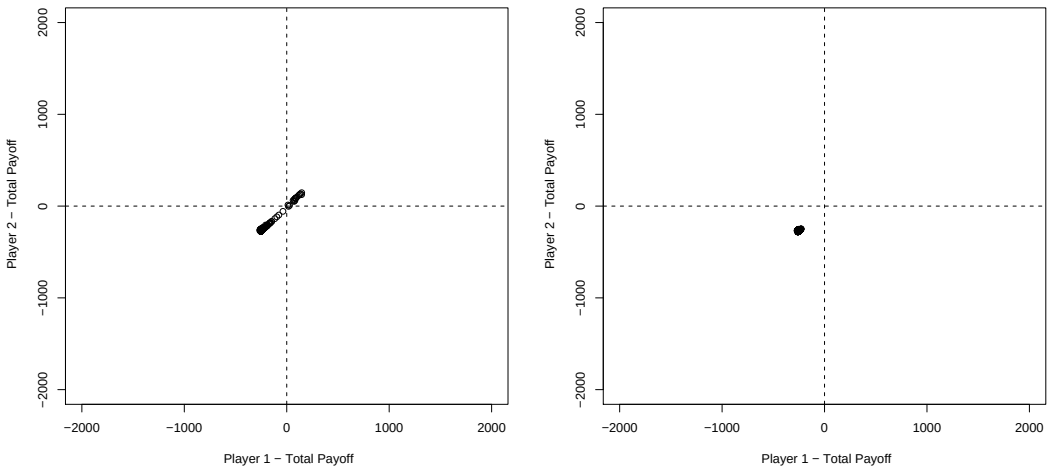
MODEL 1 vs. RAN



(a) (b)

Distribution of player scores when MODEL 1 plays against RAN strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

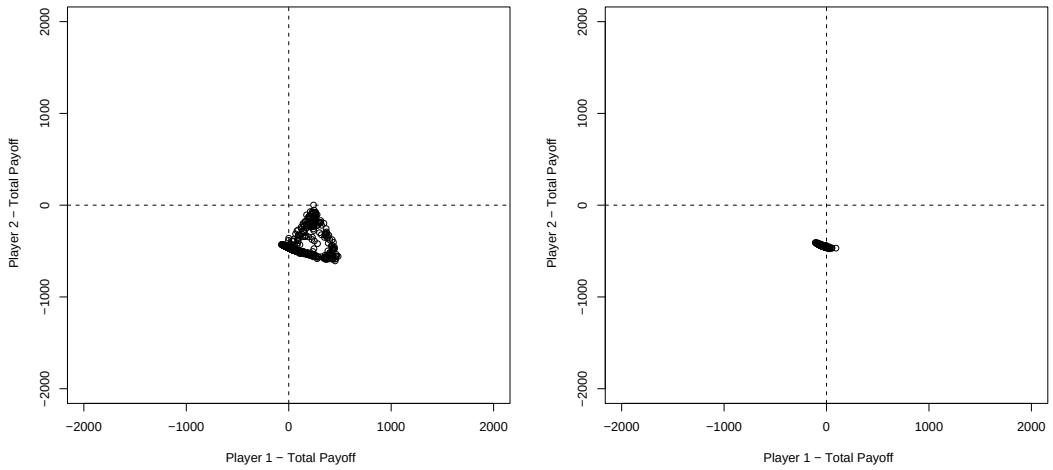
MODEL 1 vs. TFT



(a) (b)

Distribution of player scores when MODEL 1 plays against TFT strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

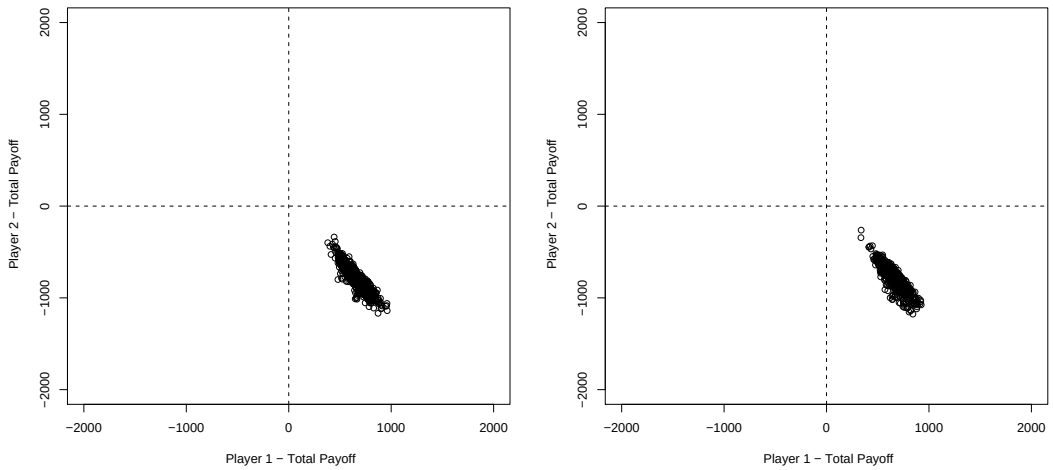
MODEL 1 vs. TFTT



(a) (b)

Distribution of player scores when MODEL 1 plays against TFTT strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

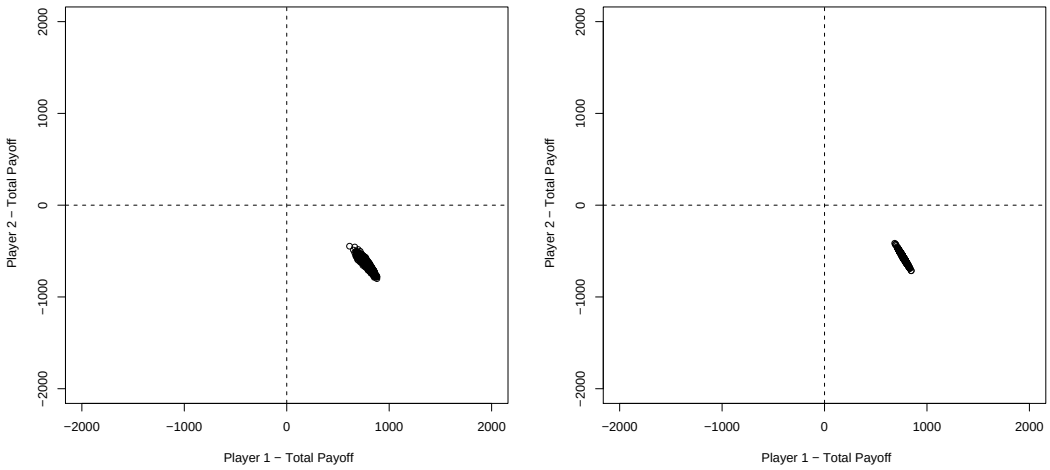
MODEL 1 vs. TFTF



(a) (b)

Distribution of player scores when MODEL 1 plays against TFTF strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

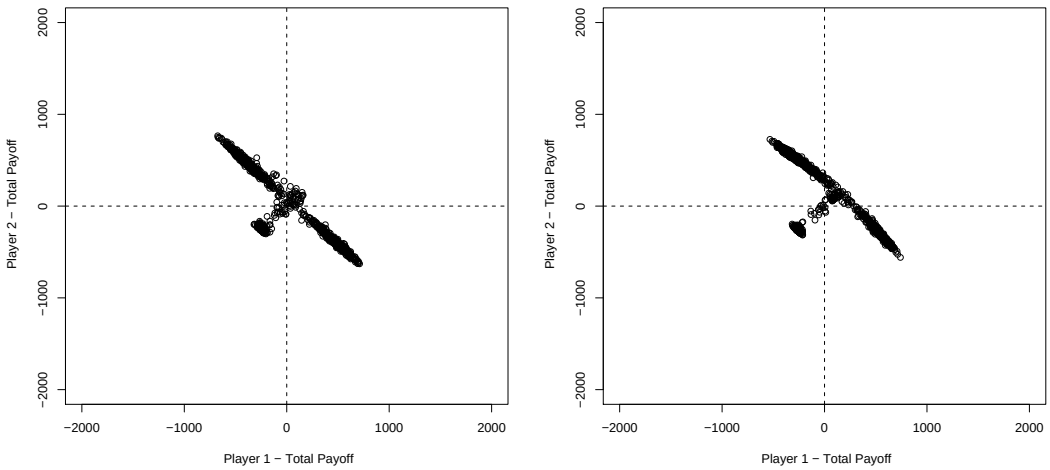
MODEL 1 vs. PAV



(a) (b)

Distribution of player scores when MODEL 1 plays against PAV strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

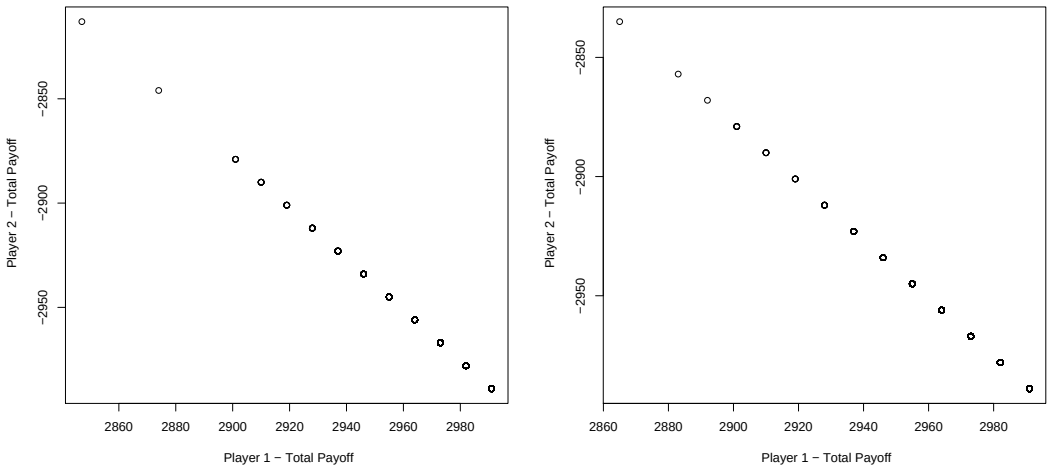
MODEL 1 vs. MODEL 1



(a) (b)

Distribution of player scores when MODEL 1 plays against MODEL 1, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

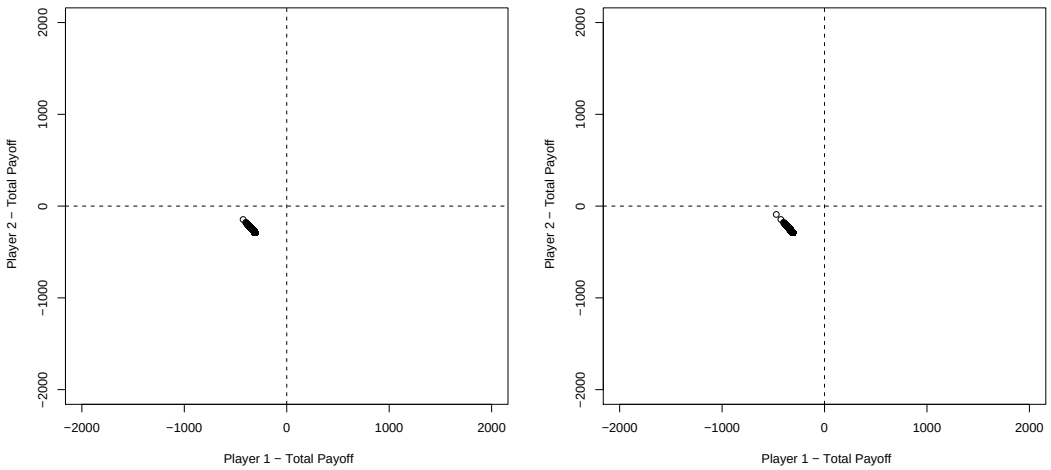
MODEL 2 vs. ALLC



(a) (b)

Distribution of player scores when MODEL 2 plays against ALLC strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

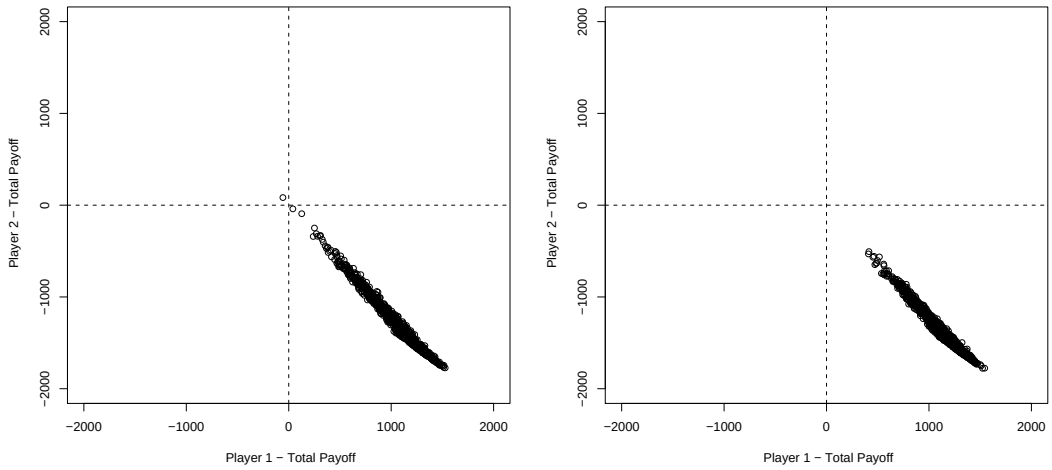
MODEL 2 vs. ALLD



(a) (b)

Distribution of player scores when MODEL 2 plays against ALLD strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

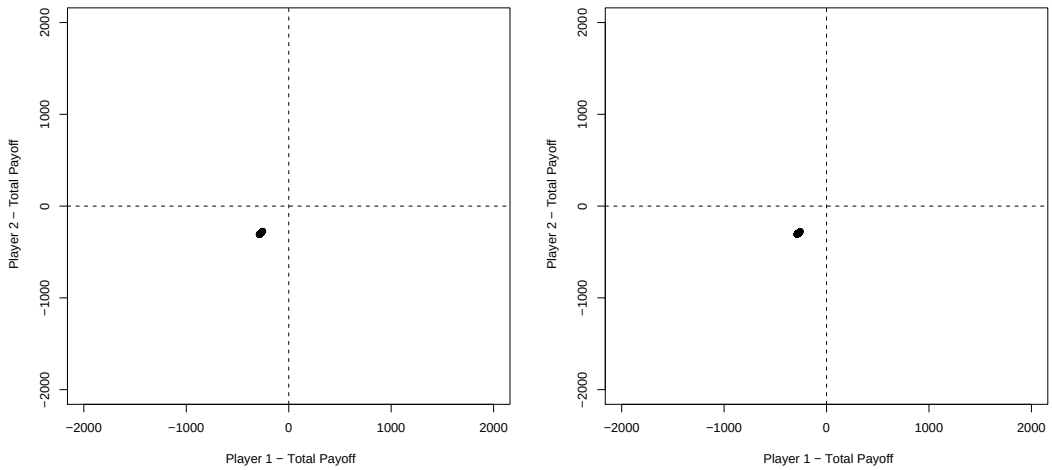
MODEL 2 vs. RAN



(a) (b)

Distribution of player scores when MODEL 2 plays against RAN strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

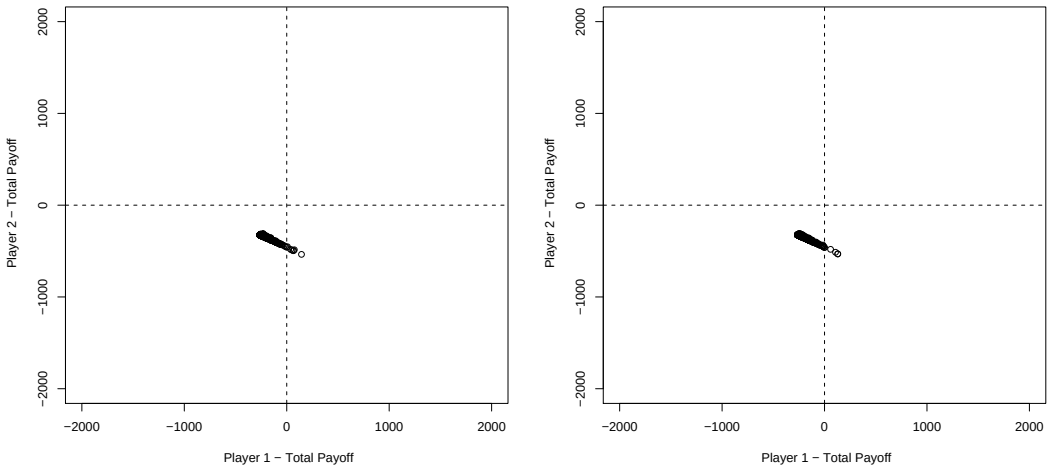
MODEL 2 vs. TFT



(a) (b)

Distribution of player scores when MODEL 2 plays against TFT strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

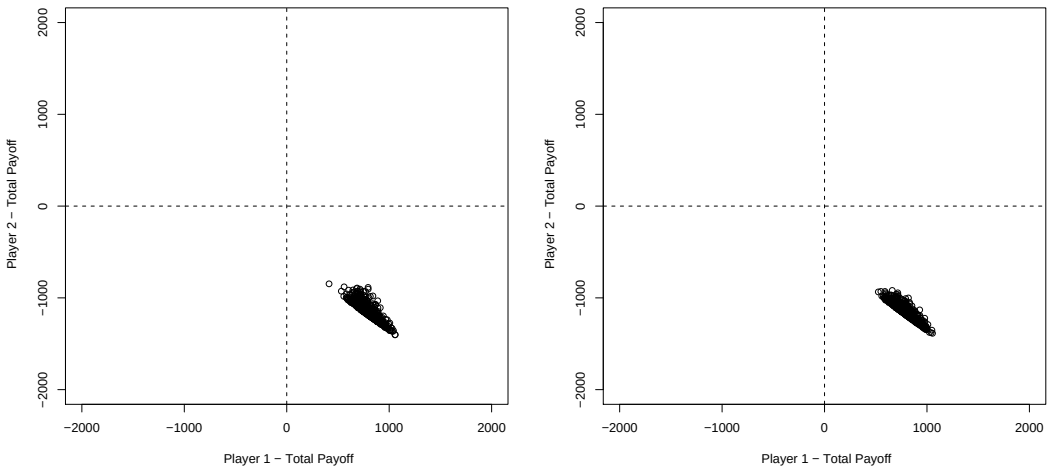
MODEL 2 vs. TFTT



(a) (b)

Distribution of player scores when MODEL 2 plays against TFTT strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

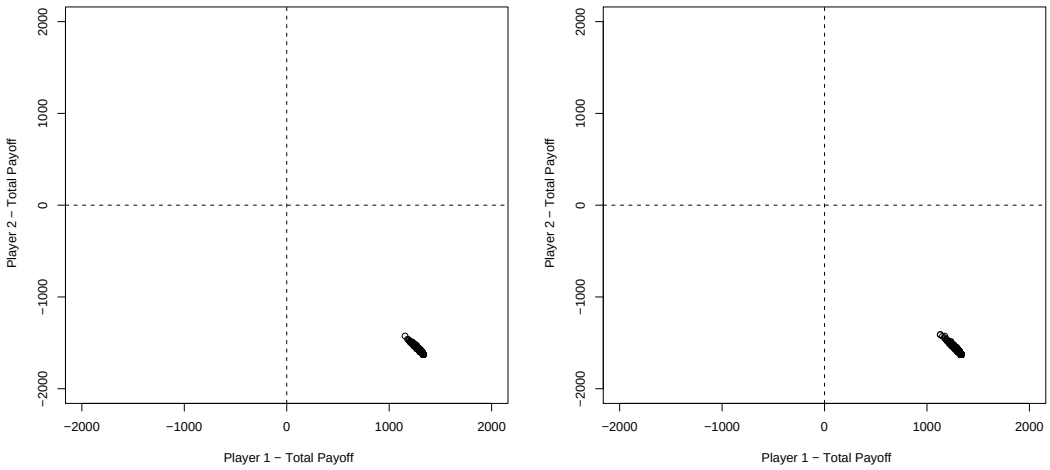
MODEL 2 vs. TFTF



(a) (b)

Distribution of player scores when MODEL 2 plays against TFTF strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

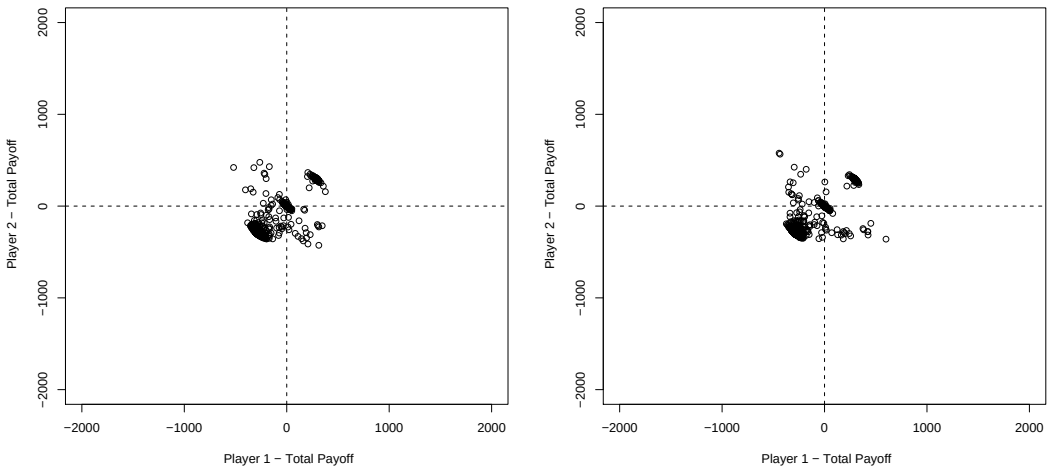
MODEL 2 vs. PAV



(a) (b)

Distribution of player scores when MODEL 2 plays against PAV strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

