ESSAYS ON LEARNING AND LEVEL-K REASONING WITH EVIDENCE FROM EXPERIMENTAL GAMES

by

Peter Wagner

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

Department of Economics

West Lafayette, Indiana

May 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

To my parents and wife, without whose continued support this dissertation would not have been possible.

ACKNOWLEDGMENTS

I am tremendously grateful to my committee members Timothy Cason, Yaroslav Rosokha, David Gill, and Cathy Zhang for their guidance and support throughout my graduate studies. I am also grateful to the Purdue Economics Department for the funding I received and to my fellow graduate students for many helpful discussions of research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In the first chapter of this dissertation, I develop a new model of learning and level-k reasoning in games. My model frames attraction learning in the language of beliefs and extends it to include two important features. The first of these features is an implicit pattern recognition mechanism that learns the importance of contextual information, while the second is a nonlinear probability weighting function with an endogenous fixed point location. The resulting beliefs determine level-1 behavior in a larger level-k rule learning model. In keeping with the literature, I assume that rule learning occurs according to a reinforcement learning mechanism, but I improve the approximation of latent rule reinforcements to simulate the effect of rule exercise. A cognitive foundation for the full model is also provided by implementing it within the ACT-R cognitive architecture.

The second chapter investigates the extent to which human agents use level-k reasoning in repeated mixed strategy games. Towards this end, the Chapter 1 model is estimated using data from a novel experiment. The experiment consisted of two between-subject treatments: in one treatment, the information provided was sufficient to use any level of reasoning, while in the other treatment subjects were only provided with enough information to be level-1. A random effects model is estimated using the data from both treatments to identify the model's belief learning parameters. In the unrestricted treatment, I find that subjects learned to engage almost exclusively in level-1 reasoning. Simulations suggest that this result may be explained by the difficulty of exploiting a player who is level-1.

1. AN ADAPTIVE MODEL OF LEARNING AND LEVEL-K REASONING IN REPEATED NORMAL-FORM GAMES

1.1 Introduction

Level-k reasoning was first introduced by Nagel (1995) and Stahl and Wilson (1994 and 1995) for the purpose of studying players' initial responses to unfamiliar games. Beginning with Stahl (1996), however, a subset of the literature has sought to use level-k reasoning to explain behavior in repeated game environments.[1] The models used in these settings modify the earlier static specifications to incorporate learning dynamics. Stahl (1999) identifies two types of learning that are believed to influence repeated play: rule learning and data learning. Rule learning posits that, rather than adhering to a fixed "level-k type", players learn what levels of reasoning (or what level-k decision rules) to use based on their experiences in the game. In Stahl's specification, this type of learning occurs via a cumulative-style reinforcement learning mechanism. The second type of learning that Stahl identifies, data learning, relates to how agents update their data-driven level-1 beliefs about the actions that will be chosen by other players.[2] Stahl models this type of learning with a distributed-lag forecasting equation using only a single parameter.

Taken as a whole, Stahl (1999, 2000, 2001, and 2003) and Haruvy and Stahl (2002 and 2012) provide ample evidence of both rule learning and data learning across a wide variety of normal-form games, but since the conception of Stahl's model there have been a number of new developments within the adaptive learning literature. First, cumulative-style reinforcement models have largely been supplanted by the more robust averaging-style variations (Ho et al., 2007), as the later allow agents' reward reference points to adjust so that they can distinguish between their most commonly received payoffs. Second, fictitious play learning (Brown, 1951) has emerged as the preeminent modeling framework for backward-looking

[1]↑Level-k reasoning in repeated games: Stahl, 1996, 1999, 2000, 2001, and 2003; Haruvy and Stahl, 2002 and 2012; Danz, Fehr, and Kübler, 2012; Ho and Su, 2013; Gill and Prowse, 2016; Weerd et al., 2018; Feng and Wang, 2019; Ho et al., 2021.

[2]↑Level-1 players are often said to respond to a non-strategic "level-0" player. From this perspective, data learning describes the evolution of the latter's action probabilities. Empirically speaking, very few people are found to fit the level-0 categorization, so I dispense with framing and omit the level-0 type from my model of level-k reasoning.

belief formation. Unlike distributed-lag forecasting, it decreases the rate at which beliefs are learned over time in congruence with the power law of practice. Third, a hybrid type of learning has been developed that combines elements of belief learning and reinforcement learning into a single motion equation (Camerer and Ho, 1999). This hybrid, called "attraction learning", conditions agents' beliefs on their own upcoming action selections. Fourth, pattern recognition has become a well-established feature of adaptive learning in some types of normal-form games (Sonsino and Sirota, 2003; Ruström and Wilcox, 2009; Spiliopoulos, 2012, 2013a, and 2013b). It captures belief volatility by letting agents' beliefs be conditioned on each player's recent action history. Finally, probability weighting has also been identified as an important component of learning in games (Spiliopoulos, 2012 and 2013a), since it allows agents' beliefs to conform to common principles of subjective perception.

As the preceding paragraph suggests, several important features of modern adaptive learning models are not included in Stahl's learning specifications, so I incorporate these features into a more comprehensive model of learning that I develop within the ACT-R cognitive architecture (Anderson and Lebiere, 1998). ACT-R is a general theory of cognition from the psychology literature that has performed successfully in a variety of domains, including sequence learning (Wallach and Lebiere, 2000; Lebiere and Wallach, 2001), dynamic decision making (Fum and Stocco, 2003; Gonzalez et al., 2003), and the playing of economic games.[3] Its procedural framing and jargon-heavy terminology, however, have made it inaccessible to many economists. ACT-R was first introduced to the economics literature through a data learning model developed by Spiliopoulos (2013a). Like the model developed in this chapter, it was used to study behavior in repeated normal-form games. On the two datasets Spiliopoulos analyzed, his model outperformed a popular generalization of fictitious play learning known as weighted fictitious play (Cheung and Friedman, 1997), but in spite of this finding there have been no further attempts to integrate ACT-R into the economics literature.

[3]↑ACT-R models of game-playing: Ritter and Wallach, 1998; Lebiere and West, 1999; Sanner et al., 2000; Kim and Taber, 2004; West et al., 2005; Reitteret al., 2010; Spiliopoulos, 2013a and 2013b, Thomson et al., 2014.

In an effort to improve the architecture's accessibility, the data learning model that I develop in this chapter is framed as an extension of weighted fictitious play learning. I show that ACT-R is able to nest weighted fictitious play beliefs as a special case given the right set of modeling assumptions. This finding is important as it relates two disjointed literatures and provides a cognitive foundation for fictitious play models. Under standard fictitious play learning, beliefs are formed by taking a simple average over all previously observed action distributions. Weighted fictitious play learning takes a time weighted average instead such that more weight is given to newer observations. Older observations are discounted to capture human forgetting and facilitate faster belief adaptation. This framework can be extended to include beliefs that are behaviorally equivalent to frequency-style reinforcement and attraction learning models. For my first major departure from Spiliopoulos, I accommodate these beliefs within the ACT-R cognitive architecture.[4]

My data learning model builds on this base-level learning in two important ways. First, it extends belief formation to include an implicit pattern recognition mechanism. In the style of classical conditioning, my model induces pattern recognition through a series of learned action associations, where associations are formed between tuples of the opponents' chosen actions and each player's action choice in the previous period. The more pairings of these actions that an agent observes, the stronger their perceived association. Beliefs are then increasing in the strength of each action tuple's association with the actions chosen last period. In contrast to the similarity matching featured in Spiliopoulos (2013a and 2013b), this model of pattern recognition allows agents to learn the importance of contextual information. By using an implicit method of learning it also consumes less working memory, making an n-player generalization of the model cognitively plausible.

My model's second extension to base-level learning introduces a nonlinear probability weighting function. In ACT-R, probability weighting naturally emerges from a memory retrieval process called "blending". Blending returns a weighted average of any desired set

[4]↑It could be argued that reinforcement learning is too unsophisticated to describe the behavior of level-1 players, but when we consider that reinforcement learning can be framed as belief learning, its hidden sophistication begins to become obvious. Under the belief learning framing, reinforcement learners rationally respond to their expectations just like the players in any other level-1 model. If rationality is considered to be a sign of sophistication, reinforcement learners are sophisticated players.

of values that are stored in declarative memory. When the values being averaged are the outcomes of a repeated random event, blending behaves like a probability weighting function. It therefore determines agents' sensitivities to probabilities as well as their levels of optimism. For my last major departure from Spiliopoulos, I weight the stored stage-game outcomes in memory using a generalization of ACT-R's standard blending function. This endogenizes the weighting function's fixed point location and makes the nesting of weighted fictitious play learning possible.

The data learning model maps agents' observed action histories into their level-1 beliefs. From there the behavior of each level is determined by a series of best response mappings. A player using level-1 reasoning, for instance, noisily best responds to his level-1 beliefs. A player using level-2 reasoning noisily best responds to his opponents' best responses to their level-1 beliefs. This process of iterating on the preceding levels' best responses may continue indefinitely, though experimental evidence suggests that it is rare for people to reason beyond level-3 (Crawford et al., 2013). Following Stahl (1999), I model players as boundedly rational agents who decide their actions using level-k rules. Each rule applies a different level of reasoning to determine an agent's action: the level-1 rule applies level-1 reasoning, the level-2 rule applies level-2 reasoning, and so forth.[5] In contrast to Camerer et al. (2004)'s closely related cognitive hierarchy model, I make no assumptions about the proportion of players that use each level of reasoning.

In Stahl's model, agents approximate their opponents' level-1 beliefs with their own level-1 beliefs when using the level-2 rule. Unfortunately, this approximation assumes two features of the environment that severely limit its scope of application. First, it assumes that all players share the same payoff functions, precluding its use in asymmetric normal-form games. Second, it assumes a large population of players so that each agent's influence on the empirical action distribution is negligible. To keep the setting I consider unrestricted by these assumptions, I generalize this aspect of Stahl's model. In contrast to Stahl, I assume that agents mirror their own belief updating processes to approximate the belief updating

[5]↑Camerer et al. (2002) specifies a "sophisticated" attraction learner that is similar to Stahl (1999)'s level-2 player, but instead of best responding to the unsophisticated type, he best responds to his beliefs about the distribution of players. In contrast to Stahl's model, there is also no mechanism for learning a different level of sophistication.

processes of other players. Any agent's beliefs can then be inferred directly from the observed action history.

At the start of every period, agents noisily select their most preferred level-k rules. Preferences over rules are learned through experience via ACT-R's averaging-style reinforcement learning mechanism. As a result, agents become more likely to choose rules that have rewarded them with favorable payoffs. If rule selection is observed by the researcher, an agent's chosen rules can be reinforced by their resulting stage-game payoffs, but if rule selection is latent then the reinforcements must be approximated to maintain the tractability of the model. In the latent-rule setting, Stahl reinforces each rule at the end of every period by its expected payoff in that period given the opponents' chosen actions, but while this approximation preserves the learning principle known as the law of effect, it neglects its sister principle, the law of exercise. The law of exercise permits learning through response repetition as opposed to learning from the magnitude of a response's rewards. To retain this type of learning, I propose a new approximation for the latent rule reinforcements that is shown to be superior to Stahl's. My model makes a rule's reinforcement proportional to the probability that the rule was selected to simulate the effect of rule exercise.

The rest of the chapter is organized as follows. Section 1.2 provides an overview of the model's state-to-action mappings and describes how agents learn their preferences over these mappings' corresponding level-k rules. Section 1.3 develops the data learning model that determines agents' level-1 beliefs. Appendix A.1 presents the ACT-R framing of the model and Section 1.4 concludes.

1.2 Level-k rules

Consider a generic normal-form game $G \equiv (N, A, P)$, where $N \equiv \{1, ..., n\}$ is the set of players, $A \equiv \{A_1, ..., A_n\}$ is the set of action spaces, and $P \equiv \{P_1, ..., P_n\}$ is the set of payoff functions. Define $h^t \equiv (\{a_1^t, ..., a_n^t\})_{t=1}^{t-1}$ as all players' joint action history from period 1 up to period $t \in \mathbb{N}^+$ and $I^t \equiv (G, h^t)$ as the information that is available to them after they experience this history.

1.2.1 Rule mappings

Following Stahl (1999)'s general theory of rule learning, I model players as boundedly rational agents who decide their actions using behavioral rules. Understand a behavioral rule to be any mapping for a generic player $\mathrm{i} \in N$ from the information state space I^* to the set of general measures on his actions $M(A_\mathrm{i})$. The specific family of rules that I consider in this chapter is derived by using level-k reasoning. To state these rules formally, let $B_\mathrm{i} : M(\prod_{\mathrm{i}' \in N} A_{\mathrm{i}'}) \to M(A_\mathrm{i})$ denote player i's best response function and $b_\mathrm{i} : I^* \to M(\prod_{\mathrm{i}' \in N} A_{\mathrm{i}'})$ be the mapping from his information to his level-1 beliefs. Also define $b_\mathrm{i}^t \equiv b_\mathrm{i}(I^t)$ as the output of this mapping when the input is information I^t. Then in period t, player i's level-k rules can be expressed as follows:

$$\begin{aligned}
&\textbf{Level-1: } B_\mathrm{i}(b_\mathrm{i}^t) \\
&\textbf{Level-2: } B_\mathrm{i}(\{B_\mathrm{j}(b_\mathrm{j}^t)\}_{\mathrm{j} \neq \mathrm{i} \in N}) \\
&\textbf{Level-3: } B_\mathrm{i}(\{B_\mathrm{j}(\{B_{\mathrm{i}'}(b_{\mathrm{i}'})\}_{\mathrm{i}' \neq \mathrm{j} \in N})\}_{\mathrm{j} \neq \mathrm{i} \in N}) \\
&\vdots
\end{aligned}$$

Thus, an agent using the level-1 rule (rule 1) best responds to his level-1 beliefs, an agent using the level-2 rule (rule 2) best responds to his opponents' best responses to their level-1 beliefs, and so forth. In keeping with Stahl, I assume that B_i only admits uniform random responses over all of the actions between which player i is indifferent. This ensures that B_i maps to a single general measure for any given set of beliefs. In contrast to Stahl, however, I assume that agents mirror their own belief updating processes to approximate the belief updating processes of other players. Any agent's beliefs can then be inferred directly from the observed action history. The mapping between action histories and beliefs is specified in the following section.