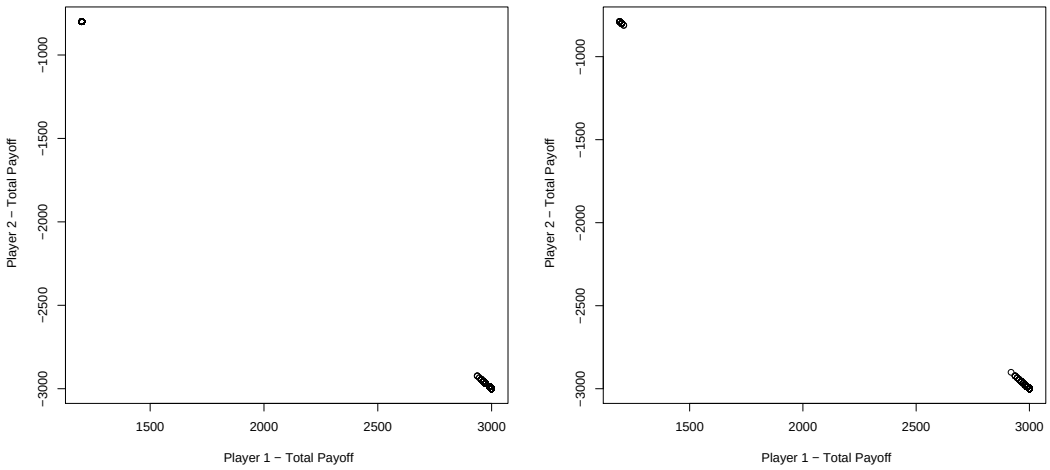
MODEL 2 vs. MODEL 2



(a) (b)

Distribution of player scores when MODEL 2 plays against MODEL 2, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

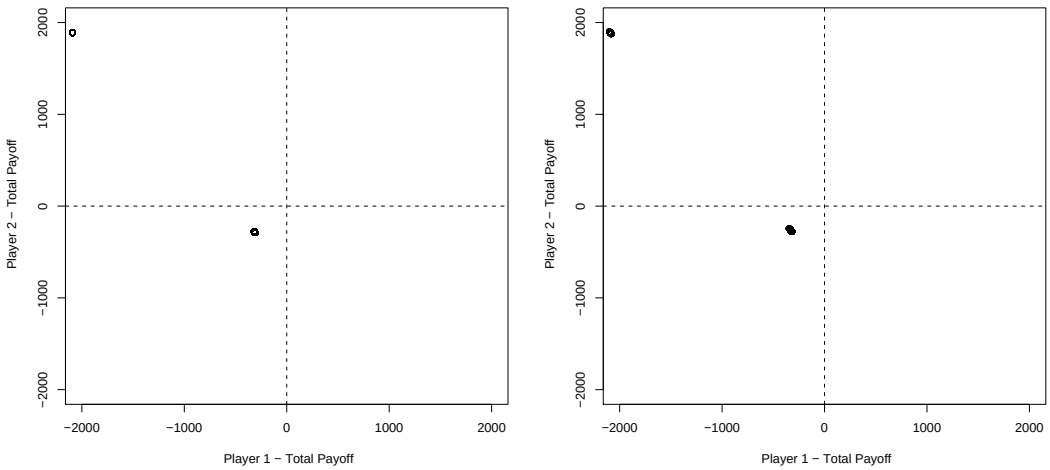
MODEL 3 vs. ALLC



(a) (b)

Distribution of player scores when MODEL 3 plays against ALLC strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

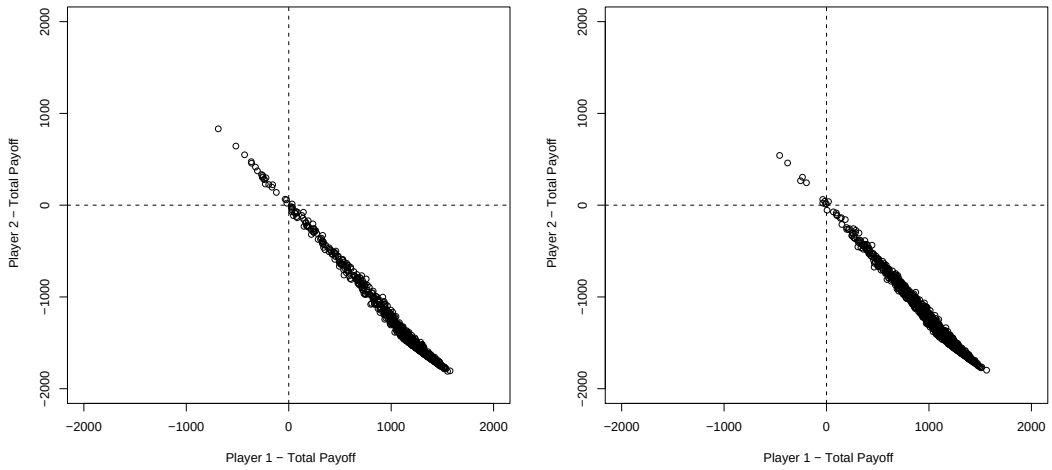
MODEL 3 vs. ALLD



(a) (b)

Distribution of player scores when MODEL 3 plays against ALLD strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

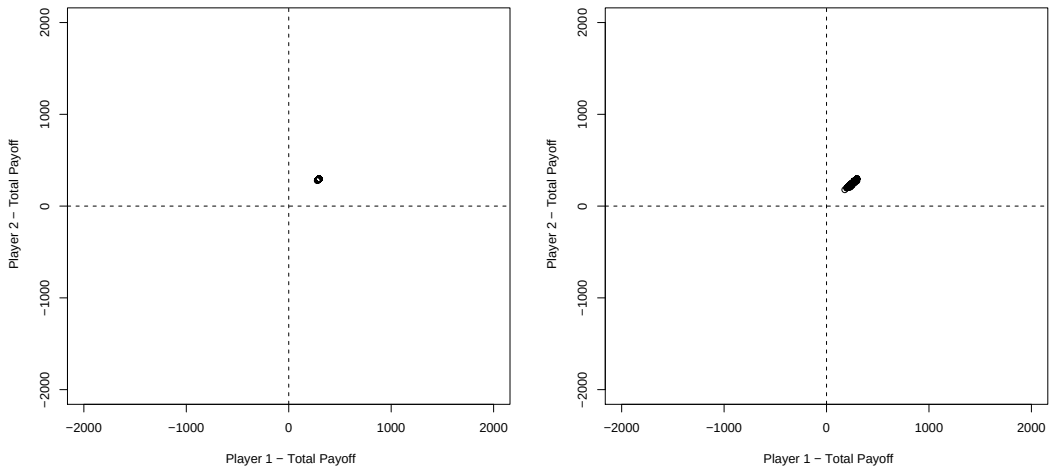
MODEL 3 vs. RAN



(a) (b)

Distribution of player scores when MODEL 3 plays against RAN strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

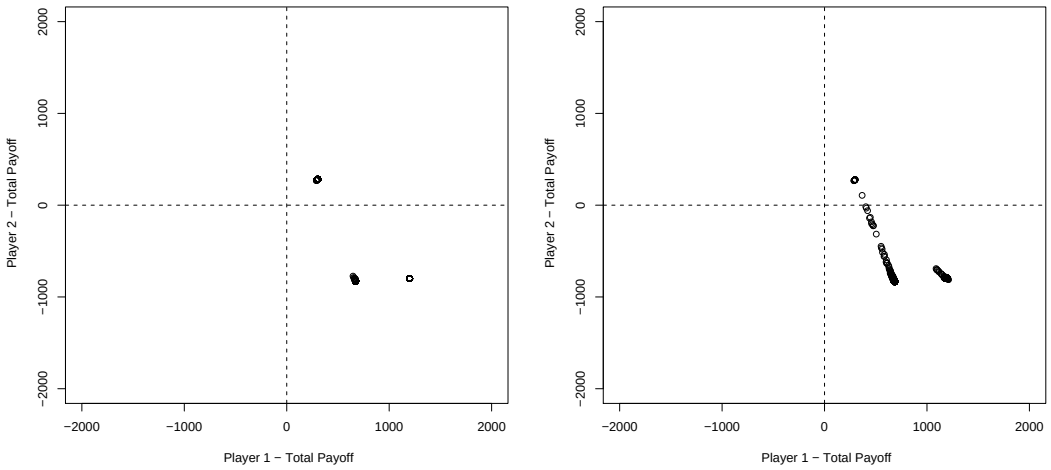
MODEL 3 vs. TFT



(a) (b)

Distribution of player scores when MODEL 3 plays against TFT strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

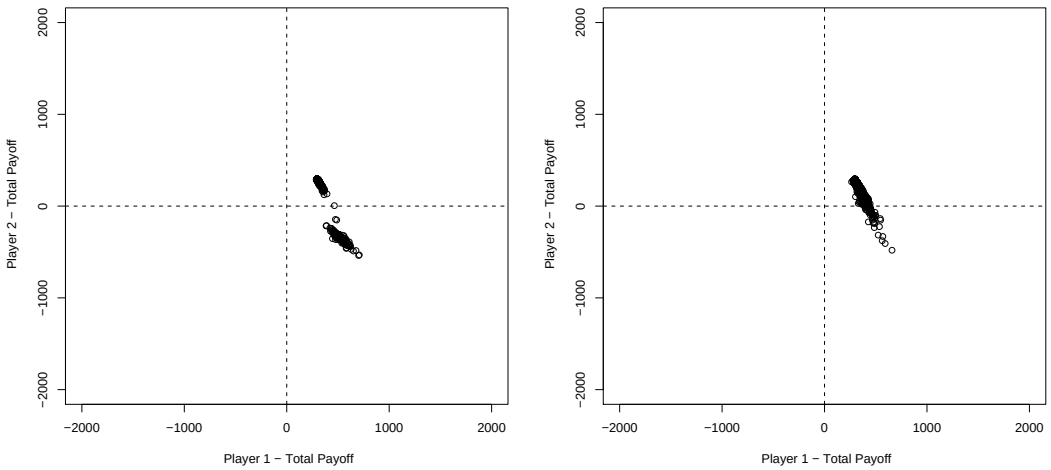
MODEL 3 vs. TFTT



(a) (b)

Distribution of player scores when MODEL 3 plays against TFTT strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

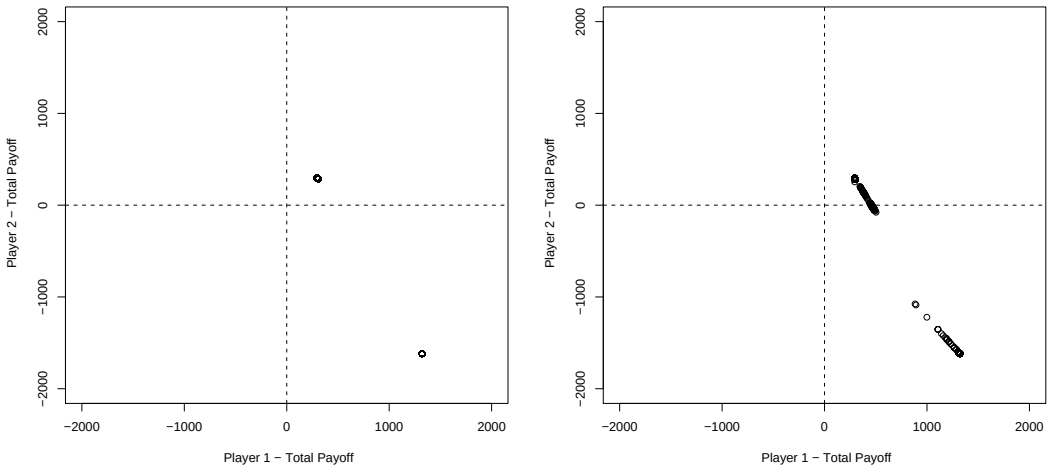
MODEL 3 vs. TFTF



(a) (b)

Distribution of player scores when MODEL 3 plays against TFTF strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

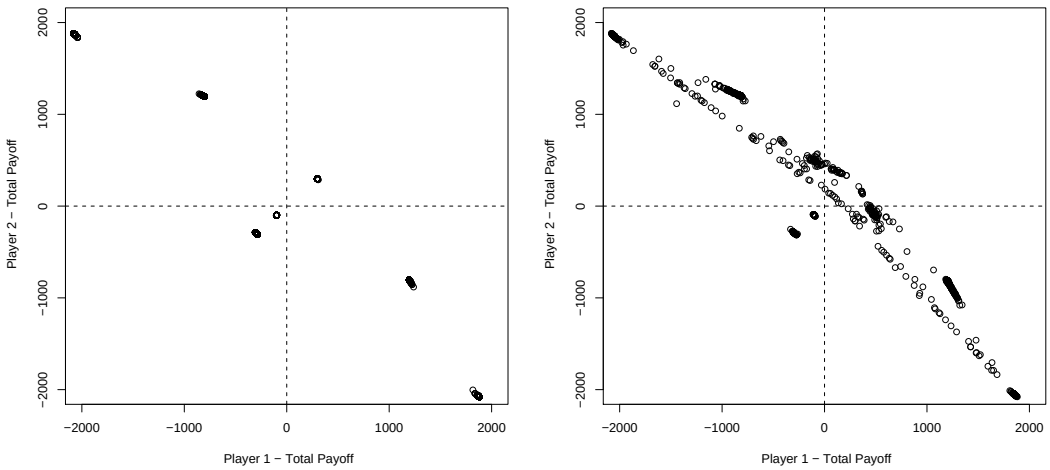
MODEL 3 vs. PAV



(a) (b)

Distribution of player scores when MODEL 3 plays against PAV strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

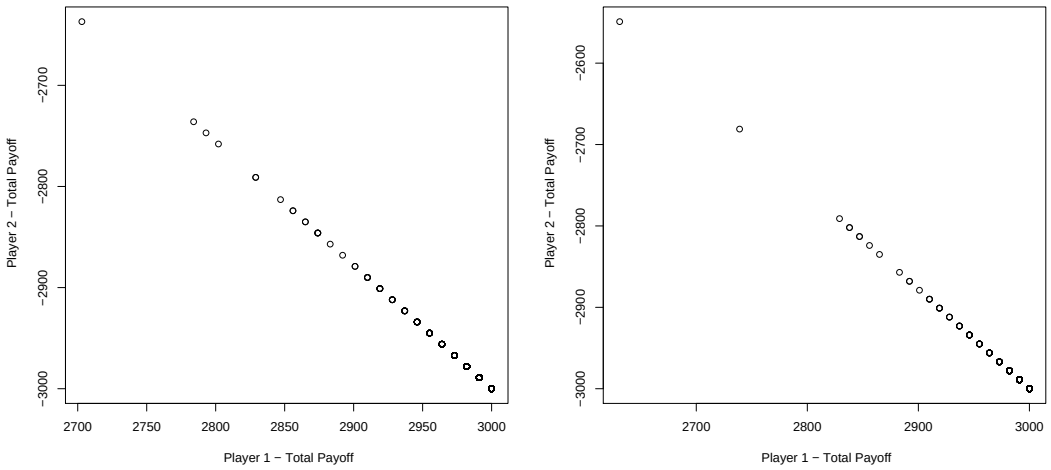
MODEL 3 vs. MODEL 3



(a) (b)

Distribution of player scores when MODEL 3 plays against MODEL 1, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

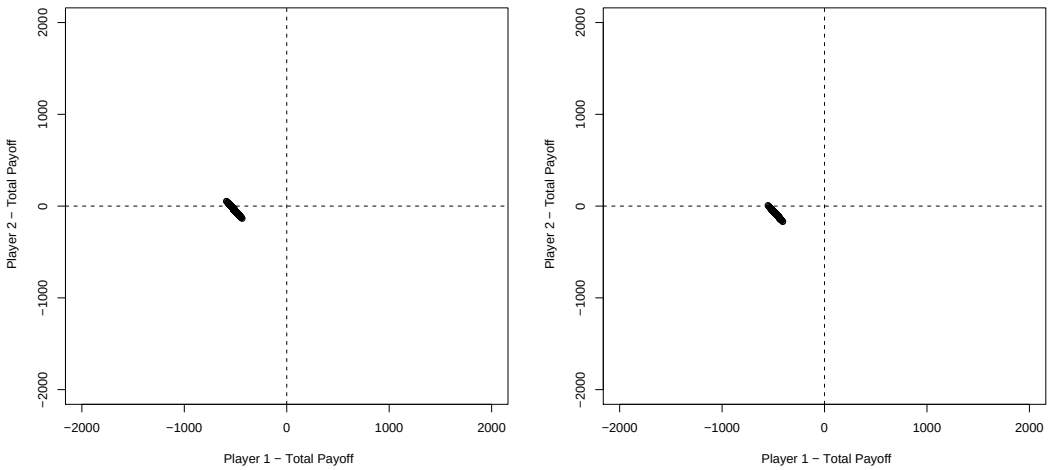
MODEL 4 vs. ALLC



(a) (b)

Distribution of player scores when MODEL 4 plays against ALLC strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

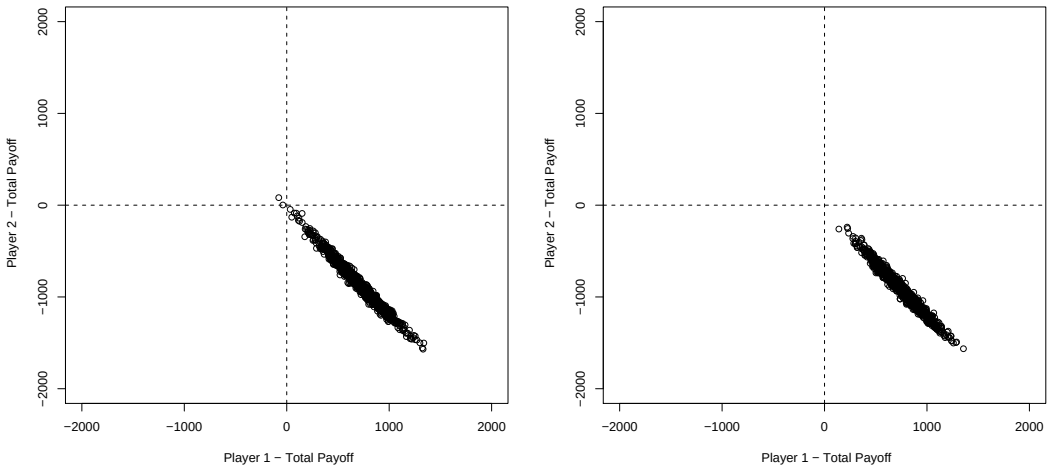
MODEL 4 vs. ALLD



(a) (b)

Distribution of player scores when MODEL 4 plays against ALLD strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

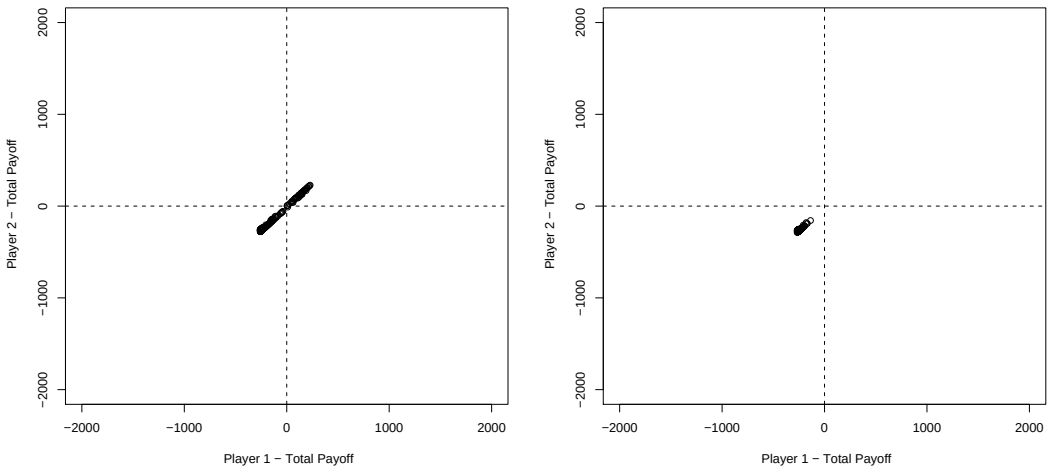
MODEL 4 vs. RAN



(a) (b)

Distribution of player scores when MODEL 4 plays against RAN strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

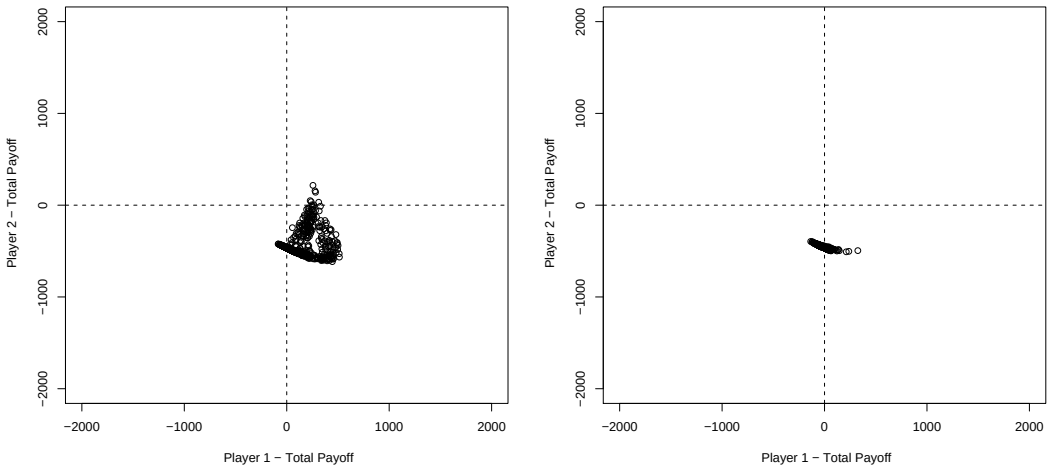
MODEL 4 vs. TFT



(a) (b)

Distribution of player scores when MODEL 4 plays against TFT strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

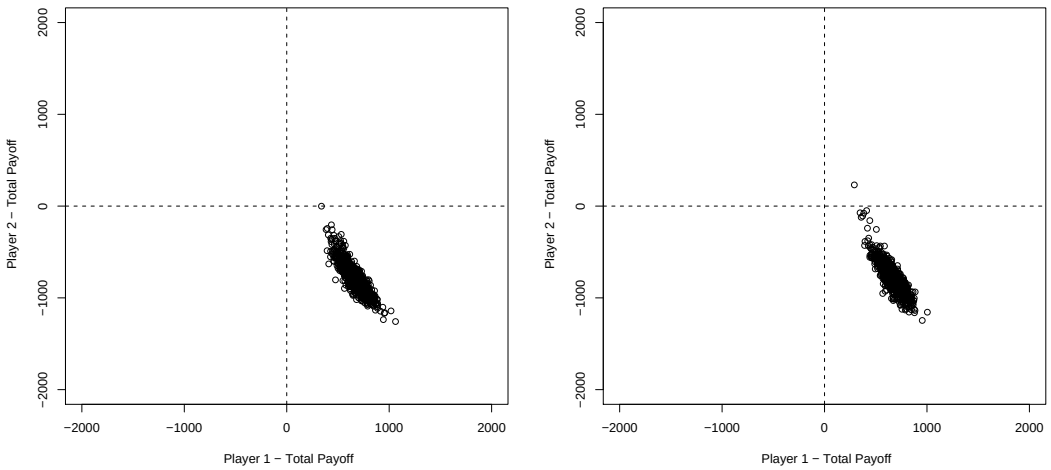
MODEL 4 vs. TFTT



(a) (b)