At the start of every period, agents noisily select their most preferred level-k rules. Preferences over rules are determined by the agents' learned utility values. Each utility reflects a player's past experience and success using a particular level of reasoning. To fix ideas, assume rule selection is governed by a simple logit choice function. Let $U_\mathrm{i}^t(k)$ denote player i's learned utility for rule $k \in \{1, ..., r\}$ as of period t and $\kappa \in \mathbb{R}^+$ be a utility sensitivity parameter. Also allow $\upsilon(k) \in \mathbb{R}$ to be any time-invariant bias that agents may have for choosing the kth level of reasoning. Then conditional on information I^t, player i's probability of selecting rule k is given by

$$p_{\mathrm{i}}^t(k) \equiv \frac{\exp(\upsilon(k) + \kappa \cdot U_{\mathrm{i}}^t(k))}{\sum_{k'=1}^{r} \exp(\upsilon(k') + \kappa \cdot U_{\mathrm{i}}^t(k'))} \tag{1.1}$$

Once his rule is selected, player i uses the rule's iterated best responses to map the specified set of level-1 beliefs into the appropriate action. Action selection is also noisy, however, and it is again described by a logit choice function. Define

$$V_{\mathrm{i}}^t(a_{\mathrm{i}}|k) \equiv \begin{cases} \sum_{a_{-\mathrm{i}} \in A_{-\mathrm{i}}} P_{\mathrm{i}}(a_{\mathrm{i}}, a_{-\mathrm{i}}) \cdot b_{\mathrm{i}}^t(a_{-\mathrm{i}}|a_{\mathrm{i}}) & \text{if } k = 1 \\ \sum_{a_{-\mathrm{i}} \in A_{-\mathrm{i}}} P_{\mathrm{i}}(a_{\mathrm{i}}, a_{-\mathrm{i}}) \cdot \prod_{a_{\mathrm{j}} \in a_{-\mathrm{i}}} B_{\mathrm{j}}(a_{\mathrm{j}}|b_{\mathrm{j}}^t) & \text{if } k = 2 \\ \sum_{a_{-\mathrm{i}} \in A_{-\mathrm{i}}} P_{\mathrm{i}}(a_{\mathrm{i}}, a_{-\mathrm{i}}) \cdot \prod_{a_{\mathrm{j}} \in a_{-\mathrm{i}}} B_{\mathrm{j}}(a_{\mathrm{j}}|\{B_{\mathrm{i}'}(b_{\mathrm{i}'}^t)\}_{\mathrm{i}' \neq \mathrm{j} \in N}) & \text{if } k = 3 \\ \vdots \end{cases} \tag{1.2}$$

as player i's expected value for action $a_{\mathrm{i}} \in A_{\mathrm{i}}$ given his level-k beliefs in period t, where $A_{-\mathrm{i}} \equiv \prod_{\mathrm{j} \neq \mathrm{i} \in N} A_{\mathrm{j}}$ is the Cartesian product of all other players' action spaces. Let $\nu(a_{\mathrm{i}}) \in \mathbb{R}$ denote any time-invariant bias that agents may have for choosing action a_{i} and $\lambda \in \mathbb{R}^+$ be a payoff sensitivity parameter. Then conditional on rule k being chosen and information I^t, player i's probability of selecting action a_{i} is

$$p_{\mathrm{i}}^t(a_{\mathrm{i}}|k) \equiv \frac{\exp(\nu(a_{\mathrm{i}}) + \lambda \cdot V_{\mathrm{i}}^t(a_{\mathrm{i}}|k))}{\sum_{a'_{\mathrm{i}} \in A_{\mathrm{i}}} \exp(\nu(a'_{\mathrm{i}}) + \lambda \cdot V_{\mathrm{i}}^t(a'_{\mathrm{i}}|k))} \tag{1.3}$$

1.2.2 Rule learning

After selecting his action, player i observes the actions chosen by his opponents and receives stage game payoff $P_{\mathrm{i}}(a_{\mathrm{i}}^t, a_{-\mathrm{i}}^t)$. His level-k rules can then be reinforced in a number of different ways depending on the nature of the environment. If rule selection is observed by the researcher, an agent's chosen rules can be reinforced by their resulting stage-game payoffs, but if rule selection is latent then the reinforcements must be approximated to maintain the tractability of the model. In the latent-rule setting, Stahl reinforces each rule at the end of every period by its expected payoff in that period given the opponents' chosen actions. My model instead makes a rule's reinforcement proportional to the probability that the rule was selected. Towards this end, let

$$p_{\mathrm{i}}^{t}(k|a_{\mathrm{i}}^{t}) = \frac{p_{\mathrm{i}}^{t}(k) \cdot p_{\mathrm{i}}^{t}(a_{\mathrm{i}}^{t}|k)}{\sum_{k'}^{r} p_{\mathrm{i}}^{t}(k') \cdot p_{\mathrm{i}}^{t}(a_{\mathrm{i}}^{t}|k')} \tag{1.4}$$

denote the probability that player i selected rule k given his action choice and information I^t. Also define $\pi(k) \equiv U_{\mathrm{i}}^{1}(k)$ as rule k's initial utility and allow $\alpha \in [0,1]$ to be a step size parameter. Then the utility of rule k can be said to evolve according to the following difference learning equation:

$$U_{\mathrm{i}}^{t+1}(k) = U_{\mathrm{i}}^{t}(k) + \alpha \cdot p_{\mathrm{i}}^{t}(k|a_{\mathrm{i}}^{t}) \cdot (P_{\mathrm{i}}(a_{\mathrm{i}}^{t}, a_{-\mathrm{i}}^{t}) - U_{\mathrm{i}}^{t}(k)) \tag{1.5}$$

In the context of (1.5), $U_{\mathrm{i}}^{t}(k)$ can be interpreted as rule k's predicted reinforcement and $P_{\mathrm{i}}(a_{\mathrm{i}}^{t}, a_{-\mathrm{i}}^{t}) - U_{\mathrm{i}}^{t}(k)$ as player i's prediction error. Learning adjusts $U_{\mathrm{i}}^{t+1}(k)$ in the direction of $P_{\mathrm{i}}(a_{\mathrm{i}}^{t}, a_{-\mathrm{i}}^{t})$ but only by a fraction of the total prediction error. This fraction, $\alpha \cdot p_{\mathrm{i}}^{t}(k|a_{\mathrm{i}}^{t})$, can be easily comprehended by considering its constituent parts. The first part, α, determines the player's rate of utility learning, with higher (lower) values of α corresponding to faster (slower) rates of updating. The second part, $p_{\mathrm{i}}^{t}(k|a_{\mathrm{i}}^{t})$, determines what share of the total reinforcement rule k receives in period t. It optimally estimates the indicator for whether rule k was selected assuming a quadratic loss function. Together, equations (1.1) and (1.5) imply that a rule's selection probability is increasing in the likelihood that its use has yielded favorable payoffs.

In order to assess the quality of my approximation for the latent rule reinforcements as well as compare it to Stahl's, I simulated 100,000 supergames of matching pennies under three different sets of parameters with each of the three ways that the reinforcements can be specified. In all of these games, I pit a level-1/level-2 rule learning agent against a level-1 player. The former's rule selection probabilities were recorded every period so that their evolution could be compared across paradigms. For the sake of simplicity, both agents formed their beliefs via standard fictitious play learning. They also selected their actions noiselessly so that there was the greatest incentive to learn the optimal level of reasoning. The rule learning parameters for the approximations were chosen to match the observable-rule model's simulated rule selection probabilities as closely as possible.

Figure 1.1 depicts the results of this moment matching procedure by graphing the simulated rule selection probabilities for each model. By looking at the starting slopes of the target time series, we can infer which of the following two learning principles is initially more dominant: the law of effect or the law of exercise. In our simulated setting, the law of effect increases the probability that the level-2 rule is selected since the level-1 opponent makes level-2 reasoning optimal, while the law of exercise decreases this probability because our chosen parameters imply that the level-1 rule is initially more probable. When the starting rule utilities are similar to the stage-game's typical payoffs, the law of effect is initially more dominant. This can be seen by the target model's upward sloping time series in the first graph of the figure. Conversely, when the starting rule utilities are smaller than the stage-game's typical payoffs, the law of exercise is initially more dominant. This is demonstrated by the downward sloping portions of the target model's time series in the second and third graphs of the figure. Comparing Stahl's approximation to these time series, we see that it provides a good fit when the law of effect is initially more dominant, but when the law of exercise is stronger it is unable to emulate the resulting decrease in level-2 reasoning. My approximation, on the other hand, fits the target time series well regardless of which learning principle is more dominant, since it allows for learning by the law of effect as well as the law of exercise.

1.3 Level-1 beliefs

Level-k reasoning is founded on agents' level-1 beliefs, as from there the behavior of each level is determined by a series of best response mappings. A good belief specification is therefore essential for the success of any level-k model. In one-shot games, it is usually sufficient to assume that players have uniform prior beliefs, but in repeated game settings the opportunities for learning necessitate that we also specify a belief updating process. For ease of exposition, the data learning model that I develop in this section is framed as an

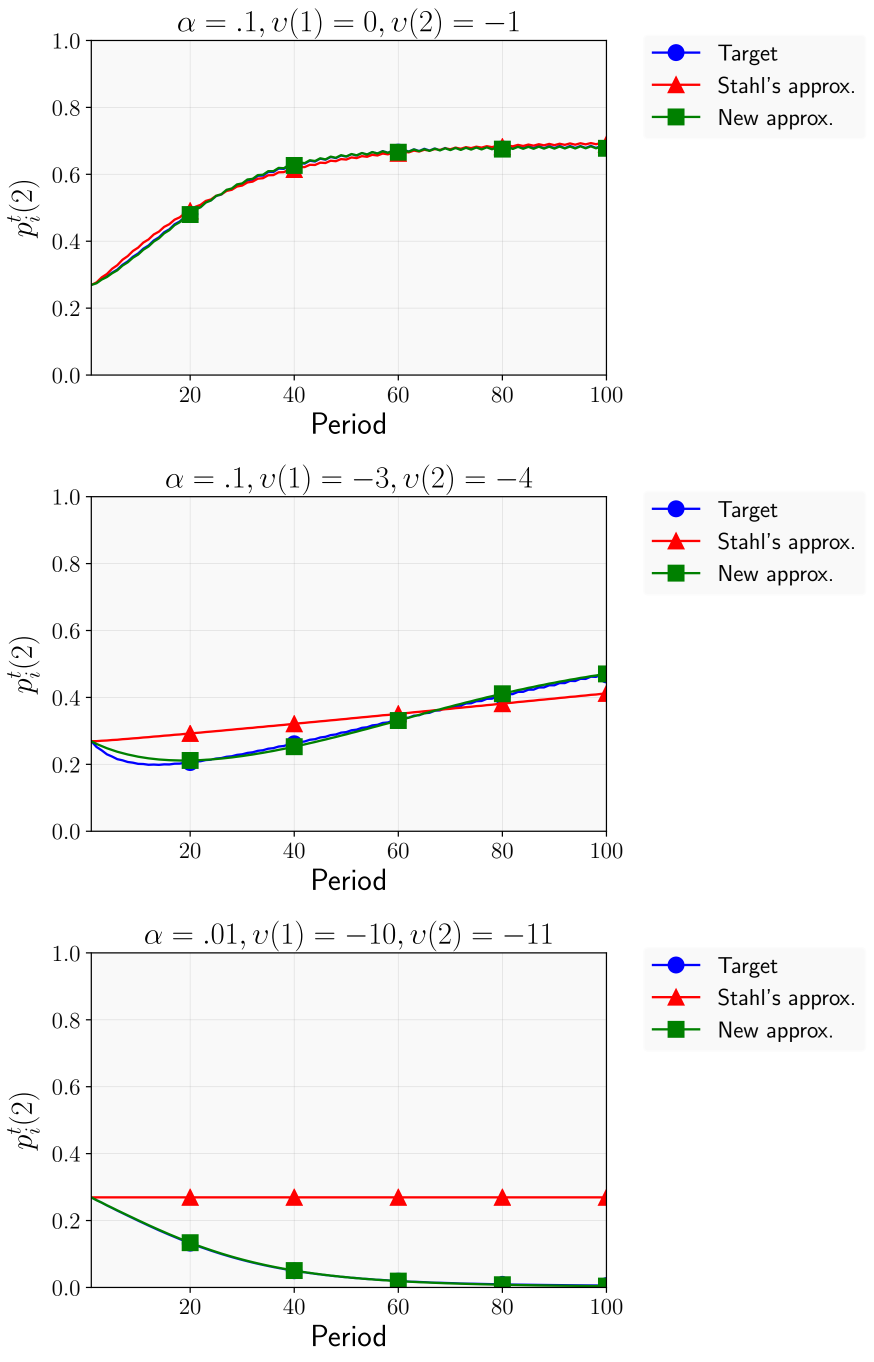


Figure 1.1. Simulated rule selection probabilities, periods 1-100

extension of weighted fictitious play learning. Appendix A.1 provides the original framing of the model as it would appear in the ACT-R architecture.

1.3.1 Base-level learning

Under standard fictitious play learning, beliefs are formed by taking a simple average over all previously observed action distributions. Weighted fictitious play learning takes a time weighted average instead such that more weight is given to newer observations. Older observations are discounted to capture human forgetting and facilitate faster belief adaptation. While memories decay exponentially in most weighted fictitious play specifications, in ACT-R we instead model forgetting by means of a power function.[6] Let $\mathbb{1}^{t}_{-\mathrm{i}}(a_{-\mathrm{i}}) \in \{0,1\}$ denote an indicator for whether player i's opponents chose actions $a_{-\mathrm{i}} \in A_{-\mathrm{i}}$ in period t and $\phi \in \mathbb{R}^{+}$ be a memory decay parameter. Also allow $\xi \in \mathbb{R}^{++}$ to be a parameter that controls the strength of a player's priors. Then given information I^{t}, player i's weighted fictitious play belief that his opponents will choose actions $a_{-\mathrm{i}}$ is

$$\mathbb{b}^{t}_{\mathrm{i}}(a_{-\mathrm{i}}) = \frac{\sum_{t'=0}^{t-1} \mathbb{1}^{t'}_{-\mathrm{i}}(a_{-\mathrm{i}}) \cdot (t-t')^{-\phi}}{\sum_{a'_{-\mathrm{i}} \in A_{-\mathrm{i}}} \sum_{t'=0}^{t-1} \mathbb{1}^{t'}_{-\mathrm{i}}(a'_{-\mathrm{i}}) \cdot (t-t')^{-\phi}} \tag{1.6}$$

where $\mathbb{1}^{0}_{-\mathrm{i}}(a_{-\mathrm{i}}) \equiv \xi \ \forall a_{-\mathrm{i}} \in A_{-\mathrm{i}}$ such that players have uniform priors. Note that a player's rate of forgetting is increasing in the decay parameter ϕ, with $\phi = 0$ representing the special case of standard fictitious play learning.