Distribution of player scores when MODEL 4 plays against TFTT strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

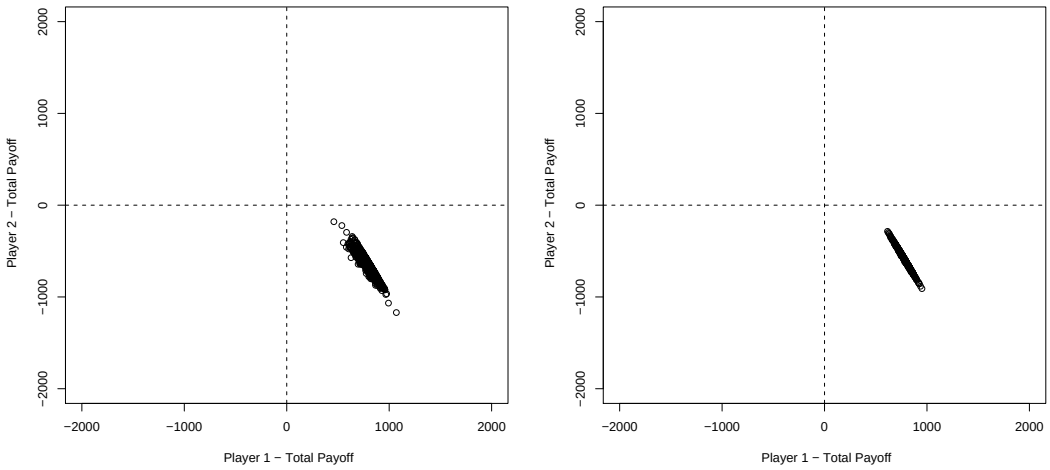
MODEL 4 vs. TFTF



(a) (b)

Distribution of player scores when MODEL 4 plays against TFTF strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

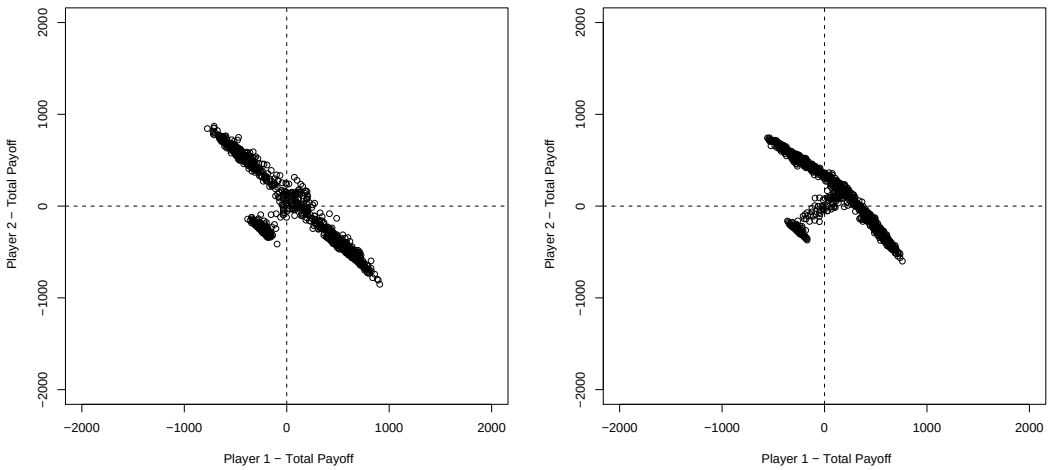
MODEL 4 vs. PAV



(a) (b)

Distribution of player scores when MODEL 4 plays against PAV strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

MODEL 4 vs. MODEL 4

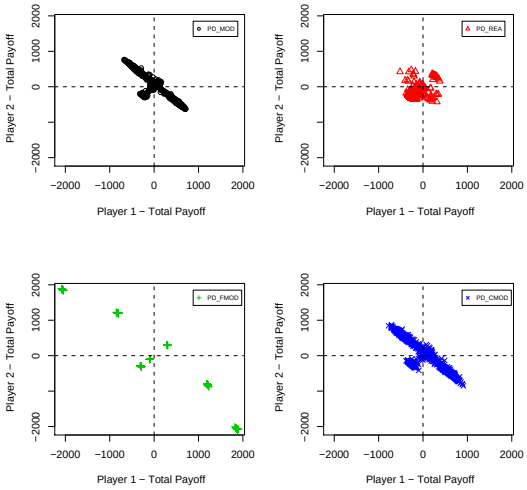


(a) (b)

Distribution of player scores when MODEL 4 plays against MODEL 4, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

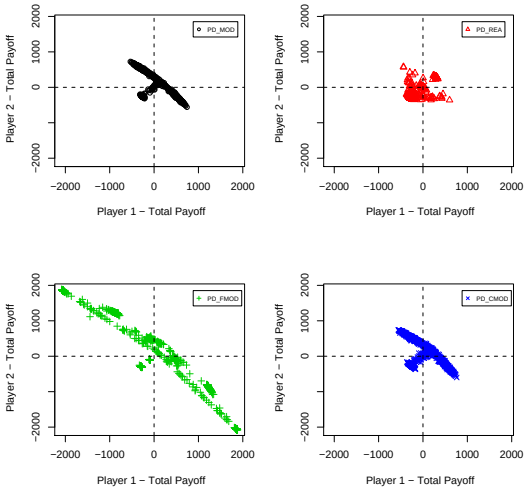
E Comparison of Model Scores

Model Scores for decay = 0.5



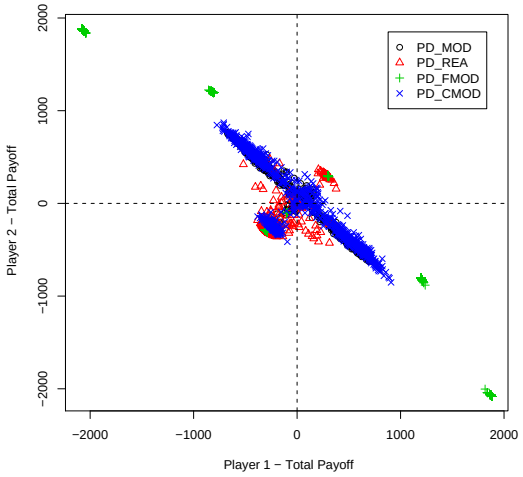
Comparison of score plots when Model plays with itself, decay = 0.5, noise = 0.14, L=30

Model Scores for decay = 0.8



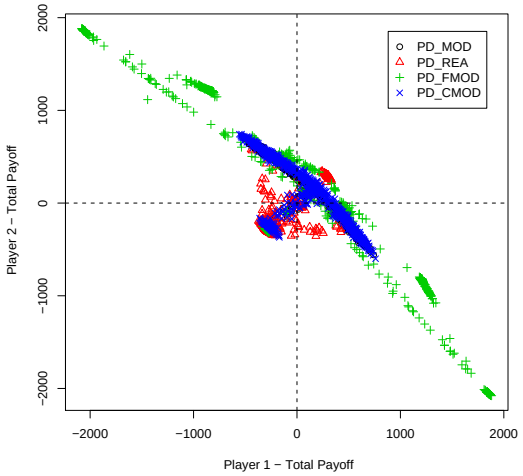
Comparison of score plots when Model plays with itself, decay = 0.8, noise = 0.14, L=30

Model Scores for decay = 0.5



Comparison of score plots when Model plays with itself, decay = 0.5, noise = 0.14, L=30

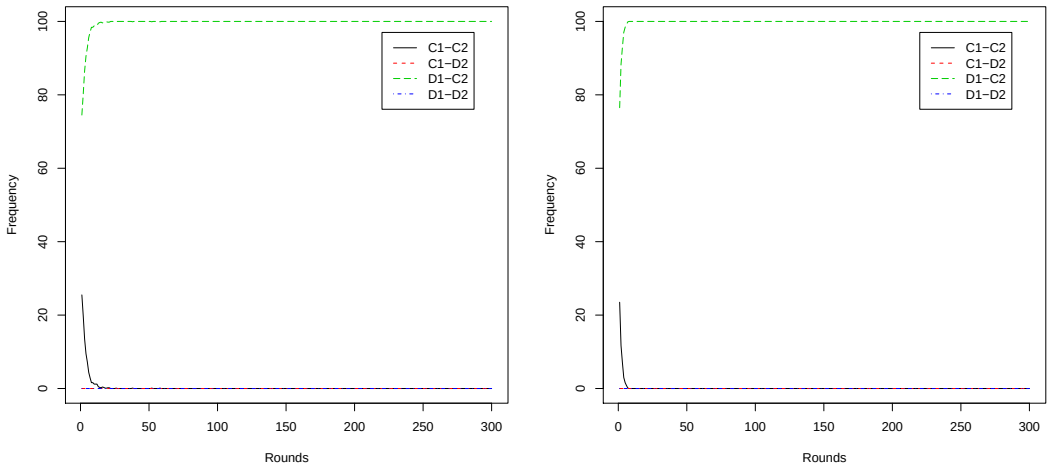
Model Scores for decay = 0.8



Comparison of score plots when Model plays with itself, decay = 0.8, noise = 0.14, L=30

F Learning in Iterated Prisoner's Dilemma

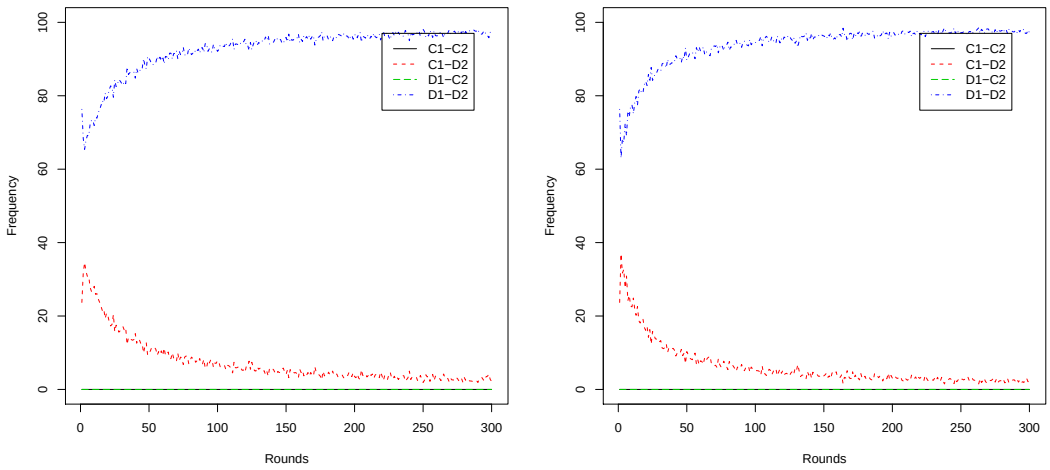
MODEL 1 vs. ALLC



(a) (b)

Frequency of outcomes as game progresses when MODEL 1 plays against ALLC strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

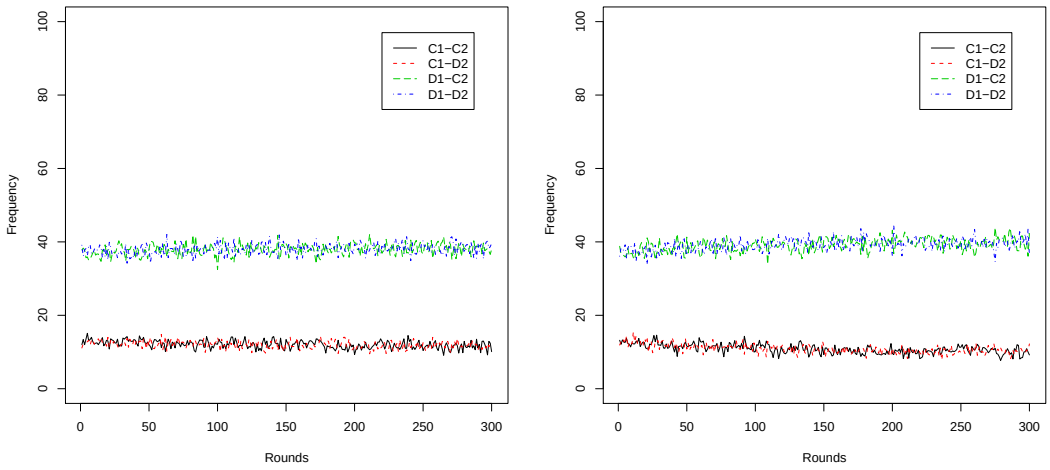
MODEL 1 vs. ALLD



(a) (b)

Frequency of outcomes as game progresses when MODEL 1 plays against ALLD strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

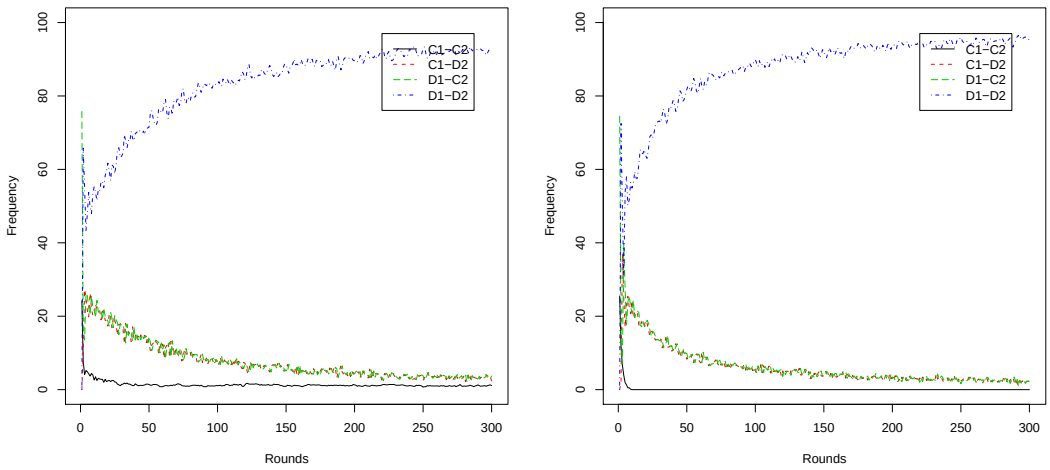
MODEL 1 vs. RAN



(a) (b)

Frequency of outcomes as game progresses when MODEL 1 plays against RAN strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

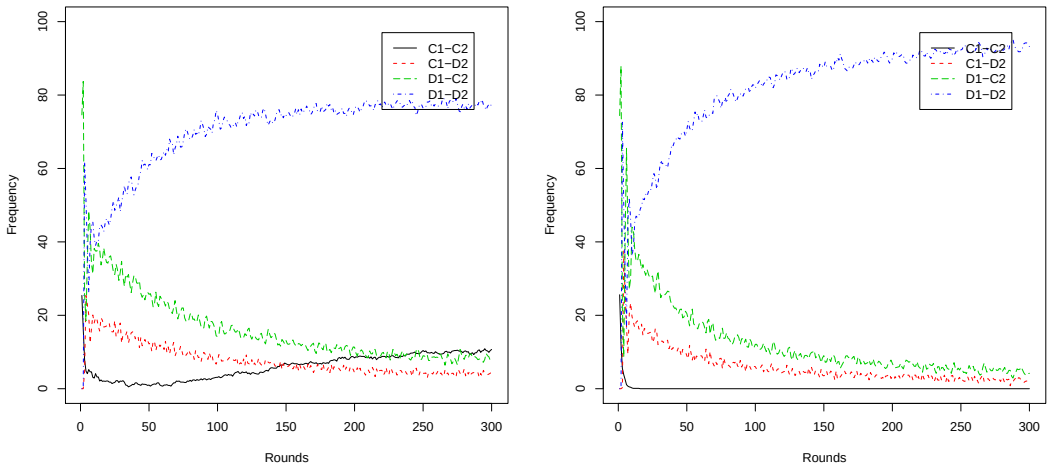
MODEL 1 vs. TFT



(a) (b)

Frequency of outcomes as game progresses when MODEL 1 plays against TFT strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

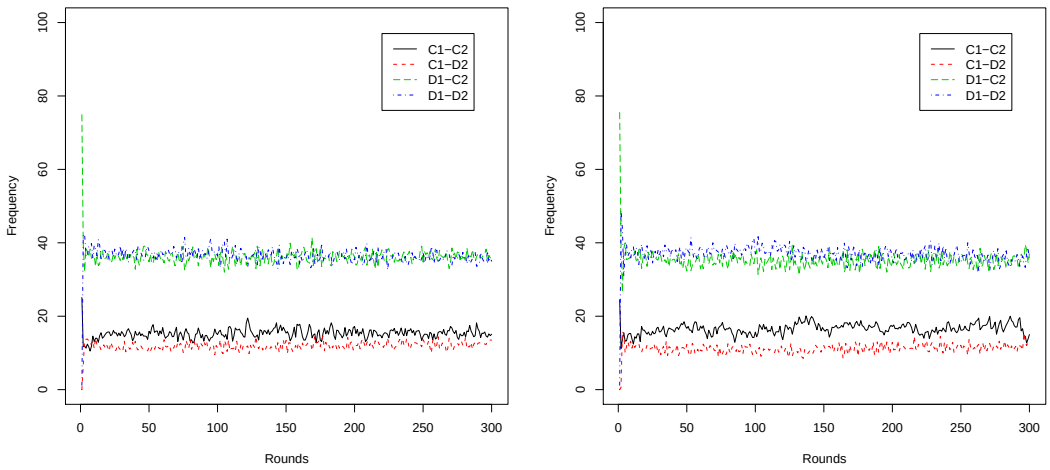
MODEL 1 vs. TFTT



(a) (b)

Frequency of outcomes as game progresses when MODEL 1 plays against TFTT strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

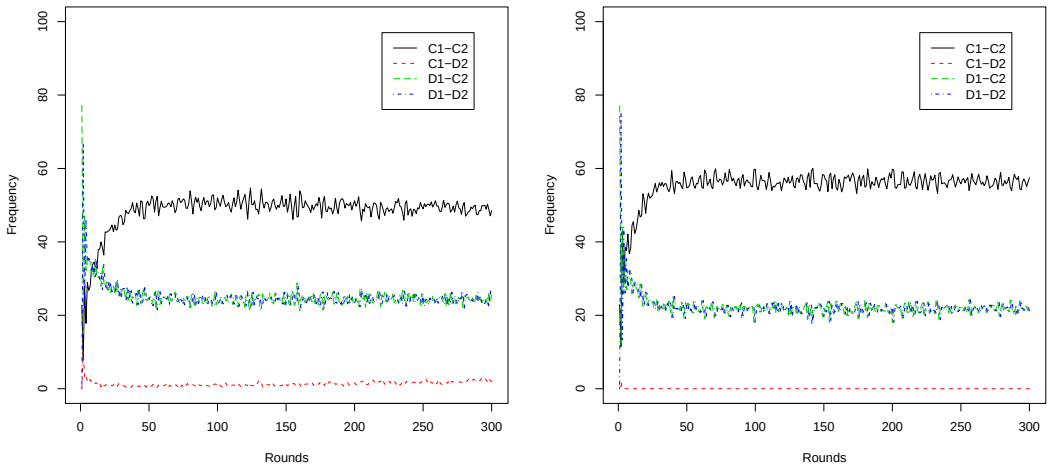
MODEL 1 vs. TFTF



(a) (b)

Frequency of outcomes as game progresses when MODEL 1 plays against TFTF strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

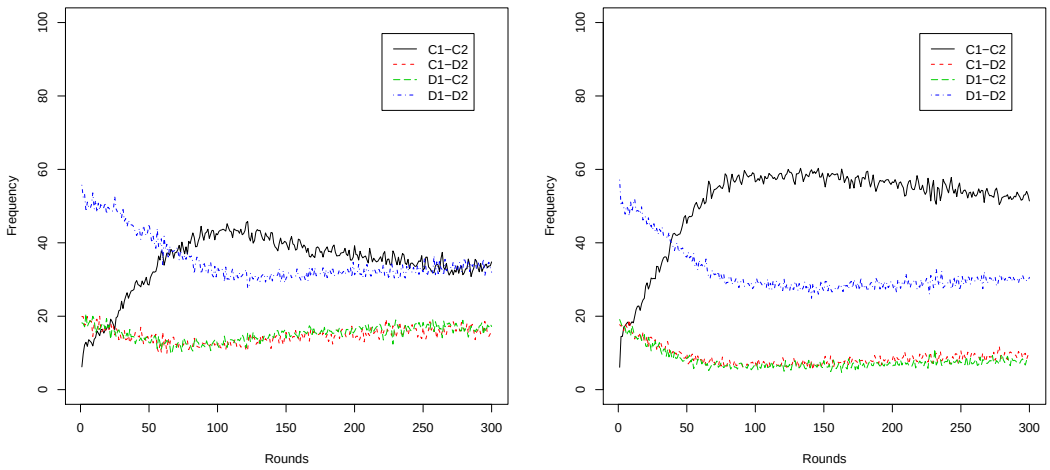
MODEL 1 vs. PAV



(a) (b)

Frequency of outcomes as game progresses when MODEL 1 plays against PAV strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

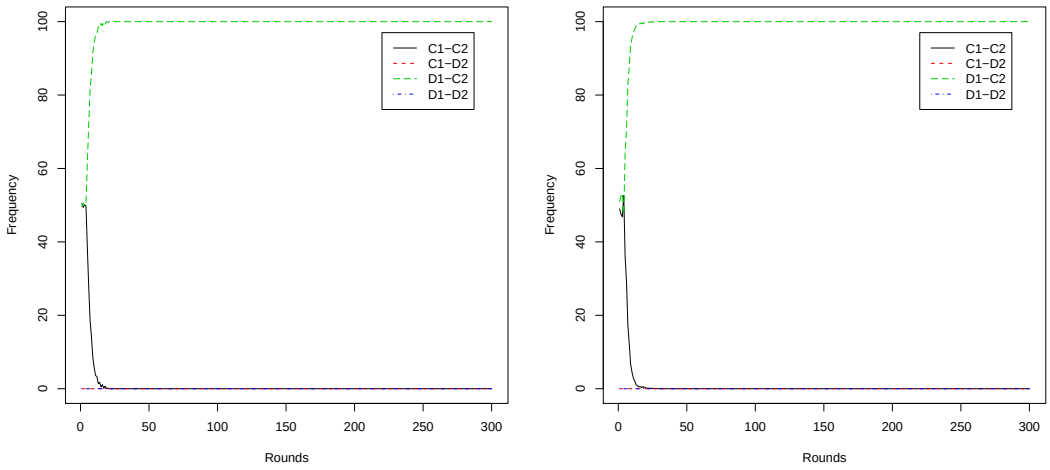
MODEL 1 vs. MODEL 1



(a) (b)