Equation (1.7) extends (1.6) to include beliefs that are behaviorally equivalent to frequency-style reinforcement and attraction learning models. These models can be framed as special types of belief learning in which agents condition their beliefs on their own upcoming action selections. When calculating the expected value of action a_{i}, a reinforcement learner uses the time-discounted frequencies with which his opponents have selected every combination of actions in all of the periods where he selected a_{i}. Observations of his opponents' action choices in any other period are omitted from the frequency calculation. A belief learner,

[6]↑The choice of decay rate reflects the trade off between realism and computational convenience, with power decay more closely resembling aggregated human rates of forgetting but exponential decay being more computationally convenient.

in contrast, uses his opponents' action choices in all previous periods regardless of whether he selected a_{i}. His observation weights are independent of the actions that he chose when he witnessed his opponents' action realizations. An attraction learner compromises between these two approaches by down-weighting the observations in which he did not choose a_{i}. The down-weighted observations receive a weighting of $\rho \in [0,1]$, which is experienced-weighted attraction's hypothetical reinforcement parameter (Camerer and Ho, 1999). When $\rho = 1$, an agent's beliefs are the same as those in a weighted fictitious play learning model. Conversely, when $\rho = 0$, his beliefs are behaviorally equivalent to a frequency-counting reinforcement learning model. It can therefore be said that as ρ tends towards zero, agents condition more strongly on their own upcoming action selections.

When calculating the expected value of action a_{i}, a reinforcement learner uses the time-discounted frequencies with which his opponents have selected every combination of actions in all of the periods where he selected a_{i}. Observations of his opponents' action choices in any other period are omitted from the frequency calculation. A belief learner, in contrast, uses his opponents' action choices in all previous periods regardless of whether he selected a_{i}. His observation weights are independent of the actions that he chose when he witnessed his opponents' action realizations. An attraction learner compromises between these two approaches by down-weighting the observations in which he did not choose a_{i}. The down-weighted observations receive a weighting of $\rho \in [0,1]$, which is experienced-weighted attraction's hypothetical reinforcement parameter (Camerer and Ho, 1999). When $\rho = 1$, an agent's beliefs are the same as those in a weighted fictitious play learning model. Conversely, when $\rho = 0$, his beliefs are behaviorally equivalent to a frequency-style reinforcement learning model. It can therefore be said that as ρ tends towards zero, agents condition more strongly on their own upcoming action selections.

$$\mathbb{b}^{t}_{\mathrm{i}}(a_{-\mathrm{i}}|a_{\mathrm{i}}) = \frac{\sum_{t'=0}^{t-1} \mathbb{1}^{t'}_{-\mathrm{i}}(a_{-\mathrm{i}}) \cdot [\rho + \mathbb{1}^{t'}_{\mathrm{i}}(a_{\mathrm{i}}) \cdot (1-\rho)] \cdot (t-t')^{-\phi}}{\sum_{a'_{-\mathrm{i}} \in A_{-\mathrm{i}}} \sum_{t'=0}^{t-1} \mathbb{1}^{t'}_{-\mathrm{i}}(a'_{-\mathrm{i}}) \cdot [\rho + \mathbb{1}^{t'}_{\mathrm{i}}(a_{\mathrm{i}}) \cdot (1-\rho)] \cdot (t-t')^{-\phi}} \tag{1.7}$$

1.3.2 Associative learning

My data learning model builds on this base-level learning in two important ways. First, it extends belief formation to include an implicit pattern recognition mechanism. In the style of classical conditioning, my model induces pattern recognition through a series of learned action associations, where associations are formed between tuples of the opponents' chosen actions and each lagged action choice contained in "the context". Understand a period's context to be the set of contextual cues that players attend to when making their decisions. In the interest of parsimony, I assume that players only attend to the actions that were selected in the previous period.[7] Consequently, I define $Q^t \equiv \{a_1^{t-1}, ..., a_n^{t-1}\}$ as the context in period $t > 1$ with $Q^1 \equiv \emptyset$.

When forming their beliefs, players consider how the available contextual cues influence the likelihood of observing each of their opponents' action combinations. The degree to which an action tuple is predicted by the presence of a cue is determined by the strength of their association: the greater the strength, the stronger a player's belief that his opponents will choose the associated actions. Associative strengths are normalized such that a value of zero indicates that the actions and cue have no association, while positive and negative strengths are taken to indicate positive and negative associations respectively. It is assumed that players begin the game with no learned action associations. Let $S_\mathrm{i}^t(a_{-\mathrm{i}}, q) \in \mathbb{R}$ denote player i's associative strength for actions $a_{-\mathrm{i}}$ and cue $q \in Q^t$ in period t and $\psi \in \mathbb{R}^+$ be a scaling parameter. Then pattern recognition can be appended to (1.7) by writing

$$\mathscr{b}_\mathrm{i}^t(a_{-\mathrm{i}}|a_\mathrm{i}) = \frac{\mathbb{b}_\mathrm{i}^t(a_{-\mathrm{i}}|a_\mathrm{i}) \cdot \exp(\psi \cdot \sum_{q \in Q^t} S_\mathrm{i}^t(a_{-\mathrm{i}}, q))}{\sum_{a'_{-\mathrm{i}} \in A_{-\mathrm{i}}} \mathbb{b}_\mathrm{i}^t(a'_{-\mathrm{i}}|a_\mathrm{i}) \cdot \exp(\psi \cdot \sum_{q \in Q^t} S_\mathrm{i}^t(a'_{-\mathrm{i}}, q))} \tag{1.8}$$

Note that $\mathscr{b}_\mathrm{i}^t(a_{-\mathrm{i}}|a_\mathrm{i})$ is a simple expression that satisfies the following criteria: i) $\mathscr{b}_\mathrm{i}^t(a_{-\mathrm{i}}|a_\mathrm{i}) \in [0, 1]$; ii) $\sum_{a_{-\mathrm{i}} \in A_{-\mathrm{i}}} \mathscr{b}_\mathrm{i}^t(a_{-\mathrm{i}}|a_\mathrm{i}) = 1$; iii) $\frac{\partial \mathscr{b}_\mathrm{i}^t(a_{-\mathrm{i}}|a_\mathrm{i})}{\partial S_\mathrm{i}^t(a_{-\mathrm{i}}, q)} \geq 0$; iv) $S_\mathrm{i}^t(a_{-\mathrm{i}}, q) = 0 \ \forall a_{-\mathrm{i}} \in A_{-\mathrm{i}}$ and $q \in Q^t \implies \mathscr{b}_\mathrm{i}^t(a_{-\mathrm{i}}|a_\mathrm{i}) = \mathbb{b}_\mathrm{i}^t(a_{-\mathrm{i}}|a_\mathrm{i})$. Together, criteria i) and ii) ensure that $\mathscr{b}_\mathrm{i}^t(a_{-\mathrm{i}}|a_\mathrm{i})$ is a well-behaved belief. Criterion iii) requires $\mathscr{b}_\mathrm{i}^t(a_{-\mathrm{i}}|a_\mathrm{i})$ to be weakly increasing in $a_{-\mathrm{i}}$'s association

[7] ↑Spiliopoulos (2013a and 2013b) consider models in which players attend to more distant lagged action choices but fail to select them over simpler lag-1 models.

with each cue in the context. Finally, criterion iv) demands that $\mathscr{b}_{\mathrm{i}}^t(a_{-\mathrm{i}}|a_{\mathrm{i}}) = \mathbb{b}_{\mathrm{i}}^t(a_{-\mathrm{i}}|a_{\mathrm{i}})$ in the absence of any learned associations.