Frequency of outcomes as game progresses when MODEL 1 plays against MODEL 1, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

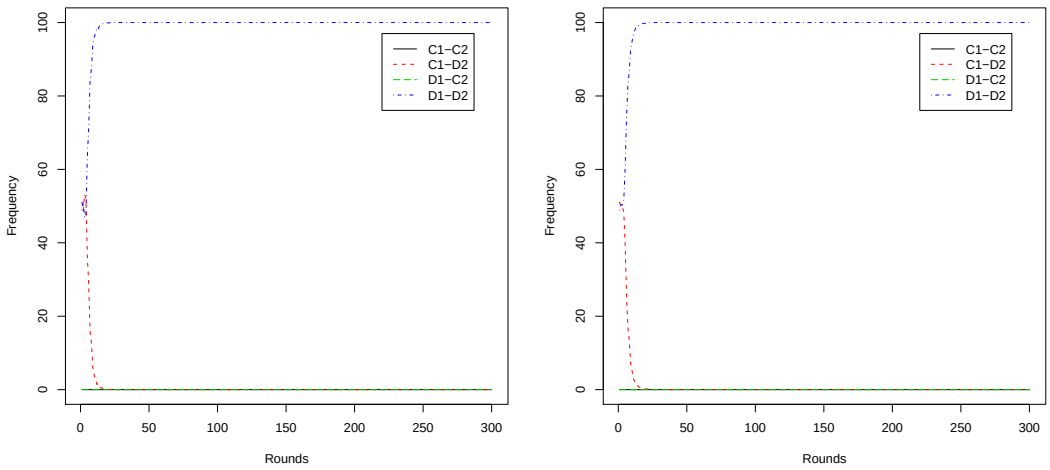
MODEL 2 vs. ALLC



(a) (b)

Frequency of outcomes as game progresses when MODEL 2 plays against ALLC strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

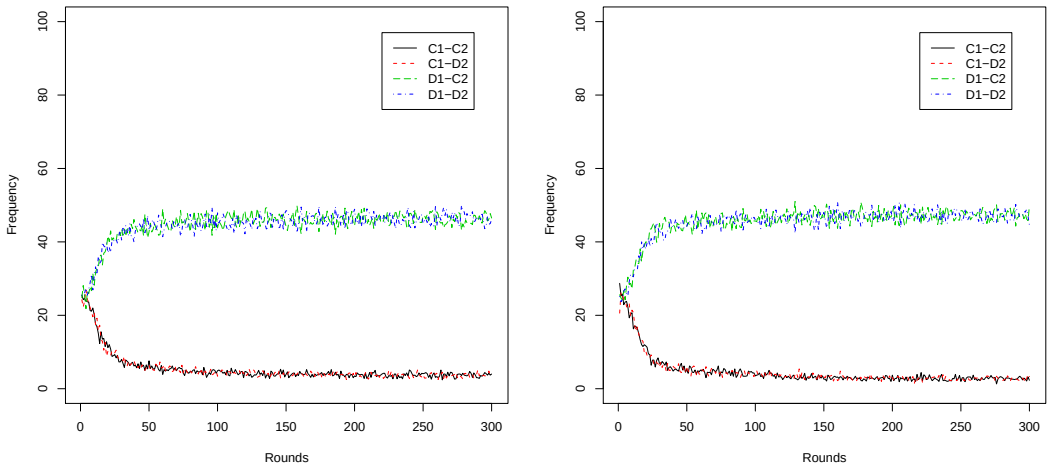
MODEL 2 vs. ALLD



(a) (b)

Frequency of outcomes as game progresses when MODEL 2 plays against ALLD strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

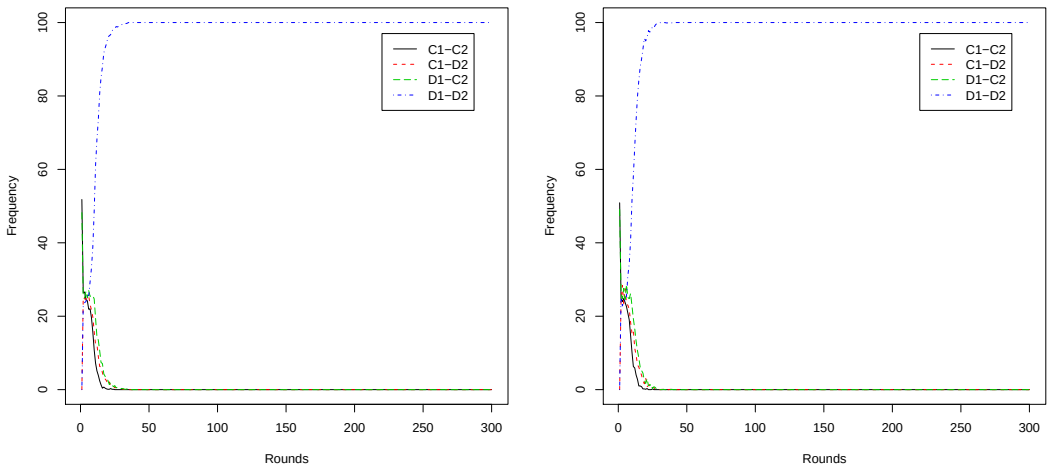
MODEL 2 vs. RAN



(a) (b)

Frequency of outcomes as game progresses when MODEL 2 plays against RAN strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

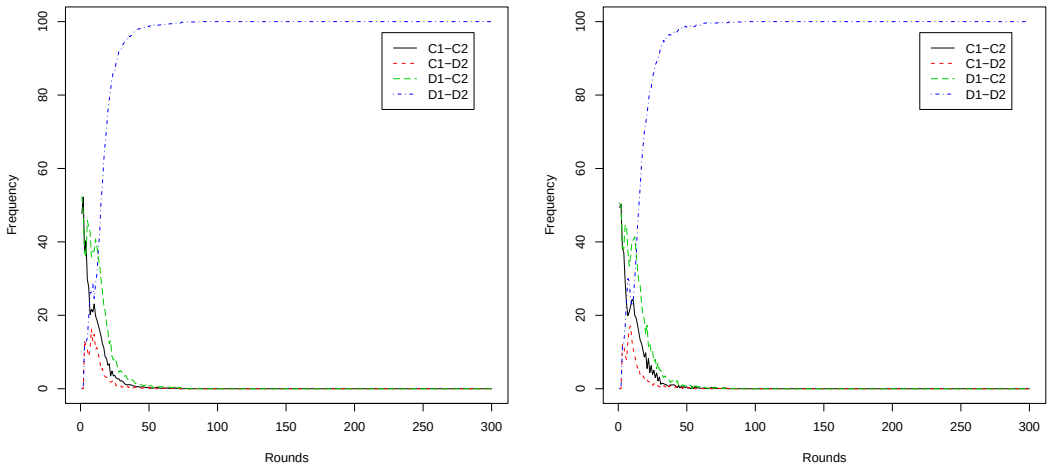
MODEL 2 vs. TFT



(a) (b)

Frequency of outcomes as game progresses when MODEL 2 plays against TFT strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

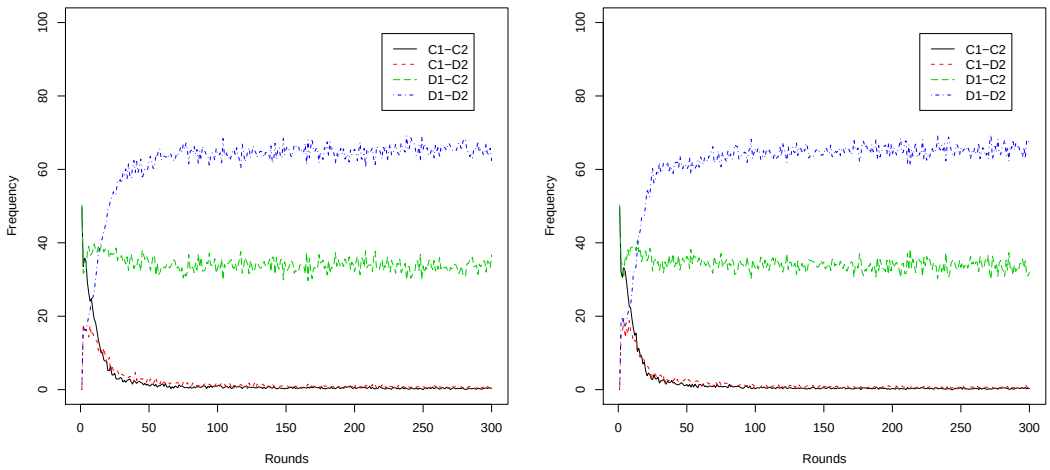
MODEL 2 vs. TFTT



(a) (b)

Frequency of outcomes as game progresses when MODEL 2 plays against TFTT strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

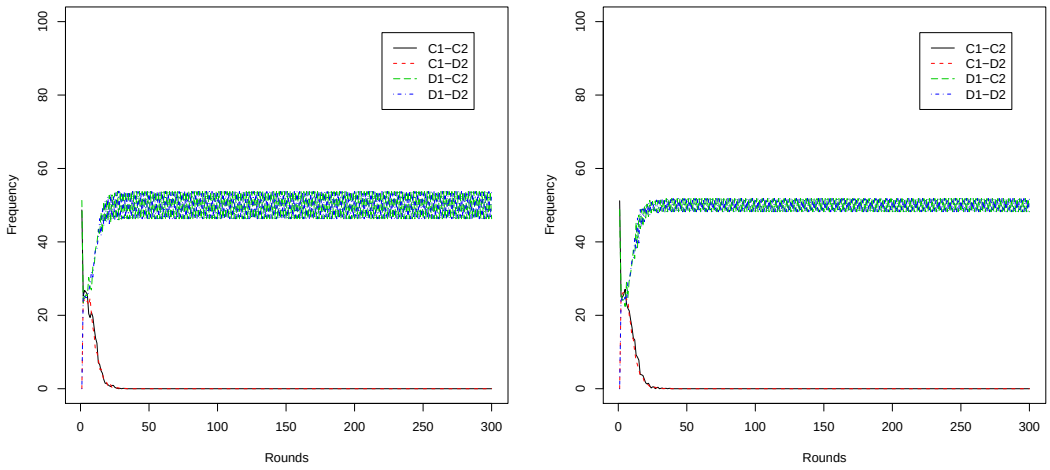
MODEL 2 vs. TFTF



(a) (b)

Frequency of outcomes as game progresses when MODEL 2 plays against TFTF strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

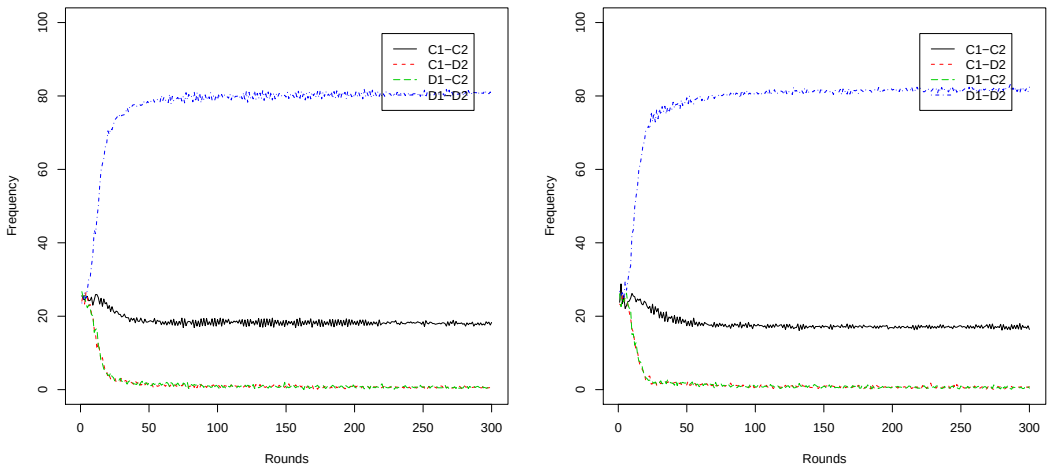
MODEL 2 vs. PAV



(a) (b)

Frequency of outcomes as game progresses when MODEL 2 plays against PAV strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

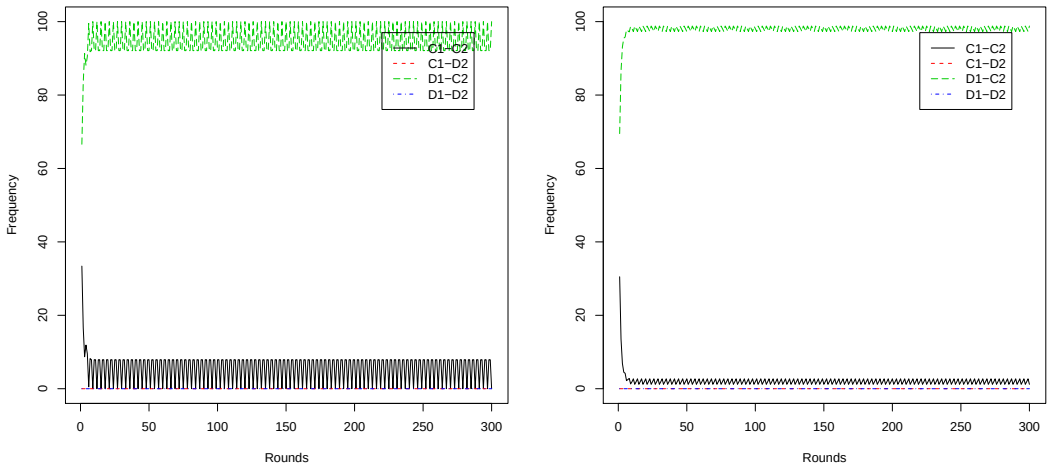
MODEL 2 vs. MODEL 2



(a) (b)

Frequency of outcomes as game progresses when MODEL 2 plays against MODEL 2, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

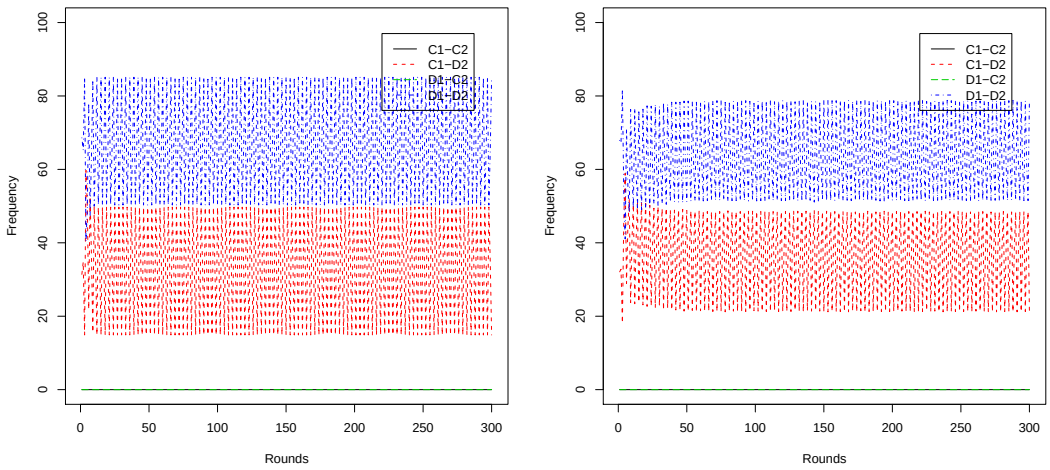
MODEL 3 vs. ALLC



(a) (b)

Frequency of outcomes as game progresses when MODEL 3 plays against ALLC strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

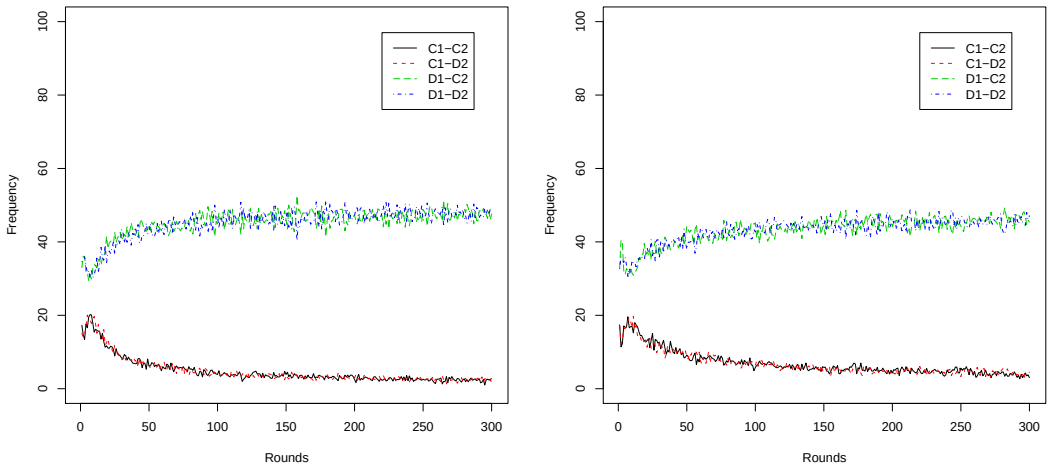
MODEL 3 vs. ALLD



(a) (b)

Frequency of outcomes as game progresses when MODEL 3 plays against ALLD strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

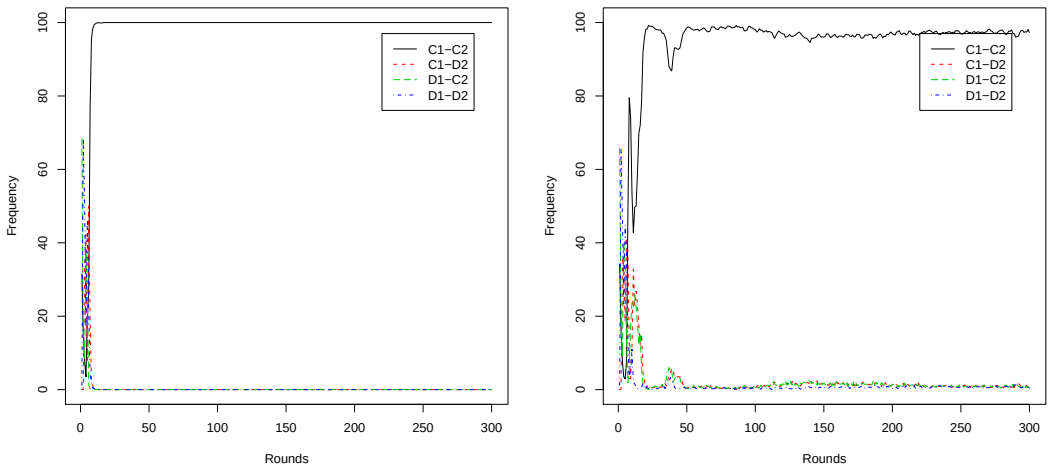
MODEL 3 vs. RAN



(a) (b)

Frequency of outcomes as game progresses when MODEL 3 plays against RAN strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

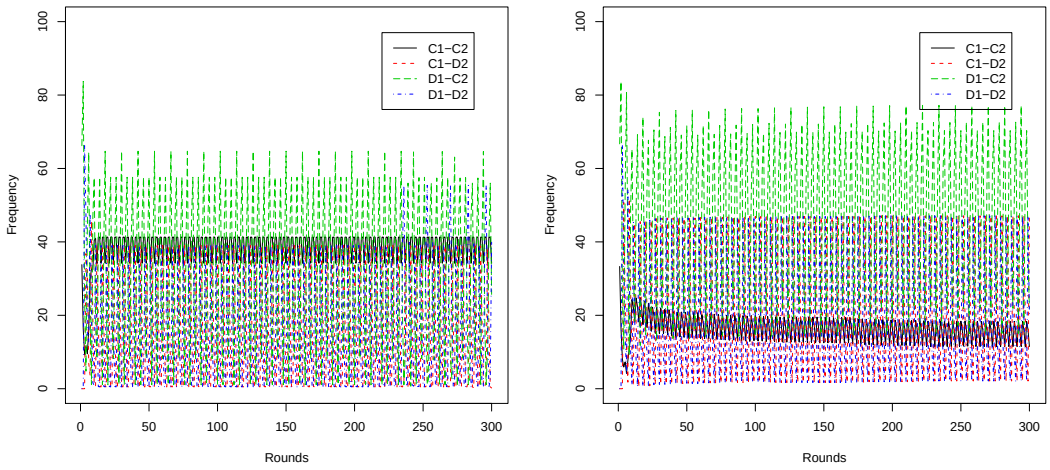
MODEL 3 vs. TFT



(a) (b)

Frequency of outcomes as game progresses when MODEL 3 plays against TFT strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

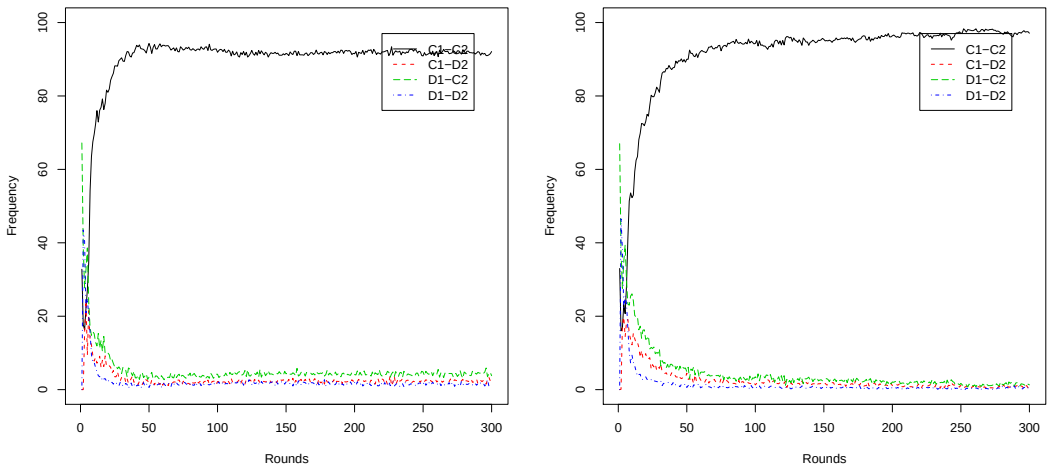
MODEL 3 vs. TFTT



(a) (b)

Frequency of outcomes as game progresses when MODEL 3 plays against TFTT strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

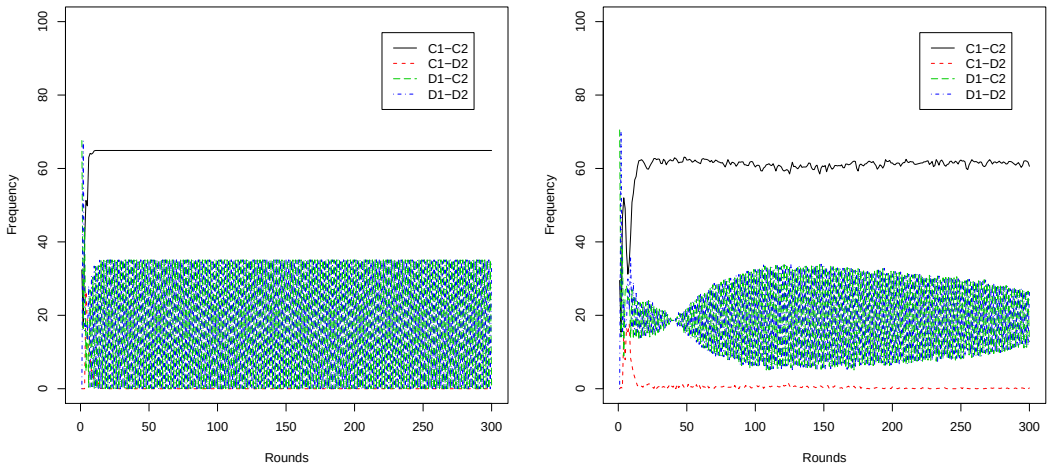
MODEL 3 vs. TFTF



(a) (b)

Frequency of outcomes as game progresses when MODEL 3 plays against TFTF strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

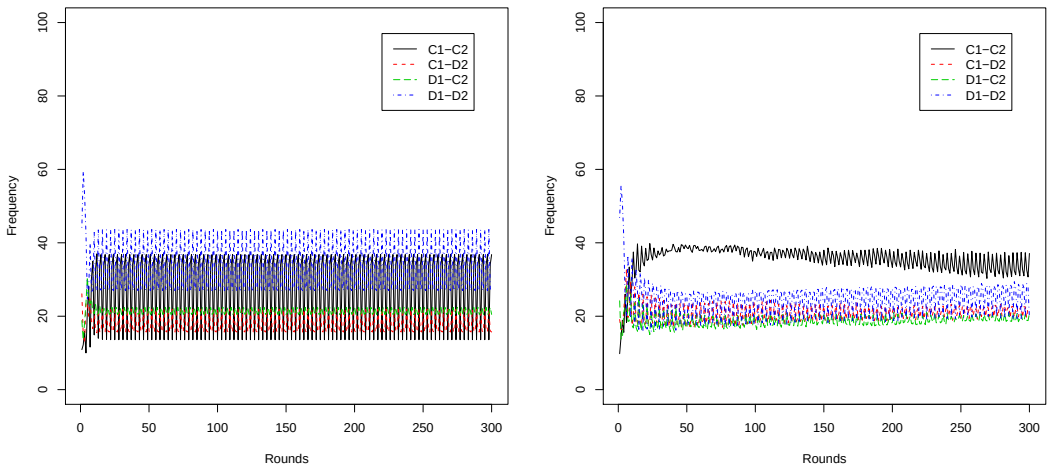
MODEL 3 vs. PAV



(a) (b)

Frequency of outcomes as game progresses when MODEL 3 plays against PAV strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

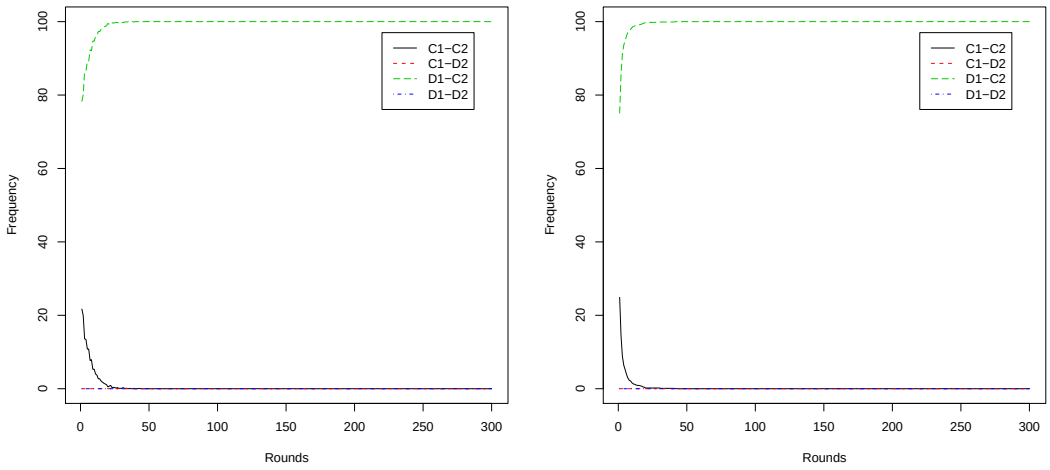
MODEL 3 vs. MODEL 3



(a) (b)

Frequency of outcomes as game progresses when MODEL 3 plays against MODEL 1, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

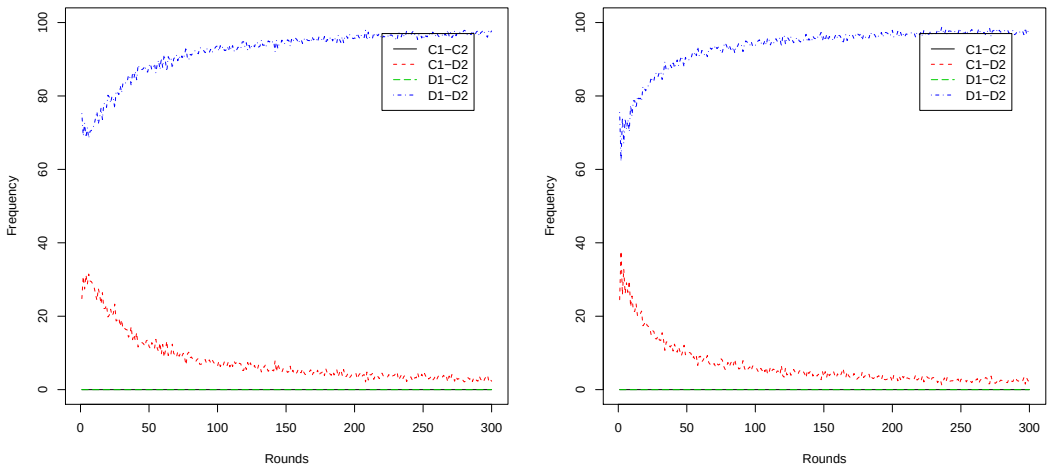
MODEL 4 vs. ALLC



(a) (b)

Frequency of outcomes as game progresses when MODEL 4 plays against ALLC strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

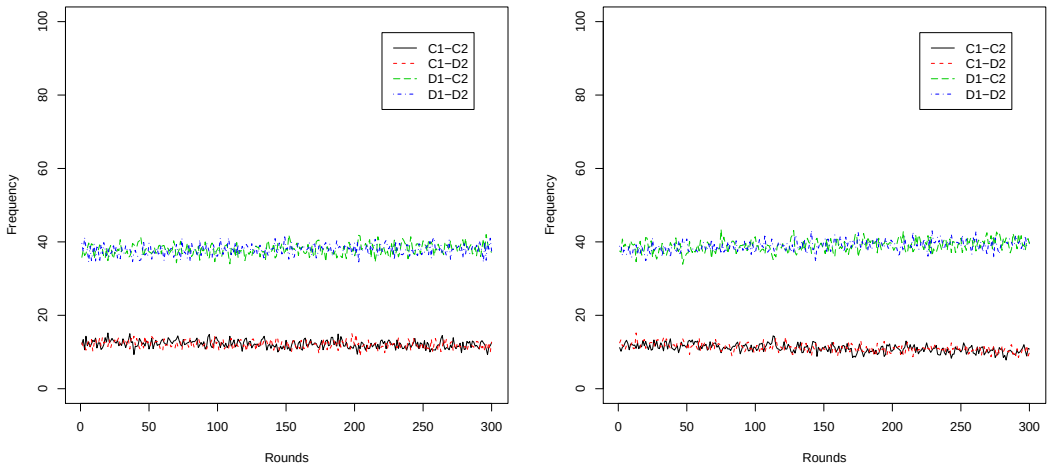
MODEL 4 vs. ALLD



(a) (b)

Frequency of outcomes as game progresses when MODEL 4 plays against ALLD strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

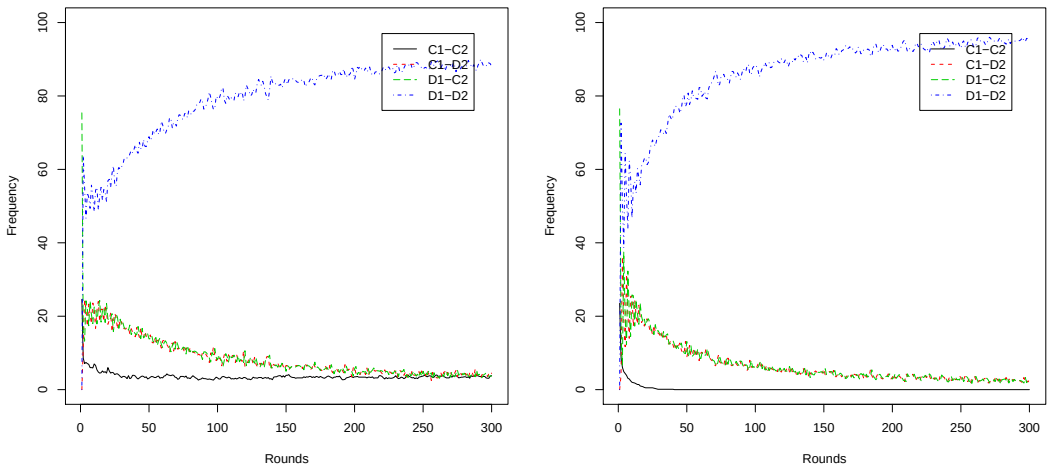
MODEL 4 vs. RAN



(a) (b)

Frequency of outcomes as game progresses when MODEL 4 plays against RAN strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

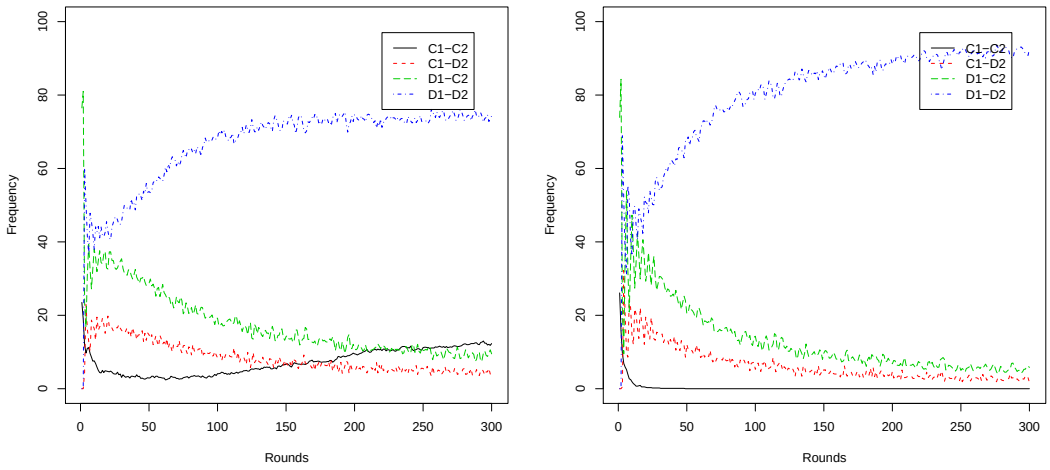
MODEL 4 vs. TFT



(a) (b)

Frequency of outcomes as game progresses when MODEL 4 plays against TFT strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

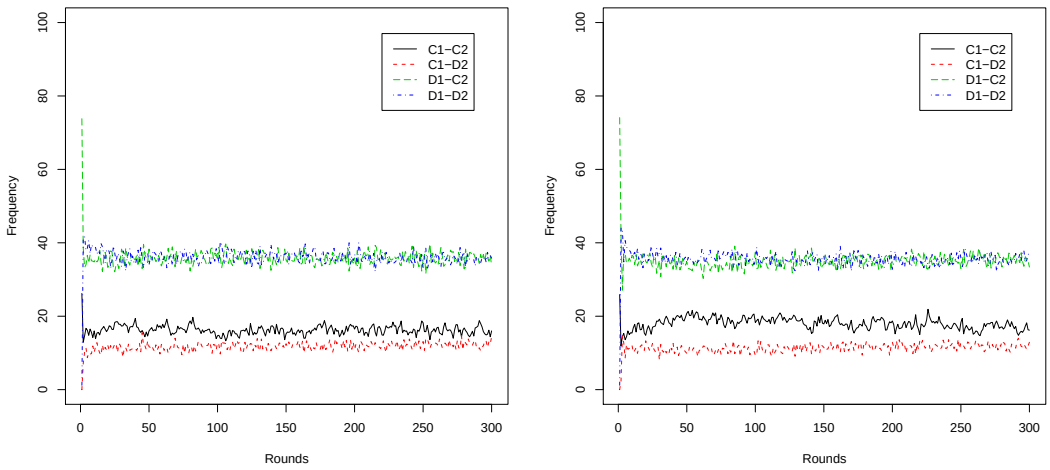
MODEL 4 vs. TFTT



(a) (b)

Frequency of outcomes as game progresses when MODEL 4 plays against TFTT strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

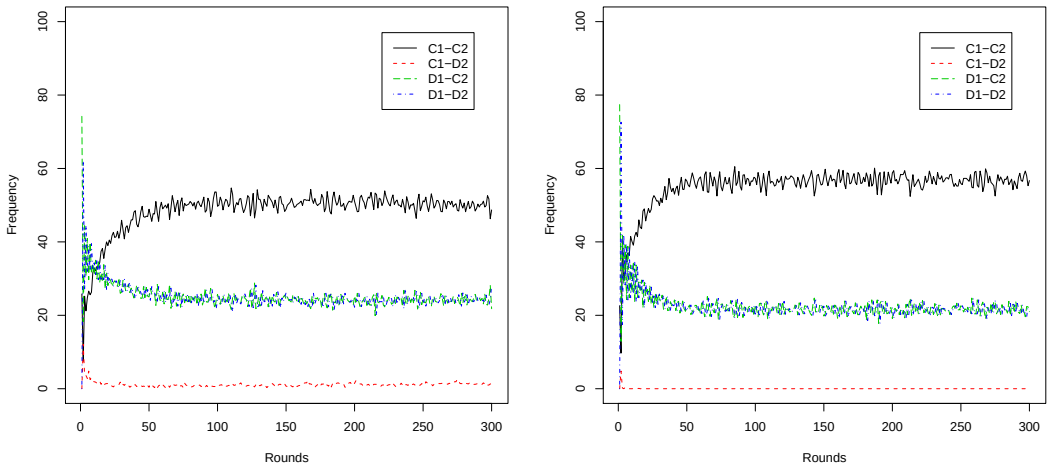
MODEL 4 vs. TFTF



(a) (b)

Frequency of outcomes as game progresses when MODEL 4 plays against TFTF strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

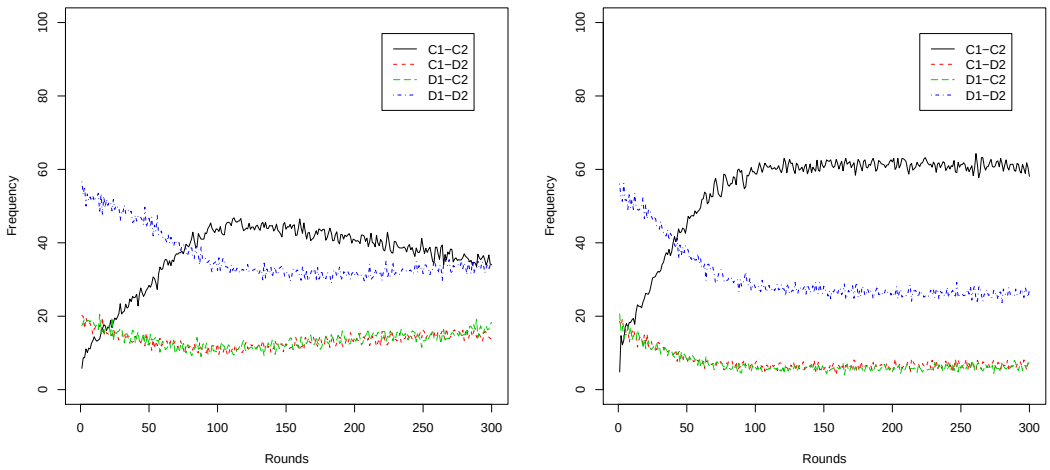
MODEL 4 vs. PAV



(a) (b)

Frequency of outcomes as game progresses when MODEL 4 plays against PAV strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

MODEL 4 vs. MODEL 4

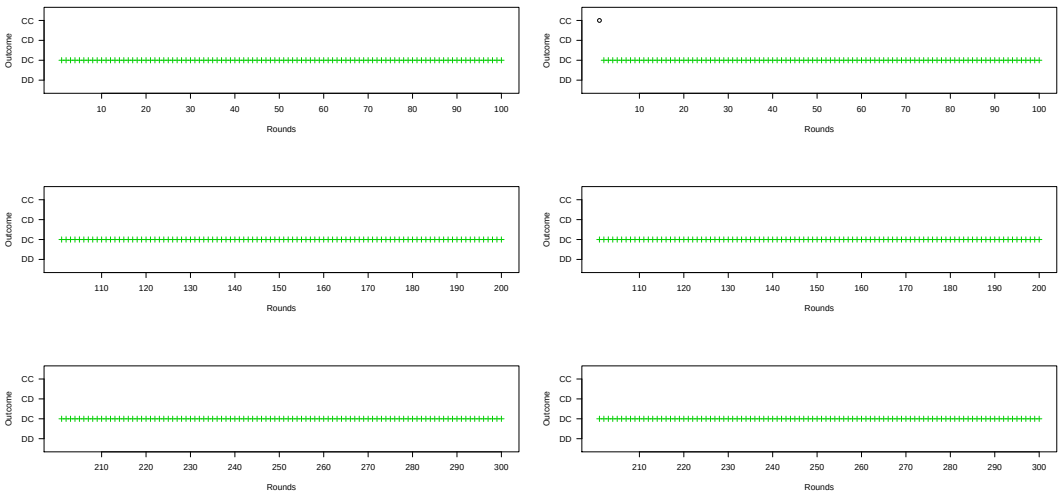


(a) (b)

Frequency of outcomes as game progresses when MODEL 4 plays against MODEL 4, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

G Outcome History of Exemplary Runs

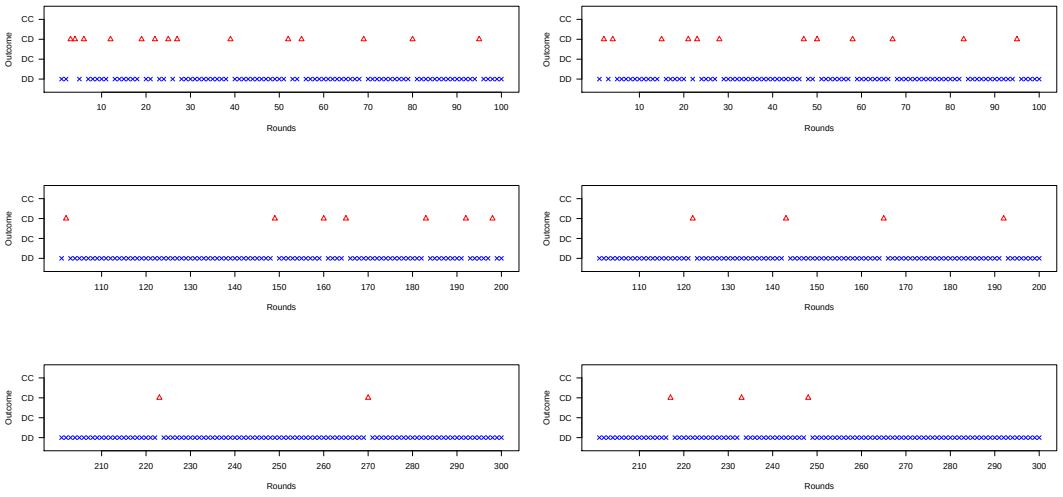
MODEL 1 vs. ALLC



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 1 plays against ALLC strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

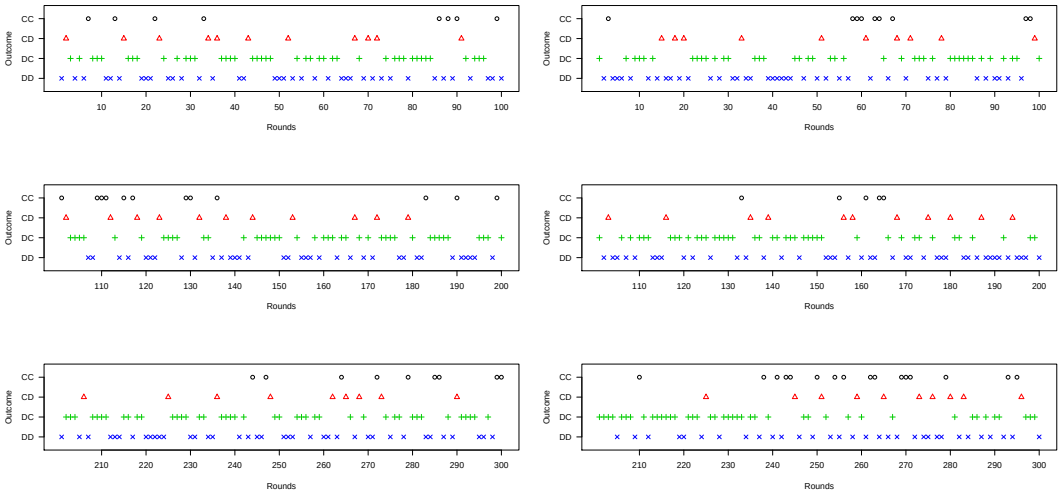
MODEL 1 vs. ALLD



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 1 plays against ALLD strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