After observing his opponents' action choices, player i updates his associative strengths for each $a_{-\mathrm{i}} \in A_{-\mathrm{i}}$ and $q \in Q^t$ pair. The manner in which these associations are learned is described by the Rescorla-Wagner model of classical conditioning (Rescorla and Wagner, 1972).[8] As in the motion equation used to model utility updating in (1.5), the Rescorla-Wagner model modulates learning according to a player's prediction errors. In this setting, player i tries to predict the value of $\mathbb{1}_{-\mathrm{i}}^t(a_{-\mathrm{i}})$. $\sum_{q \in Q^t} S_{\mathrm{i}}^t(a_{-\mathrm{i}}, q)$ can be interpreted as his prediction of this value given the available context. Let $\omega_{\mathrm{i}}(q) \in [0,1]$ denote the attentional weight that player i assigns to cue q such that $\sum_{q \in Q^t} \omega_{\mathrm{i}}(q) = 1$ and $\beta \in [0,1]$ be a step size parameter. Then $S_{\mathrm{i}}^t(a_{-\mathrm{i}}, q)$ can be said to evolve according to the following difference learning equation:

$$S_{\mathrm{i}}^{t+1}(a_{-\mathrm{i}}, q) = S_{\mathrm{i}}^t(a_{-\mathrm{i}}, q) + \omega_{\mathrm{i}}(q) \cdot \beta \cdot (\mathbb{1}_{-\mathrm{i}}^t(a_{-\mathrm{i}}) - \sum_{q' \in Q^t} S_{\mathrm{i}}^t(a_{-\mathrm{i}}, q')) \tag{1.9}$$

As in (1.5), the motion equation in (1.9) adjusts $\sum_{q \in Q^t} S_{\mathrm{i}}^{t+1}(a_{-\mathrm{i}}, q)$ in the direction of $\mathbb{1}_{-\mathrm{i}}^t(a_{-\mathrm{i}})$ by fraction β of the total prediction error. This adjustment is made indirectly, however, by updating each of the associative strengths individually. The sum of the adjustments made to these strengths is equal to $\beta \cdot (\mathbb{1}_{-\mathrm{i}}^t(a_{-\mathrm{i}}) - \sum_{q \in Q^t} S_{\mathrm{i}}^t(a_{-\mathrm{i}}, q))$, with the adjustment to $S_{\mathrm{i}}^{t+1}(a_{-\mathrm{i}}, q)$ accounting for fraction $w_{\mathrm{i}}(q)$ of the total adjustment. $w_{\mathrm{i}}(q)$ then controls the relative rate at which an action's association with cue q can be learned, increasing the rate the more salient cue q is compared to the other cues in the context. Note that $\sum_{q \in Q^t} w_{\mathrm{i}}(q) = 1$ also implies $w_{\mathrm{i}}(q) = w_{\mathrm{i}}(q')\ \forall q, q' \in A_{\mathrm{i}'}$, as otherwise there would exist a Q^t for which this constraint was not satisfied. $\sum_{q \in Q^t} w_{\mathrm{i}}(q) = 1$ then provides a complete characterization of player i's attentional weights up to $n - 1$ parameters.

[8]↑Previous versions of ACT-R modeled associative learning using a pseudo-Bayesian updating rule, but this approach was deprecated in ACT-R 6 due to problems with long-run model stability. More recent models have instead used variants of the Rescorla-Wagner learning rule, as they are supported by larger bodies of neural and behavioral evidence and are not susceptible to long-run model instability (Thomson and Lebiere, 2013; Thomson et al., 2014).

1.3.3 Probability weighting

My model's second extension to base-level learning introduces a nonlinear probability weighting function. In ACT-R, probability weighting naturally emerges from a memory retrieval process called blending. Blending returns a weighted average of any desired set of values that are stored in declarative memory. When the values being averaged are the outcomes of a repeated random event, blending behaves like a probability weighting function (see Appendix A.1.2). It therefore determines agents' sensitivities to probabilities as well as their levels of optimism. The degree to which agent's discriminate between action probabilities is controlled by the curvature of the probability weighting function. Likewise, the perceived attractiveness of gambling is controlled by the level of absolute weights, or the elevation of the probability weighting function. Let $\gamma \in \mathbb{R}^{+}$ and $\delta \in \mathbb{R}^{++}$ denote the weighting function's curvature and elevation parameters respectively. Then probability weighting can be appended to (1.8) by writing

$$b^t_{\mathrm{i}}(a_{-\mathrm{i}}|a_{\mathrm{i}}) = \frac{\delta \cdot \mathscr{b}^t_{\mathrm{i}}(a_{-\mathrm{i}}|a_{\mathrm{i}})^{\gamma}}{\delta \cdot \mathscr{b}^t_{\mathrm{i}}(a_{-\mathrm{i}}|a_{\mathrm{i}})^{\gamma} + \sum_{a'_{-\mathrm{i}} \neq a_{-\mathrm{i}} \in A_{-\mathrm{i}}} \mathscr{b}^t_{\mathrm{i}}(a'_{-\mathrm{i}}|a_{\mathrm{i}})^{\gamma}} \tag{1.10}$$

In the case where $n, |A_{\mathrm{j}}| = 2$, this equation is equivalent to the popular linear in log odds weighting function (Goldstein and Einhorn, 1987). More generally it follows the ratio form weighting function proposed by Lattimore et al. (1992). As an instance of these functions, equation (1.10) is capable of accommodating several key types of empirically relevant beliefs, including beliefs that are subcertain, beliefs that are subadditive, and beliefs that are subproportional.

The top two graphs in Figure 1.2 illustrate the effects of γ and δ on players' probability weighted beliefs. These graphs assume that $n, |A_{\mathrm{j}}| = 2$ to keep the weighting function's graphical representation two-dimensional. When $\gamma < 1$, players over-discriminate (under-discriminate) between the likelihoods of low/high (intermediate) probability actions; when $\gamma > 1$, the direction of these deviations from $\mathscr{b}^t_{\mathrm{i}}(a_{\mathrm{j}}|a_{\mathrm{i}})$ is reversed. Likewise, when $\delta < 1$, players are pessimistic (optimistic) regarding the value of gain (loss) domain gambles; when