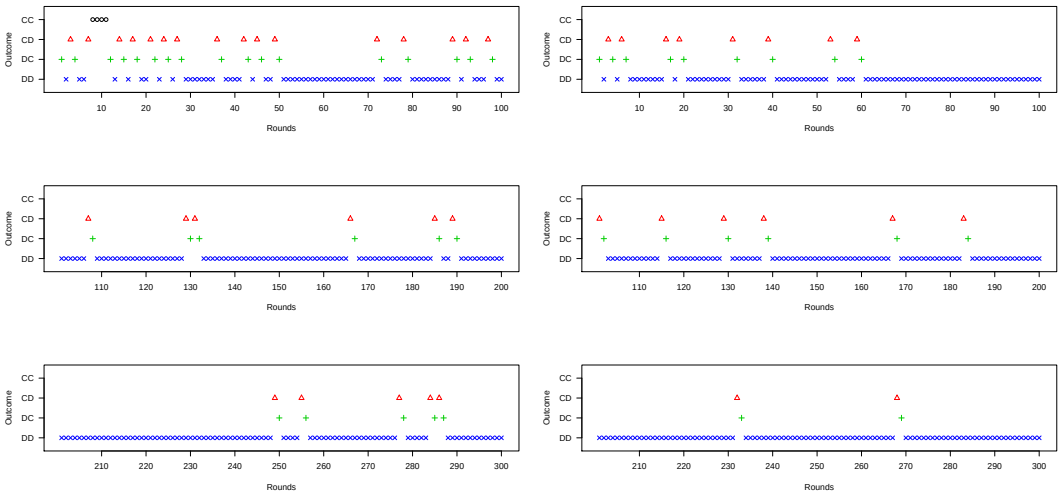
MODEL 1 vs. RAN



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 1 plays against RAN strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

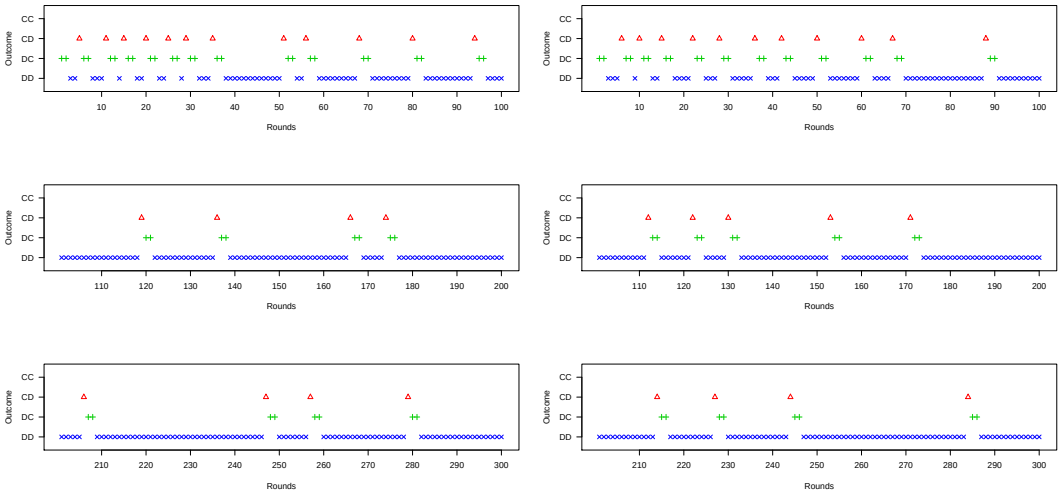
MODEL 1 vs. TFT



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 1 plays against TFT strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

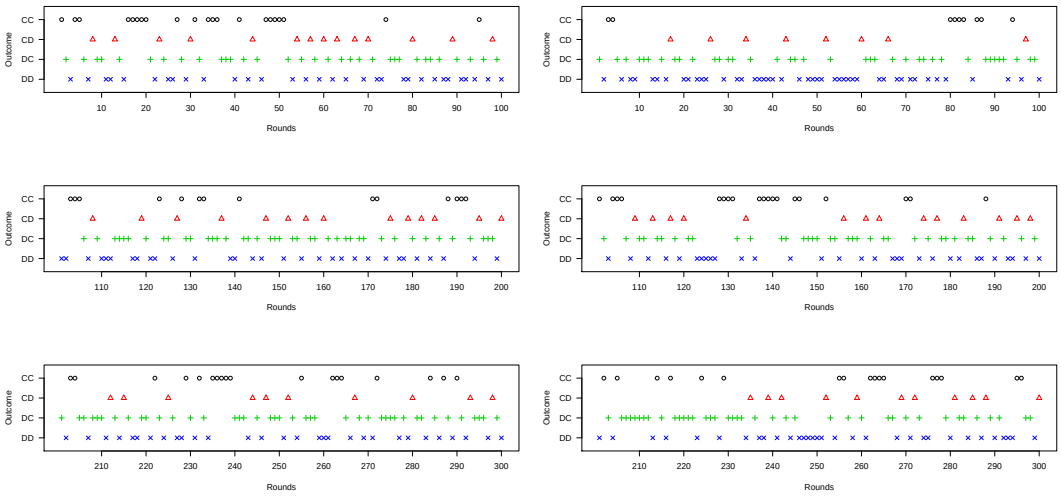
MODEL 1 vs. TFTT



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 1 plays against TFTT strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

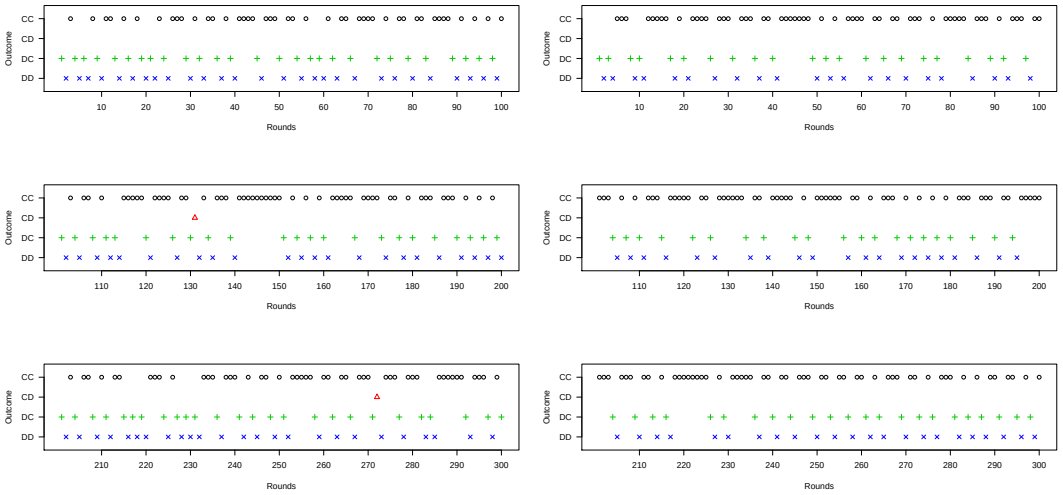
MODEL 1 vs. TFTF



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 1 plays against TFTF strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

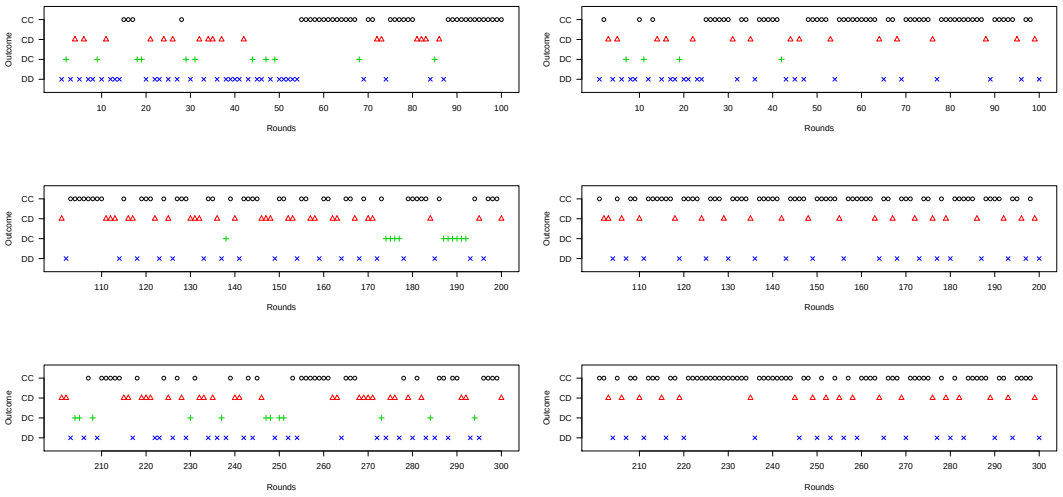
MODEL 1 vs. PAV



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 1 plays against PAV strategy, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

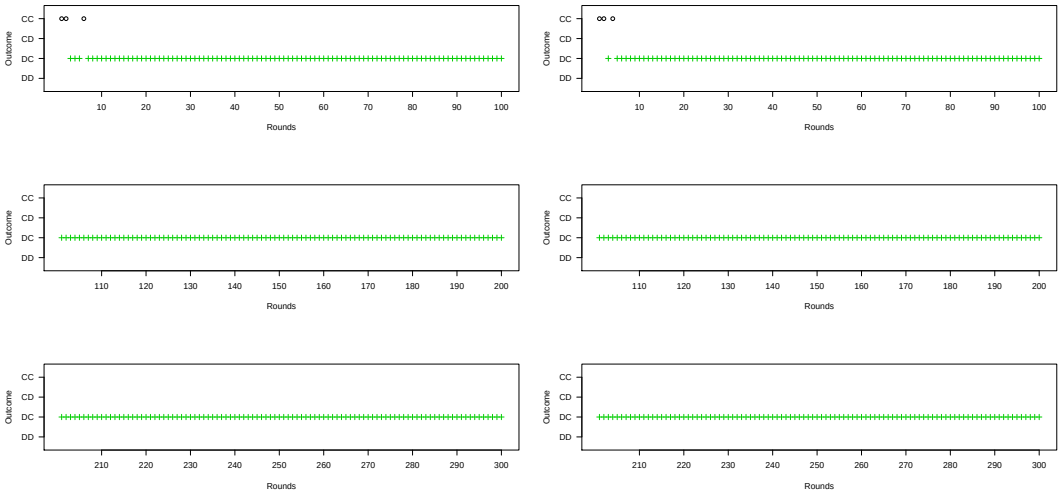
MODEL 1 vs. MODEL 1



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 1 plays against MODEL 1, (a) decay = 0.5, noise = 0.14, L=30, (b) decay = 0.8, noise = 0.14, L=30

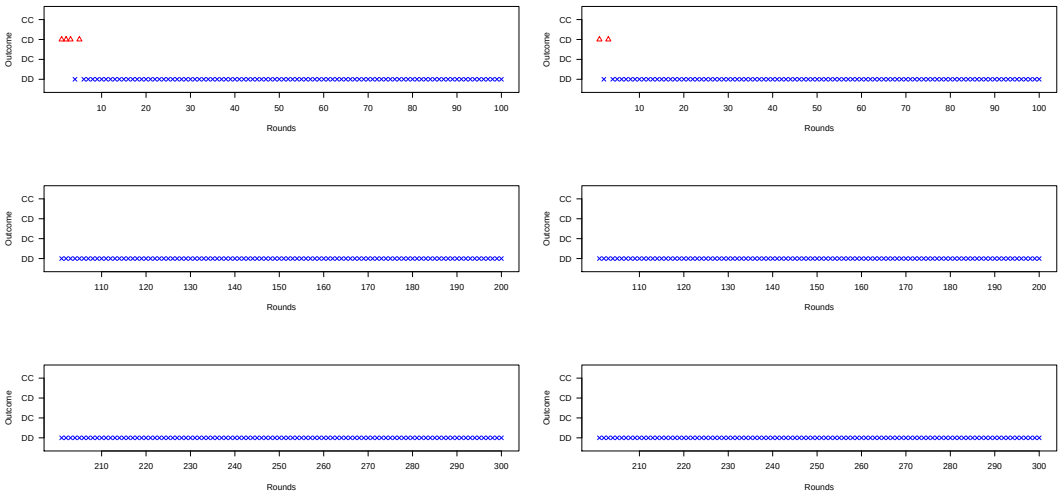
MODEL 2 vs. ALLC



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 2 plays against ALLC strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

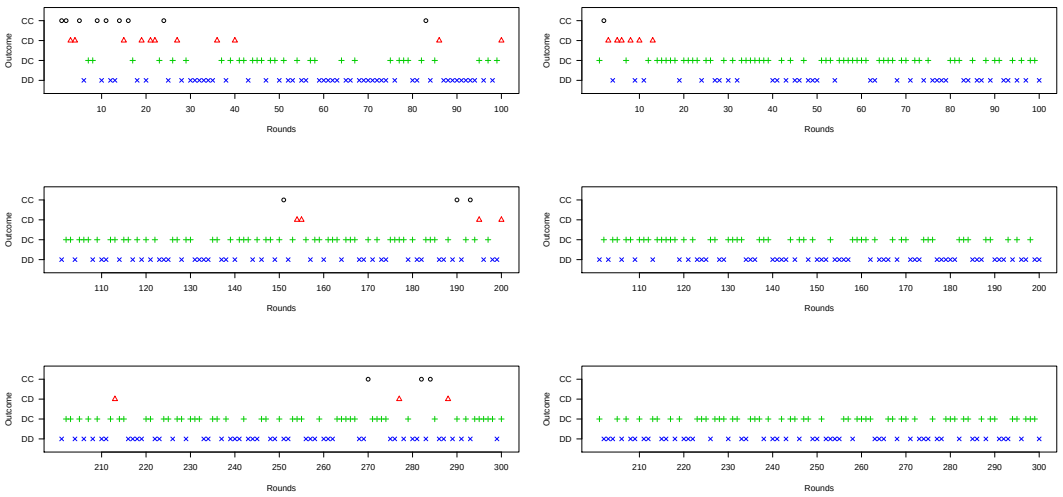
MODEL 2 vs. ALLD



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 2 plays against ALLD strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

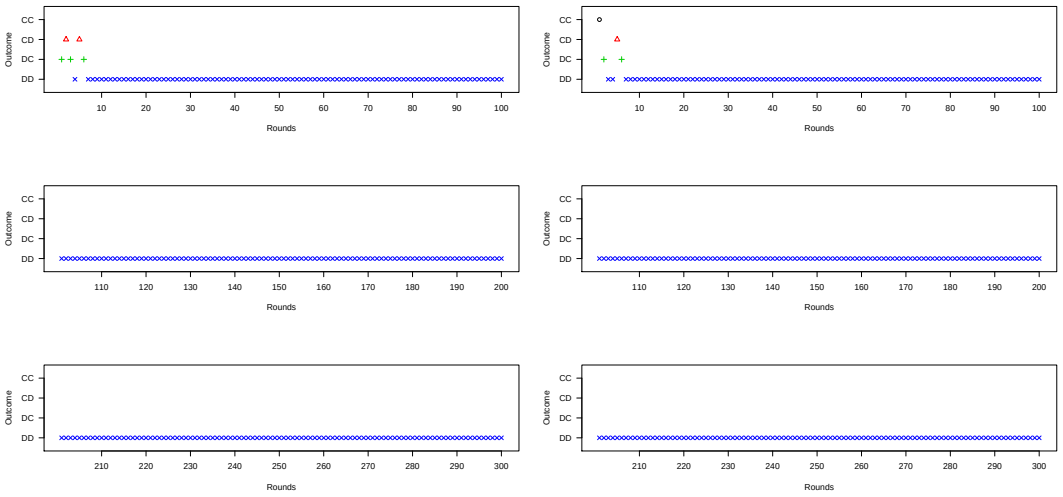
MODEL 2 vs. RAN



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 2 plays against RAN strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

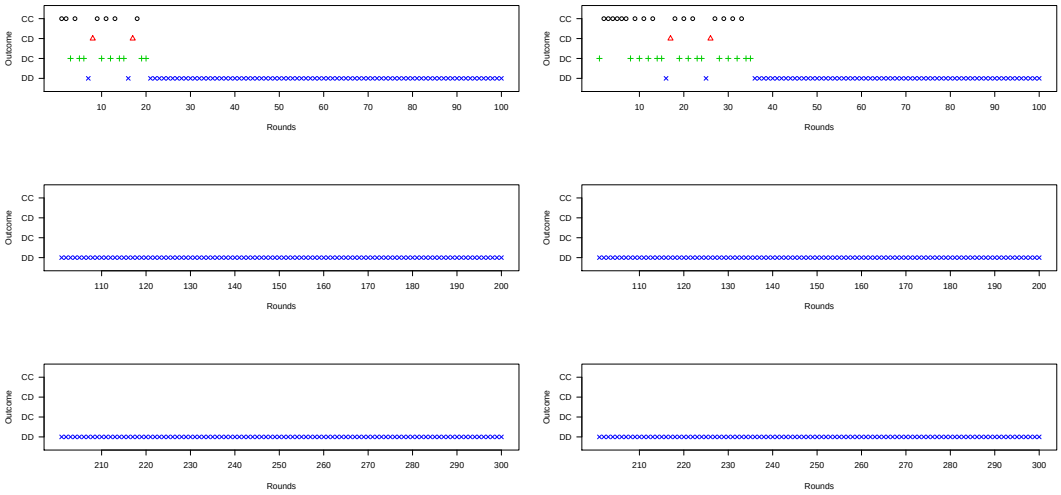
MODEL 2 vs. TFT



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 2 plays against TFT strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

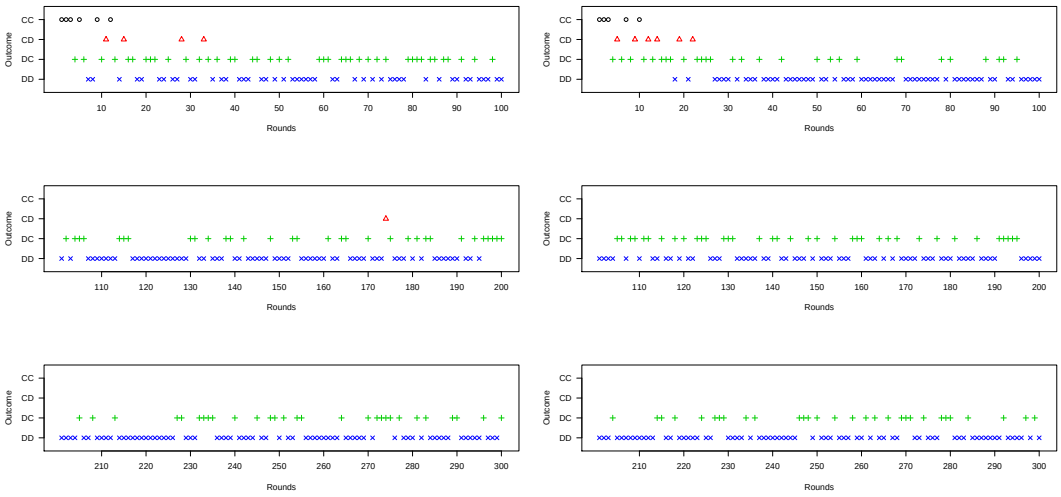
MODEL 2 vs. TFTT



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 2 plays against TFTT strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

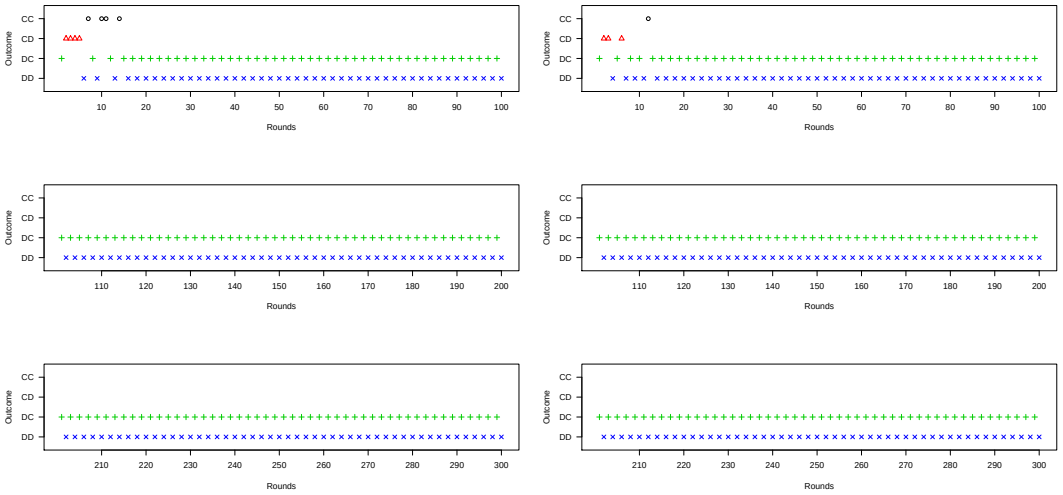
MODEL 2 vs. TFTF



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 2 plays against TFTF strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

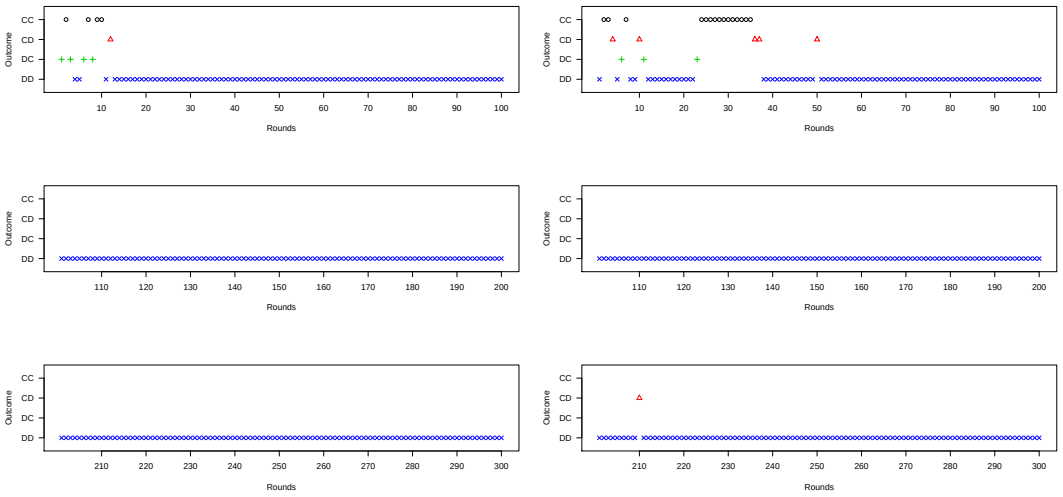
MODEL 2 vs. PAV



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 2 plays against PAV strategy, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

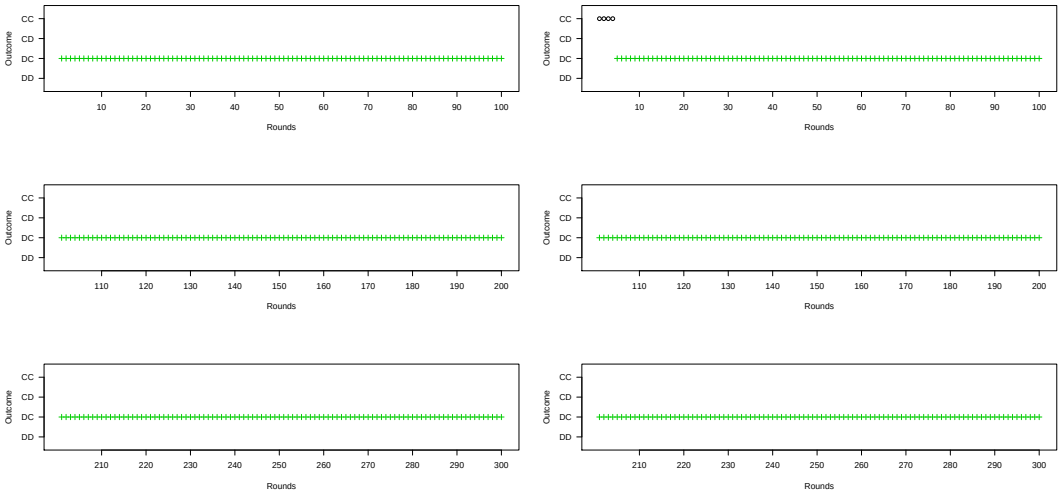
MODEL 2 vs. MODEL 2



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 2 plays against MODEL 2, (a) decay = 0.5, noise = 0.1, L=30, (b) decay = 0.8, noise = 0.1, L=30

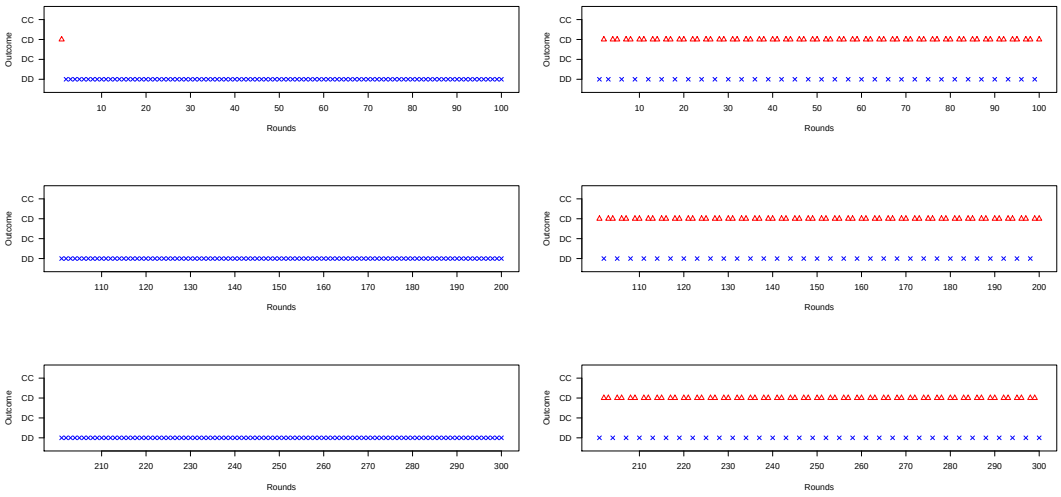
MODEL 3 vs. ALLC



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 3 plays against ALLC strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

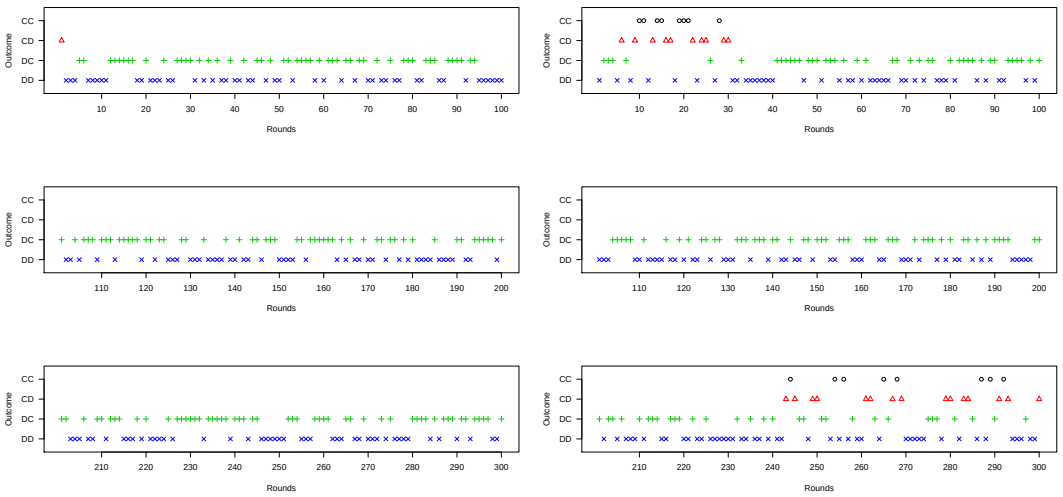
MODEL 3 vs. ALLD



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 3 plays against ALLD strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

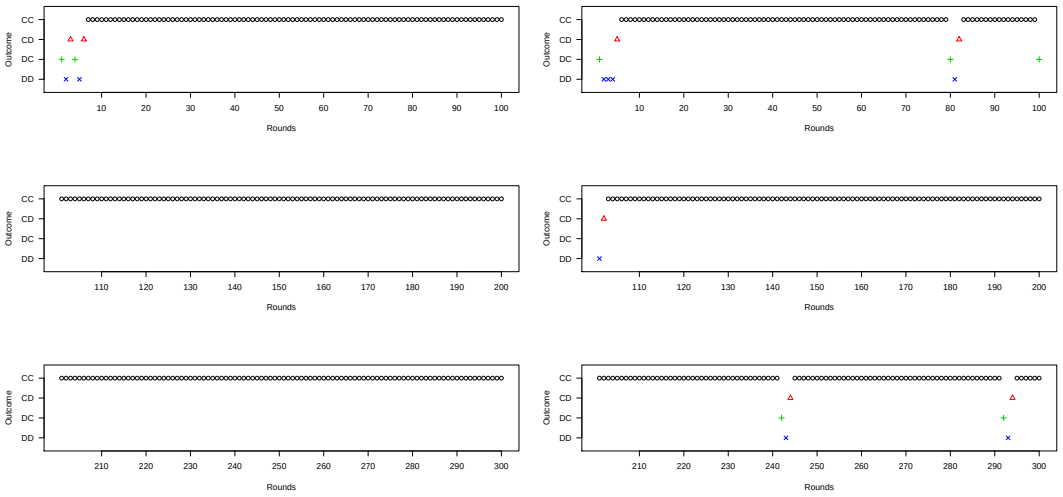
MODEL 3 vs. RAN



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 3 plays against RAN strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

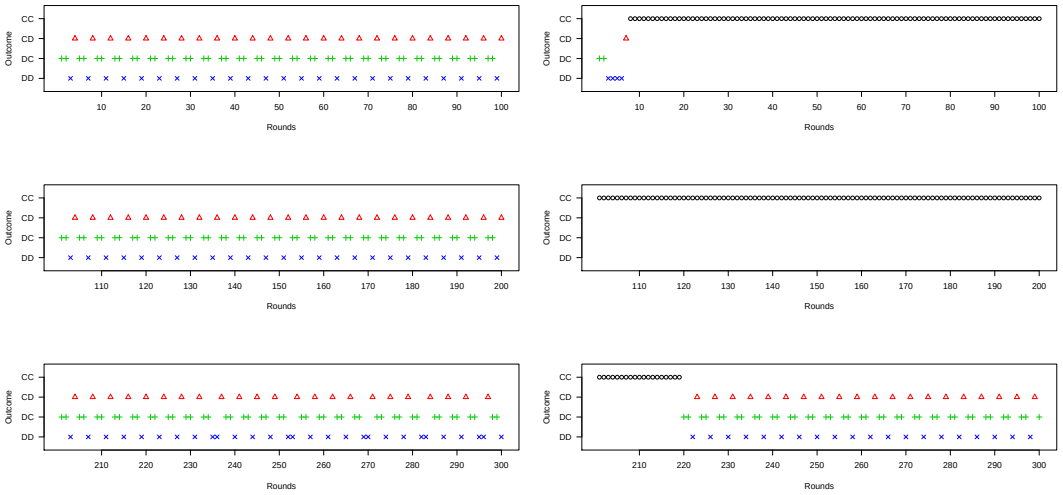
MODEL 3 vs. TFT



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 3 plays against TFT strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

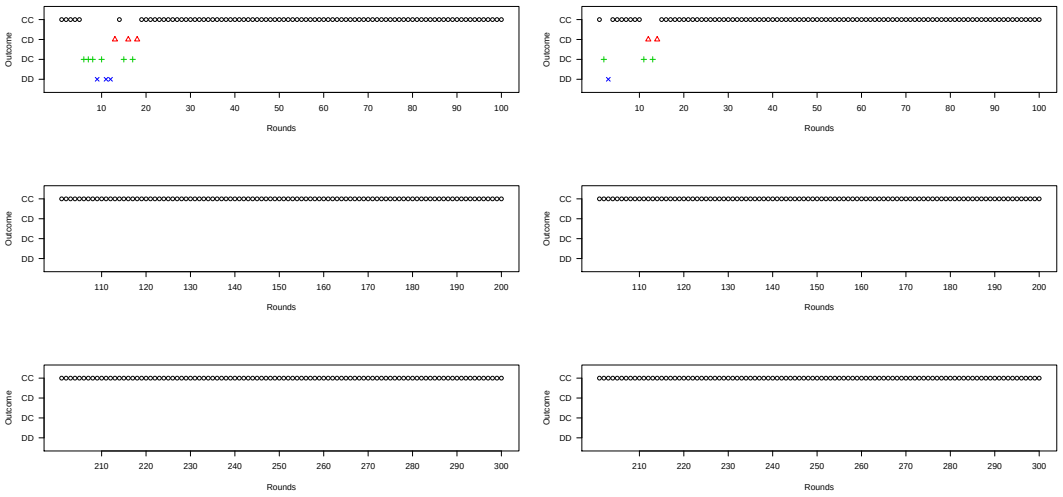
MODEL 3 vs. TFTT



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 3 plays against TFTT strategy,(a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

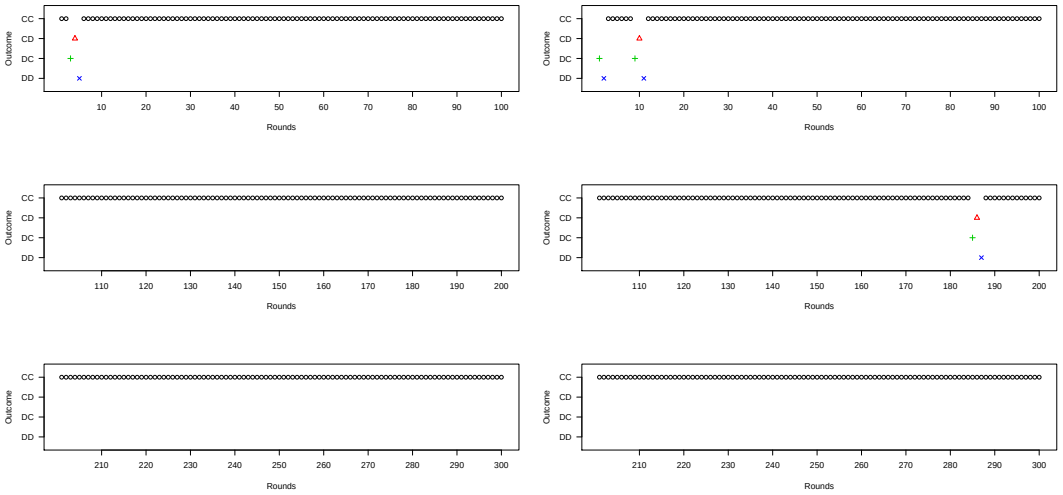
MODEL 3 vs. TFTF



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 3 plays against TFTF strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

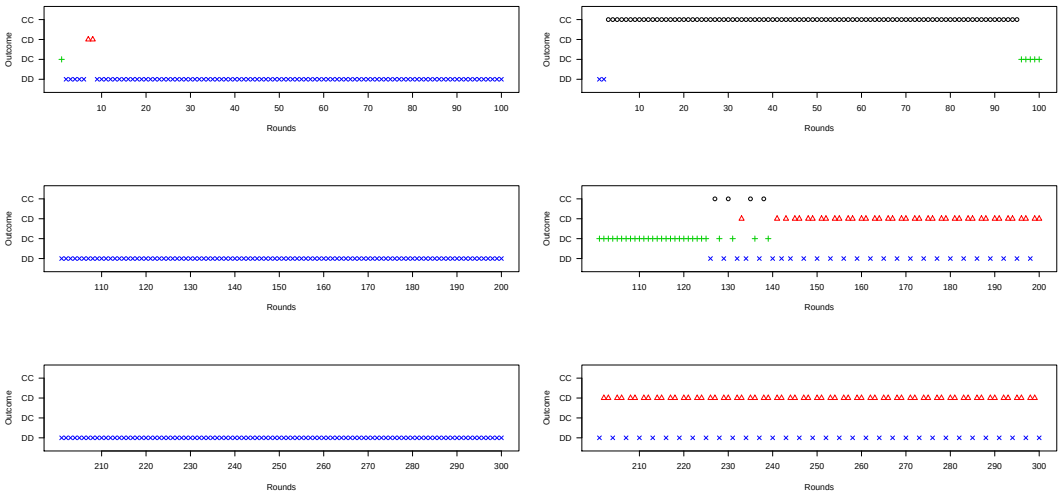
MODEL 3 vs. PAV



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 3 plays against PAV strategy, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

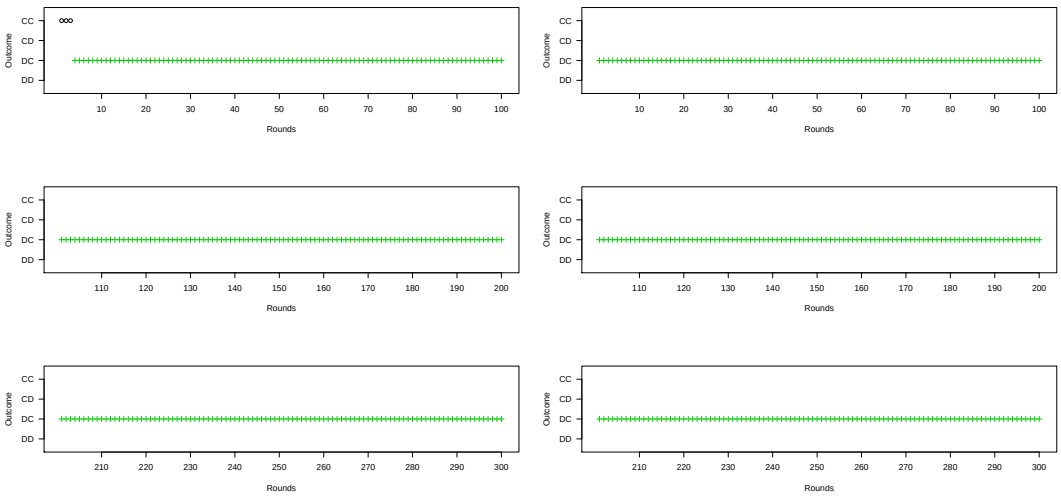
MODEL 3 vs. MODEL 3



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 3 plays against MODEL 1, (a) decay = 0.5, noise = 0.0, L=30, α = 1.5, (b) decay = 0.8, noise = 0.0, L=30, α = 1.5

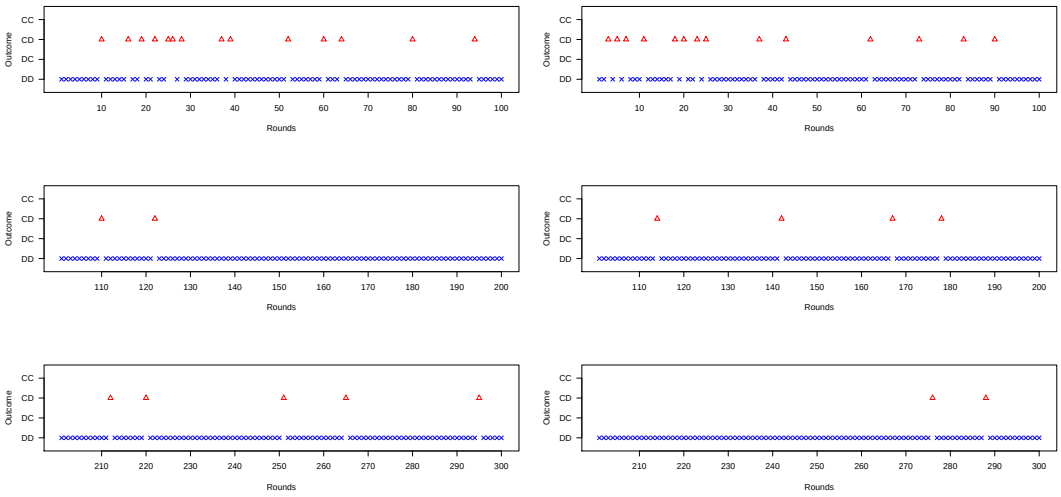
MODEL 4 vs. ALLC



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 4 plays against ALLC strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

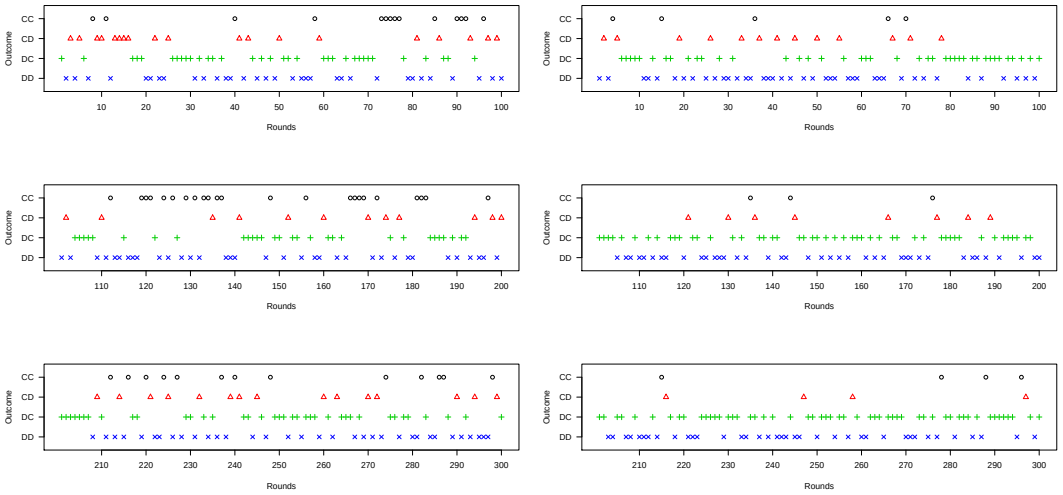
MODEL 4 vs. ALLD



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 4 plays against ALLD strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

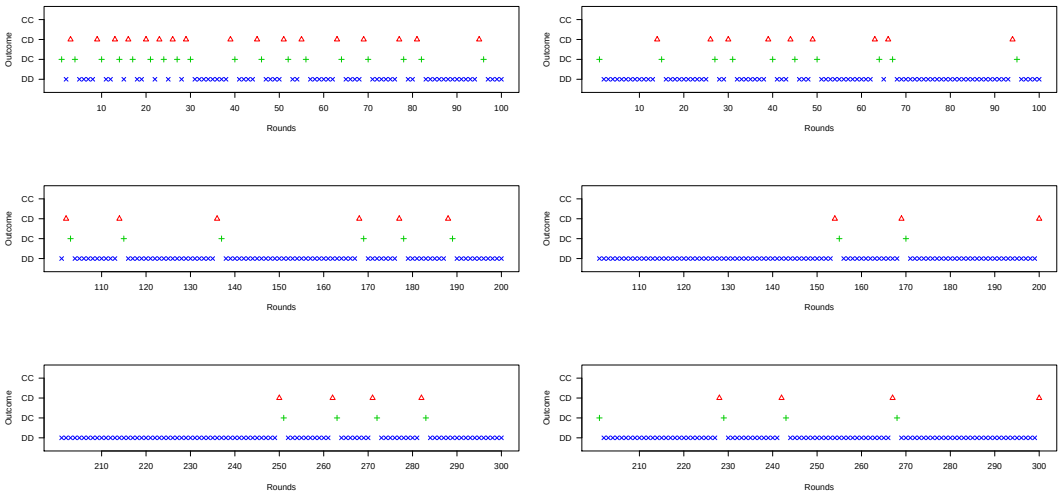
MODEL 4 vs. RAN



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 4 plays against RAN strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

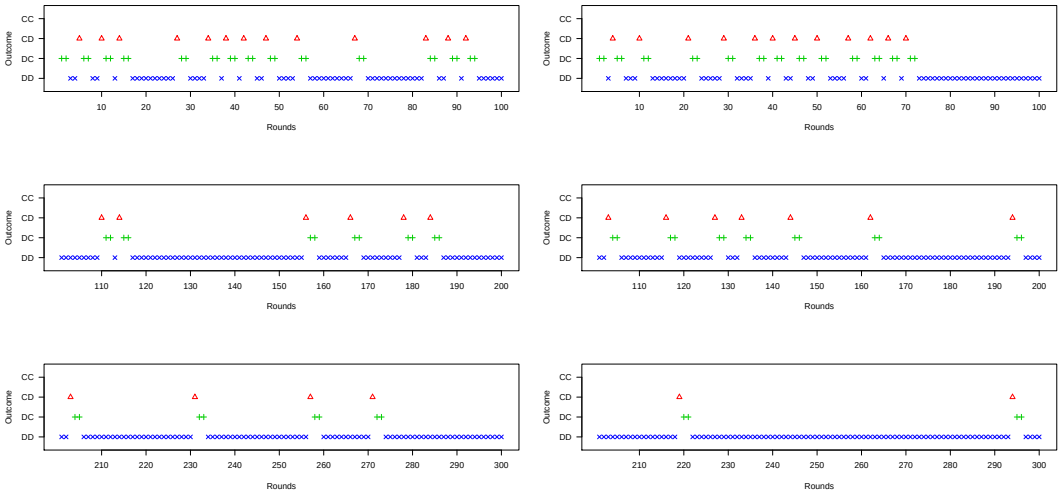
MODEL 4 vs. TFT



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 4 plays against TFT strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