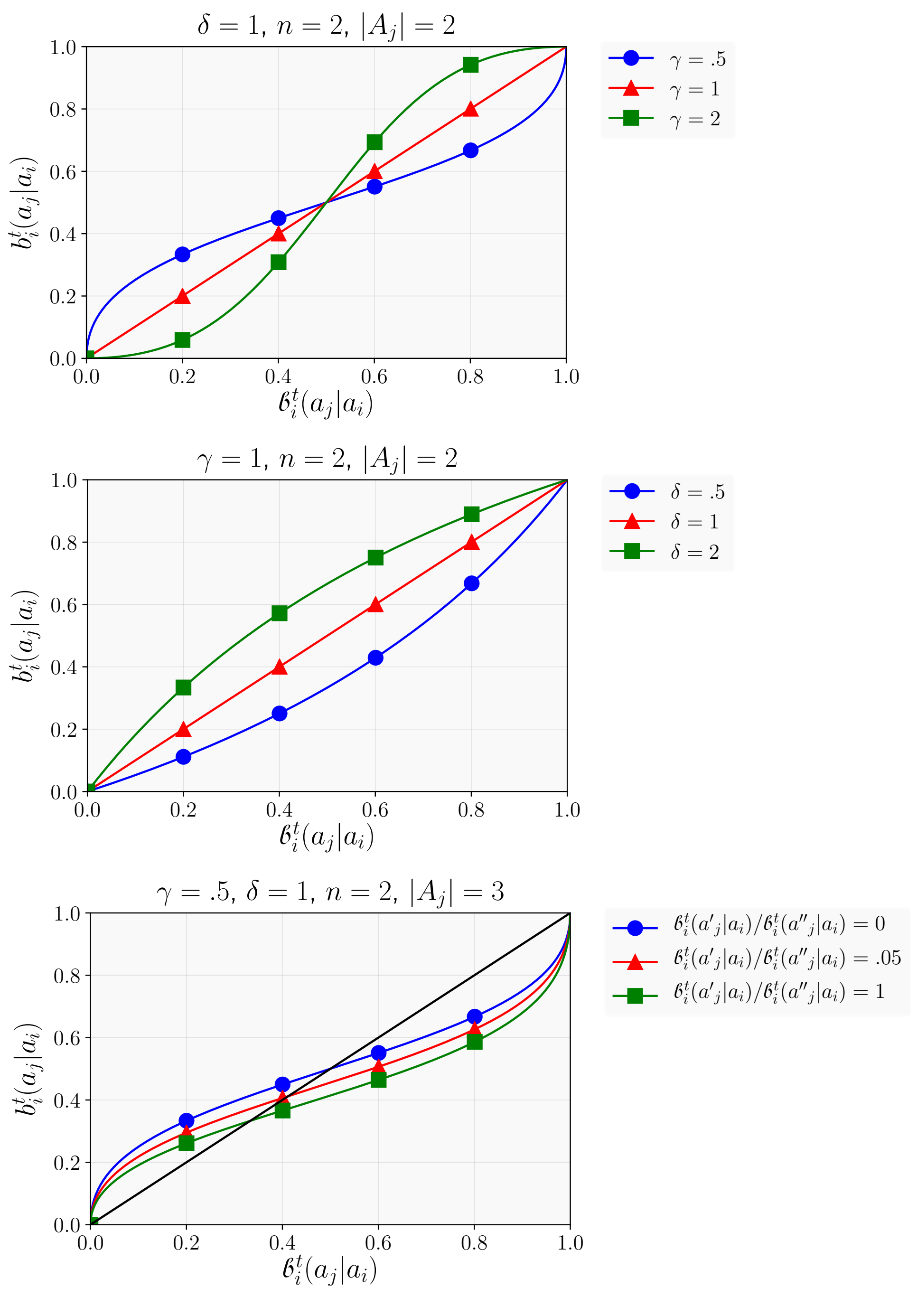


Figure 1.2. The effects of γ, δ, and $\beta_{\mathrm{i}}^t(a'_{\mathrm{j}}|a_{\mathrm{i}})/\beta_{\mathrm{i}}^t(a''_{\mathrm{j}}|a_{\mathrm{i}})$ on players' probability weighted beliefs

$\delta > 1$, players' levels of optimism within each payoff domain are reversed. Finally, when $\gamma, \delta = 1$, the weighting function is linear and $b^t_\mathrm{i}(a_\mathrm{j}|a_\mathrm{i}) = \mathscr{b}^t_\mathrm{i}(a_\mathrm{j}|a_\mathrm{i})$.

A unique feature of the weighting function in equation (1.10) is that it allows its fixed point to vary independently of the δ parameter. It does this by making its fixed point a function of the belief ratios for the opponents' other action combinations. If $\delta = 1$, the fixed point is chosen so that agents properly perceive when all possible outcomes are equiprobable. Equation (1.10) interpolates this prediction across the entire belief space, yielding a continuous prediction model. In ACT-R's standard weighting function, fixed point determination is entirely endogenous because there is no elevation parameter, but equation (1.10) includes δ so that the level of absolute of weights and the fixed point can vary exogenously.

The bottom graph in Figure 1 illustrates how the weighting function's fixed point decreases as $\mathscr{b}^t_\mathrm{i}(a'_\mathrm{j}|a_\mathrm{i})/\mathscr{b}^t_\mathrm{i}(a''_\mathrm{j}|a_\mathrm{i})$ increases from zero to one. Note that this graph assumes $n = 2$ and $|A_\mathrm{j}| = 3$ to keep the weighting function's graphical representation two dimensional. When $\mathscr{b}^t_\mathrm{i}(a'_\mathrm{j}|a_\mathrm{i}) = 0$, the fixed point is $\frac{1}{2}$ because that is where a_j and a''_j are equiprobable. Likewise, when $\mathscr{b}^t_\mathrm{i}(a'_\mathrm{j}|a_\mathrm{i}) = \mathscr{b}^t_\mathrm{i}(a''_\mathrm{j}|a_\mathrm{i})$, the fixed point is $\frac{1}{3}$ because that is where all three of the opponent's actions are equiprobable. Finally, when $0 < \mathscr{b}^t_\mathrm{i}(a'_\mathrm{j}|a_\mathrm{i}) < \mathscr{b}^t_\mathrm{i}(a''_\mathrm{j}|a_\mathrm{i})$, the fixed point is located somewhere between those two points.

Together, equations (1.7) through (1.10) completely characterize the mapping from players' observed action histories into their level-1 beliefs. The resulting model is flexible and nests weighted fictitious play learning as a special case when $\psi = 0$ and $\gamma, \delta, \rho = 1$. In the following chapter, I provide graphical evidence for this model's goodness of fit on a novel set of experimental data. Hypothesis testing is also conducted to determine the significance of the pattern recognition and probability weighting extensions.

1.4 Conclusion

This chapter developed a new model of learning and level-k reasoning in games. The model framed attraction learning in the language of beliefs and extended it to include two important features. The first of these features is an implicit pattern recognition mechanism that learns the importance of contextual information, while the second is a nonlinear probability weighting function with an endogenous fixed point location. The resulting beliefs determined level-1 behavior in a larger level-k rule learning model. In keeping with the literature, I assumed that rule learning occurs according to a reinforcement learning mechanism, but I improved the approximation of latent rule reinforcements to simulate the effect of rule exercise. A cognitive foundation for the full model was also provided by implementing it within the ACT-R cognitive architecture. In the following chapter, I provide graphical evidence for this model's goodness of fit on a novel set of experimental data. Hypothesis testing is also conducted to determine the significance of the extension parameters.

2. AN EXPERIMENTAL ANALYSIS OF LEVEL-K REASONING IN REPEATED MIXED STRATEGY GAMES

2.1 Introduction

Level-k reasoning is prominently featured in popular perceptions of repeated mixed strategy games. In the card game poker, for instance, strategy guides often refer to an analogous concept called “multiple level thinking”. But while many non-academics embrace level-k reasoning as a positive theory within this class of games, it is not at all clear that the model is descriptive of agents’ long run behavior in mixed strategy environments. Presently, there are surprisingly few analyses of level-k reasoning in repeated mixed strategy games. Data learning models have historically dominated attempts to characterize behavior in these environments.[1] From a level-k perspective, data learning restricts all players to level-1 reasoning. A more general level-k model remains to be estimated to determine the validity of this assumption.