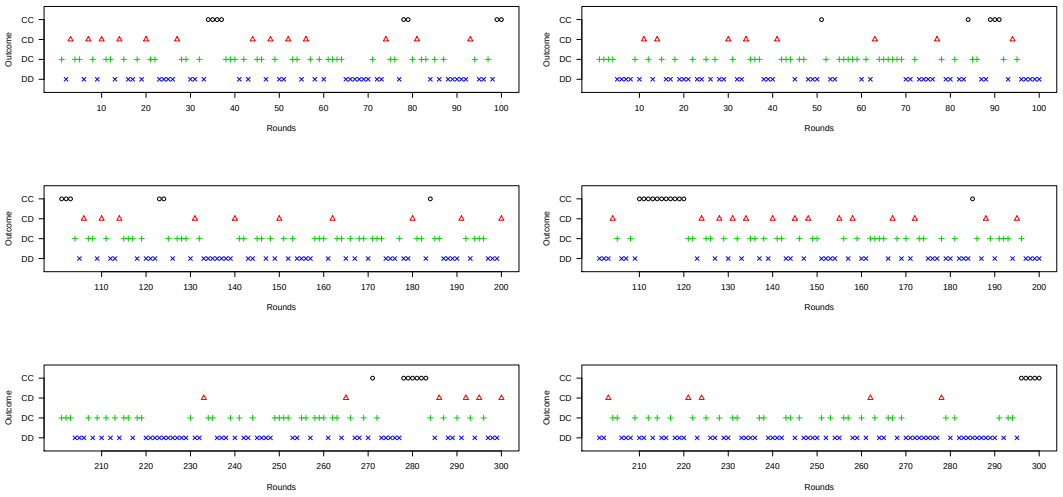
MODEL 4 vs. TFTT



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 4 plays against TFTT strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

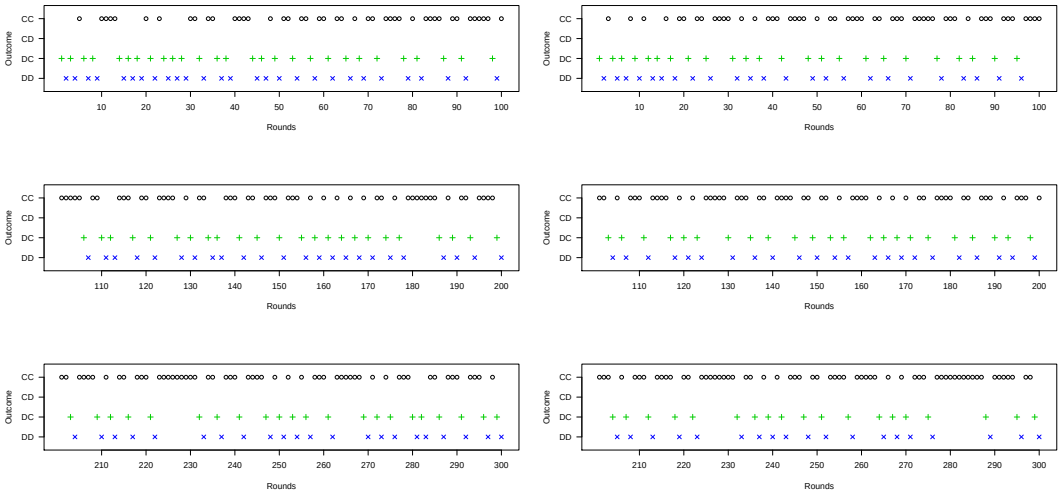
MODEL 4 vs. TFTF



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 4 plays against TFTF strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

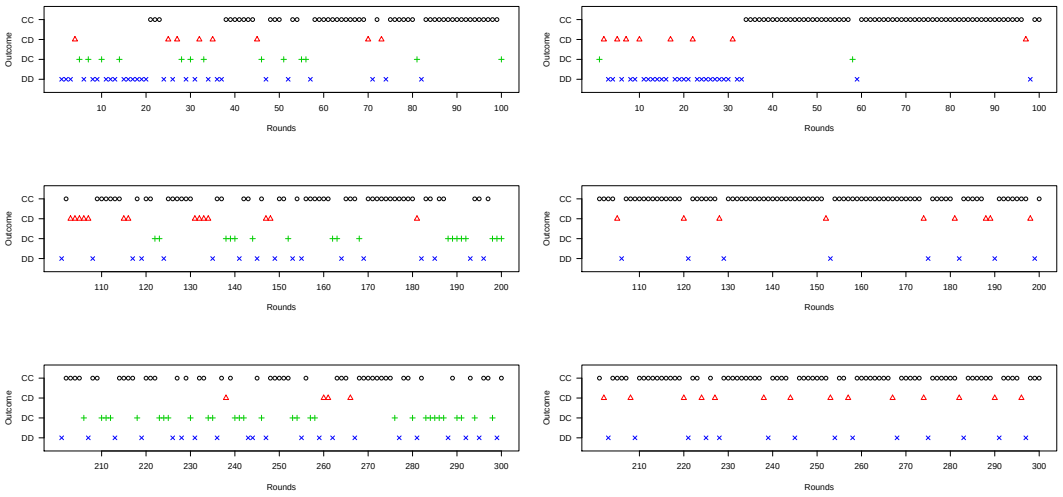
MODEL 4 vs. PAV



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 4 plays against PAV strategy, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

MODEL 4 vs. MODEL 4



(a) (b)

Outcome history of an exemplary run of 300 rounds when MODEL 4 plays against MODEL 4, (a) decay = 0.5, noise = 0.2, L = 30, w = 0.1, (b) decay = 0.8, noise = 0.2, L = 30, w = 0.1

H Simulation Code

```
import psyco
psyco.full()

from ccm.lib.actr import * # allows use of Python ACT-R
import random
import ccm
import sys log=ccm.log()

noiseArg = float(sys.argv[1])
lArg = int(sys.argv[2])

class PD_MOD(ACTR):
    goal=Buffer()
    retrieval=Buffer()
    imaginal=Buffer()

    memory=Memory(retrieval,threshold=None) # to make sure chunks are retrieved even with low activation
    memNoise=DMNoise(memory,noise=noiseArg) #noise variation for the chunks (set to 0 to turn off)
    memBase=DMBaseLevel(memory,decay=0.5) #rate of decay of chunks in memory (set to None to turn off)
    lArgLocal = lArg

    def init():
        for i in range(lArgLocal):
            memory.add('d1-c2 move1:defect1 move2:cooperate2 payoff1:10 payoff2:-10')
            memory.add('d1-d2 move1:defect1 move2:defect2 payoff1:-1 payoff2:-1')
            memory.add('c1-d2 move1:cooperate1 move2:defect2 payoff1:-10 payoff2:10')
            memory.add('c1-c2 move1:cooperate1 move2:cooperate2 payoff1:1 payoff2:1')

    def play(goal='play pd '):
        memory.request('? move1:defect1 move2:? payoff1:? payoff2:?')
        goal.set('payoff p1:None p2:None')

    def predict1(goal='payoff p1:None p2:None',retrieval='? move1:defect1 move2:?m1 payoff1:?x payoff2:?'):
        memory.request('? move1:cooperate1 move2:? payoff1:? payoff2:?')
        goal.modify(p2=x )

    def predict2(goal='payoff p1:None p2:!None ',retrieval='? move1:cooperate1 move2:?m2 payoff1:?y payoff2:?'):
        goal.modify(p1 = y )

    def respond1(goal='payoff p1:!None?m p2:!None?n'):
        if ((float (m )) < (float (n))):
            self.choice='defect'
        else:
```

```
                self.choice='cooperate'
                imaginal.clear()
                goal.clear()

class PD_REA(ACTR): ##general
    goal=Buffer()
    retrieval=Buffer()
    imaginal=Buffer()
    goal2=Buffer()
    goal3=Buffer()
    memory=Memory(retrieval,threshold=None)
    memBase=DMBaseLevel(memory,decay=0.5) #rate of decay of chunks in memory (set to None to turn off)
    memNoise=DMNoise(memory,noise=noiseArg)
    def init():
        for i in range(lArgLocal):
            memory.add('d1-c2 move1:defect1 move2:cooperate2 payoff1:10 payoff2:-10')
            memory.add('d1-d2 move1:defect1 move2:defect2 payoff1:-1 payoff2:-1')
            memory.add('c1-d2 move1:cooperate1 move2:defect2 payoff1:-10 payoff2:10')
            memory.add('c1-c2 move1:cooperate1 move2:cooperate2 payoff1:1 payoff2:1')
        goal2.set('None')
    def remember(goal='play pd ?x ?y ?z ?t', goal2='None'):
        memory.add('pattern ?x ?y ?z ?t')
        memory.request('pattern defect ? ?x ?y')
        imaginal.set('payoff p1:None p2:None ')
        goal2.set('defect')
    def play1(goal='play pd ?x ?y ? ? ', goal2='defect', retrieval= 'pattern defect ?t ? ?'):
        memory.request('? move1:defect1 move2:%s2 payoff1:? payoff2:?'%t)
    def predict1(goal='play pd ?x ?y ', goal2='defect',retrieval='? move1:defect1 move2:?m1 payoff1:?p payoff2:?'):
        memory.request('pattern cooperate ? ?x ?y')
        goal2.set('cooperate')
        goal.set('payoff p1:None p2:?p ')
    def predict2(goal='payoff p1:None p2:!None ', goal2='cooperate', retrieval= 'pattern cooperate ?t ? ?'):
        memory.request('? move1:cooperate1 move2:%s2 payoff1:? payoff2:?'%t)
    def predict3(goal='payoff p1:None p2:!None ',retrieval='? move1:cooperate1 move2:?m2 payoff1:?y payoff2:?'):
        goal.modify(p1 = y )
    def respond1(goal='payoff p1:!None?m p2:!None?n'):
        if ((float (m )) < (float (n))):
            self.choice='defect'
        else:
            self.choice='cooperate'
        imaginal.clear()
        goal.clear()
        goal2.set('None')
    def predictNothing( goal2='defect',memory='error:True'):
```

```
                z=random.uniform(0,2)
                if z < 1:
                    self.choice = 'defect'
                else:
                    self.choice = 'cooperate'
                goal.clear()
                goal2.set('None')
                imaginal.clear()
            def predictNothing1( goal2= 'cooperate',memory='error:True'):
                z=random.uniform(0,2)
                if z < 1:
                    self.choice = 'defect'
                else: self.choice = 'cooperate'
                    goal.clear()
                    goal2.set('None')
                    imaginal.clear()


        class PD_FMOD(ACTR):
            goal=Buffer()
            retrieval=Buffer()
            imaginal=Buffer()
            goal2=Buffer()
            goal3=Buffer()
            memory=Memory(retrieval,threshold=None) # to make sure chunks are retrieved even with low activation
            memNoise=DMNoise(memory,noise=noiseArg) #noise variation for the chunks (set to 0 to turn off)
            memBase=DMBaseLevel(memory,decay=0.5) #rate of decay of chunks in memory (set to None to turn off)
            lArgLocal = lArg
            alphaArgLocal = alphaArg
            def init():
                for i in range(lArgLocal):
                    memory.add('d1-c2 move1:defect1 move2:cooperate2 payoff1:10 payoff2:-10')
                    memory.add('d1-d2 move1:defect1 move2:defect2 payoff1:-1 payoff2:-1')
                    memory.add('c1-d2 move1:cooperate1 move2:defect2 payoff1:-10 payoff2:10')
                    memory.add('c1-c2 move1:cooperate1 move2:cooperate2 payoff1:1 payoff2:1')
                goal2.set('None')
            def play(goal='play pd ?x ?y ?z ?t', goal2='None'):
                memory.add('pattern ?x ?y ?z ?t')
                memory.request('? move1:defect1 move2:? payoff1:? payoff2:?')
                goal2.set('payoff p1:None p2:None p3:None p4:None')
            def predict1(goal2='payoff p1:None p2:None p3:None p4:None',retrieval='?  move1:defect1 move2:?m1 payoff1:?x
payoff2:?'):
                goal2.modify(p2=x )
                goal3.set('1')
                if m1 == 'defect2':
```

```
                memory.request('pattern ? ? defect defect')
            else:
                memory.request('pattern ? ? defect cooperate')
        def predict2(goal2='payoff p1:None p2:!None p3:None p4:None',memory='error:True', goal3='1'):
            memory.request('? move1:? move2:? payoff1:? payoff2:?')
            goal3.set('2')
        def predict3(goal2='payoff p1:None p2:!None p3:None p4:None',goal3='2',retrieval='? move1:? move2:? payoff1:?x payoff2:?'):
            memory.request('? move1:cooperate1 move2:? payoff1:? payoff2:?')
            goal2.modify(p4=x )
            goal3.set('3')
        def predict4(goal2='payoff p1:None p2:!None p3:None p4:None',retrieval= 'pattern ?a ?b ? ?',goal3='1'):
            memory.request('? move1:%s1 move2:%s2 payoff1:? payoff2:?' %(a,b))
        goal3.set('2')
        def predict5(goal2='payoff p1:None p2:!None p3:None p4:!None',retrieval='? move1:cooperate1 move2:?m1 payoff1:?y payoff2:?'):
            if m1 == 'defect2':
                memory.request('pattern ? ? cooperate defect')
            else:
                memory.request('pattern ? ? cooperate cooperate')
            goal2.modify(p1 = y )
        def predict6(goal2='payoff p1:!None p2:!None p3:None p4:!None',goal3= '3' ,memory='error:True'):
            memory.request('? move1:? move2:? payoff1:? payoff2:?')
            goal3.set('4')
        def predict7(goal2='payoff p1:!None p2:!None p3:None p4:!None',goal3= '4',retrieval='? move1:? move2:? payoff1:?x payoff2:?'):
            goal2.modify(p3=x )
            goal3.set('5')
        def predict8(goal2='payoff p1:!None p2:!None p3:None p4:!None',goal3= '3',retrieval= 'pattern ?x ?y ? ?'):
            memory.request('? move1:%s1 move2:%s2 payoff1:? payoff2:?' %(x,y))
        goal3.set('4')
        def respond1(goal2='payoff p1:!None?m p2:!None?n p3:!None?k p4:!None?l'): #a=float(2)
            a = alphaArgLocal
            if ((float (m ) + a * float (k)) < (float (n) + a * float (l))):
                self.choice='defect'
            elif ((float (m ) + a * float (k)) == (float (n) + a * float (l))):
                z=random.uniform(0,2)
                if z < 1:
                    self.choice = 'defect'
                else:
                    self.choice = 'cooperate'
            else:
                self.choice='cooperate'
            goal2.set('None')
```

```
            imaginal.clear()
        goal.clear()


class PD_CMOD(ACTR):
        goal=Buffer()
        retrieval=Buffer()
        imaginal=Buffer()
        goal2=Buffer()
        memory=Memory(retrieval,threshold=None) # to make sure chunks are retrieved even with low activation
        memNoise=DMNoise(memory,noise=noiseArg) #noise variation for the chunks (set to 0 to turn off)
        memBase=DMBaseLevel(memory,decay=0.5) #rate of decay of chunks in memory (set to None to turn off)
        DMAssociateN(memory,imaginal,weight=weightArg)
        lArgLocal = lArg
        def init():
        for i in range(lArgLocal):
            memory.add('d1-d2 move1:defect1 move2:defect2 payoff1:-1 payoff2:-1')
            memory.add('d1-c2 move1:defect1 move2:cooperate2 payoff1:10 payoff2:-10')
            memory.add('c1-c2 move1:cooperate1 move2:cooperate2 payoff1:1 payoff2:1')
            memory.add('c1-d2 move1:cooperate1 move2:defect2 payoff1:-10 payoff2:10')
        goal2.set('None')
        def play(imaginal='play pd ', goal2='None'):
            memory.request('? move1:defect1 move2:? payoff1:? payoff2:?')
            goal2.set('payoff p1:None p2:None')
        def predict1(goal2='payoff p1:None p2:None',retrieval='? move1:defect1 move2:?m1 payoff1:?x payoff2:?'):
            memory.request('? move1:cooperate1 move2:? payoff1:? payoff2:?')
            goal2.modify(p2=x )
        def predict2(goal2='payoff p1:None p2:!None ',retrieval='? move1:cooperate1 move2:?m2 payoff1:?y payoff2:?'):
            goal2.modify(p1 = y )
        def respond1(goal2='payoff p1:!None?m p2:!None?n'):
            if ((float (m )) < (float (n))):
                self.choice='defect'
            else:
                self.choice='cooperate'
            goal2.set('None')
            imaginal.clear()
            goal.clear()


def runGame():
        player1=PD_MOD()
        player2=PD_MOD()
        history1=[None,None,None,None]
        history2=[None,None,None,None] r=[]
        cc=0
        cd=0
```

```
    dc=0
    dd=0
    for i in range(300):
        player1.goal.set('play pd %s %s %s %s'%(history1[-2],history1[-1],history1[-4],history1[-3]))
        player1.imaginal.set('play pd %s %s'%(history1[-2],history1[-1]))
        player1.run()
        player2.goal.set('play pd %s %s %s %s'%(history2[-2],history2[-1],history2[-4],history2[-3]))
        player2.imaginal.set('play pd %s %s'%(history2[-2],history2[-1]))
        player2.run()
        c1=player1.choice
        history1.append(c1)
        c2=player2.choice
        history2.append(c2)
        history2.append(c1)
        history1.append(c2)
        if (c1,c2) ==('cooperate','cooperate'):
            cc+=1
            player1.memory.add('c1-c2 move1:cooperate1 move2:cooperate2 payoff1:1 payoff2:1')
            player2.memory.add('c1-c2 move1:cooperate1 move2:cooperate2 payoff1:1 payoff2:1')
        elif (c1,c2)== ('cooperate','defect'):
            cd+=1
            player1.memory.add('c1-d2 move1:cooperate1 move2:defect2 payoff1:-10 payoff2:10')
            player2.memory.add('d1-c2 move1:defect1 move2:cooperate2 payoff1:10 payoff2:-10')
        elif (c1,c2) == ('defect','cooperate'):
            dc+=1
            player1.memory.add('d1-c2 move1:defect1 move2:cooperate2 payoff1:10 payoff2:-10')
            player2.memory.add('c1-d2 move1:cooperate1 move2:defect2 payoff1:-10 payoff2:10')
        else:
            dd+=1
            player1.memory.add('d1-d2 move1:defect1 move2:defect2 payoff1:-1 payoff2:-1')
            player2.memory.add('d1-d2 move1:defect1 move2:defect2 payoff1:-1 payoff2:-1')
    r=[cc ,cd ,dc ,dd ] #print '%30s'%(r) print str(cc)+','+str(cd)+','+str(dc)+','+str(dd)
    return r

def runExperiment():
    x=1
    for i in range(1000):
    runGame()

runExperiment()
```

References

Anderson, J., Bothell, D., Byrne, M., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, 111(4), 1036–1060.

Andreoni, J. & Miller, J. (1993). Rational cooperation in the finitely repeated prisoner's dilemma: Experimental evidence. *The Economic Journal*, (pp. 570–585).

Axelrod, R. (1984). The evolution of cooperation. *Basic Brook, New York*.

Axelrod, R. & Hamilton, W. (1981). The evolution of cooperation. *Science*, 211(4489), 1390–1396.

Baker, F. & Rachlin, H. (2002). Teaching and learning in a probabilistic prisoner's dilemma. *Behavioural Processes*, 57(2-3), 211–226.

Camerer, C. (2003). Behavioural studies of strategic thinking in games. *Trends in Cognitive Sciences*, 7(5), 225–231.

Kim, S., Taber, C., Mechanism, A., & Framework, S. (2004). A cognitive/affective model of strategic behavior-2-person repeated prisoner's dilemma game. In *Proceedings of the sixth International Conference on Cognitive Modeling* (pp. 360–361).

Langley, P., Laird, J., & Rogers, S. (2009). Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10(2), 141–160.

Lebiere, C., Wallach, D., & West, R. (2000). A Memory-based account of the prisoner's dilemma and other 2x2 games. In *Proceedings of International Conference on Cognitive Modeling* (pp. 185–193).

Lebiere, C. & West, R. (1999). A dynamic ACT-R model of simple games. In *Proceedings of the Twenty-First Annual Conference of the Cognitive Science Society: August 19-21, 1999, Simon Fraser University, Vancouver, British Columbia* (pp. 296).: Lawrence Erlbaum Associates.

Milinski, M. & Wedekind, C. (1998). Working memory constrains human cooperation in the Prisoner's Dilemma.

Rapoport, A., Guyer, M., & Gordon, D. (1976). *The 2 x 2 game*. Univ of Michigan Pr.

Rick, S. & Weber, R. (2008). Meaningful Learning and Transfer of Learning in Games Played Repeatedly Without Feedback.

Rilling, J., Glenn, A., Jairam, M., Pagnoni, G., Goldsmith, D., Elfenbein, H., & Lilienfeld, S. (2007). Neural correlates of social cooperation and non-cooperation as a function of psychopathy. *Biological Psychiatry*, 61(11), 1260–1271.

Rilling, J., Gutman, D., Zeh, T., Pagnoni, G., Berns, G., & Kilts, C. (2002). A neural basis for social cooperation. *Neuron*, 35(2), 395–405.

Stephens, D., McLinn, C., & Stevens, J. (2002). Discounting and reciprocity in an iterated prisoner's dilemma.

Stocco, A., Fum, D., & Zalla, T. (2005). Revising the role of somatic markers in the Gambling Task: A computational account for neuropsychological impairments. In *Proceedings of the 27th Annual Conference of the Cognitive Science Society. Mahwah, NJ: Lawrence Erlbaum Associates.[PDF][info]*.

Sun, R. (2006). *Cognition and multi-agent interaction: From cognitive modeling to social simulation*. Cambridge University Press.

Wedekind, C. & Milinski, M. (1996). Human cooperation in the simultaneous and the alternating Prisoner's Dilemma: Pavlov versus Generous Tit-for-Tat.