Motivated by these considerations, this chapter investigates the extent to which human agents use level-k reasoning in repeated mixed strategy games. Towards this end, the model developed in Chapter 1 is estimated using data from a novel experiment. The experiment consisted of two between-subject treatments: the player-versus-player (PvP) treatment and the player-versus-data (PvD) treatment. In both treatments, subjects played a repeated version of a modified rock-paper-scissors game. The treatments varied the availability of information needed to use higher levels of reasoning. In the PvP treatment, the information provided was sufficient to use any level of reasoning, while in the PvD treatment subjects were only provided with enough information to be level-1. This restriction on reasoning was realized in part by making PvD players’ stage game payoffs private knowledge. Additionally, each PvD subject was matched against a predetermined sequence of actions instead of another live human player. The sequences they were matched with were taken directly from the action histories of PvP subjects in opposing player roles. PvD subjects were fully informed of this setup and made to understand the differences between the two treatments.

[1]↑Data learning models in mixed strategy games: Mookherjee and Sopher, 1994; Ochs, 1994; Cheung and Friedman, 1997; Camerer and Ho, 1999; Nyarko and Schotter, 2002; Ho et al., 2007; Rutström and Wilcox, 2009; Spiliopoulos, 2012, 2013a, and 2013b

The PvD treatment removes rule selection uncertainty so subjects' beliefs can be more easily inferred from their actions. This helps identify the model's belief learning parameters which are estimated by pooling the data from both treatments. In order to accommodate latent subject heterogeneity, I specify a random effects model with four random parameters. Maximum likelihood estimation therefore requires the simulation of a four-dimensional random effect. To make this estimation feasible, I combine GPU accelerated computing with an automatic differentiation method. Together these tools reduce the total estimation time by more than two orders of magnitude.

Comparing behavior between treatments, I find that PvD subjects behaved more adaptively than PvP players. This was determined by measuring their adherence to a win-stay-lose-shift strategy. While PvD subjects were 22.6 percent more likely to stay with their actions after a win than a loss, PvP subjects were actually 8.6 percent *less* likely to repeat their actions following a positive payoff. In the first period of the supergame, PvD row players were also more likely to select the level-1 best response. This action can be identified without estimation because agents begin the game with uniform level-1 priors. 82.1 percent of PvD row players selected this action compared to 42.9 percent of PvP row players. These results are consistent with PvP subjects using higher levels of reasoning. If we look at the action choices over all periods, however, we find a conflicting result: PvP row players were more likely to select an action that is never a best response to any pure strategy. In the absence of noise, agents using higher levels of reasoning would almost never select this response,[2] and yet PvP row players selected this action in 30.7 percent of their decisions. PvD row players chose this action as well, but they did so only 17.9 percent of the time. This difference suggests that PvP subjects scarcely used higher levels of reasoning.

Using the results of my estimation to simulate subjects' latent rule selections, I find that the initial proportions of level-k rules in the PvP treatment were 67.9 percent, 18.9 percent, and 13.3 percent for the level-1, level-2, and level-3 rules respectively. By the end

[2]↑All higher levels of reasoning respond to the level-1 rule, either directly or through an iterated best response. The level-1 rule will only prescribe a mixed strategy in the knife-edge cases where a player is indifferent between two or more of his actions. In a mixed strategy game, the best response to any of the opponent's pure strategies is also a pure strategy response, so an agent using higher levels of reasoning would almost never select an action that is never a best response to any pure strategy.

of the supergame, however, level-1 rule use had risen to 97.9 percent and the rest of the rules had fallen to negligible levels. Over the course of the entire supergame, PvP subjects used level-1 reasoning to make over 93 percent of their decisions, so it should come as no surprise that the level-k model does a poor job of explaining most of the long-run treatment differences. To see whether subjects left any money on the table by stopping at the first level of reasoning, I simulate the earnings of each level-k rule using the model's estimated parameters. In contrast to several related works that study level-k reasoning in repeated pure strategy games (Stahl, 1999, 2000, 2001, and 2003; Gill and Prowse, 2016), I find that earnings could not be appreciably improved by reasoning beyond level-1. This was because level-1 behavior became too unpredictable to exploit after only a couple of periods.

Taken together, these results reject the chapter's motivating folk wisdom and validate the use of adaptive learning models in mixed strategy games. They also differ from a couple of key findings from the level-k learning literature, namely, that players learn to use and earn more money by using higher levels of reasoning (Stahl, 1996, 1999, 2000, 2001, and 2003; Danz, Fehr, and Kübler, 2012; Ho and Su, 2012 and 2020; Gill and Prowse, 2016). One explanation for these discrepancies is that level-1 agents may have weaker preferences between their actions in repeated mixed strategy games. This leads to noisier behavior which in turn lowers the incentive for players to engage in higher levels of reasoning. If this explanation is correct, then my results should generalize to other repeated mixed strategy games. Further investigation of level-k reasoning in these environments will be needed to confirm the scope of my findings.

Although there have been several past analyses of level-k reasoning in repeated mixed strategy games, differences in the environments that they studied and the models that they used make comparisons of their results with mine somewhat difficult. In Stahl (1999), subjects played 15 different normal-form games for only a total of two periods. Three of these games were mixed strategy games while the remainder had unique pure strategy solutions. Subjects' levels of reasoning were only jointly inferred from their behavior across all 15 games. In Lindner and Sutter (2013), subjects repeated five periods of the 11-20 money request game. No model of data learning was used in this analysis, so subjects' levels of reasoning were almost certainly misidentified. Weerd et al. (2018) analyzed human subjects who were matched against computer opponents in two repeated matching pennies games. Subjects were told what level of reasoning the computer would noiselessly employ before they started playing the supergames. Unsurprisingly, many subjects were found to respond with the optimal levels of reasoning. Finally, Feng and Wang (2019) analyzed human behavior in two repeated mixed strategy games. They assumed a Poisson distribution of fixed level-k types who iteratively best responded to a level-0 attraction learner. It is worth noting though that the authors did not estimate any of the attraction learning parameters. Their estimated Poisson rates varied greatly between games (.441 and 2.64), so it difficult to draw any general conclusions.

The rest of the chapter is organized as follows. Section 2.2 describes the experimental design and explains my choice of repeated mixed strategy game. Section 2.3 summarizes the experimental data and compares behavior between the two treatments. Section 2.4 presents the structural analysis of subjects' chosen levels of reasoning. Section 2.5 concludes with a summary of the methods/results and takes a look towards future research.

2.2 Experiment

Ten experimental sessions were conducted at the Vernon Smith Experimental Economics Laboratory at Purdue University in the 2016 Fall semester. Eight to twelve subjects participated in each session with 112 subjects in total. Participants were recruited from Purdue University's undergraduate population using the ORSEE online recruiting system (Greiner, 2015). The experiment was programmed using the zTree toolbox for ready-made economic experiments (Fischbacher, 2007). Subjects received paper copies of the instructions which were read aloud to them before the start of the experiment (see Appendix B.1). They then completed a short comprehension quiz to make sure that they understood the instructions. Sessions took approximately 45 minutes to complete and participants were paid an average of $15.20. This amount included a $5 show-up fee as well as an additional $10.20 in average earnings during the experiment.[3]

[3]↑Subjects began the experiment with an initial endowment of experimental currency worth $15. This was intended to prevent them from finishing the experiment with negative earnings. While it was still possible to accrue negative earnings even with the $15 endowment, subjects were told that everyone in the experiment

PvP

		Period															
		1	2	3	4	5	6	7	...	34	35	36	37	39	39	40	
Player	row	M	T	M	T	M	T	T	...	B	M	T	T	B	M	T	1 Subject
	col	R	C	L	R	C	L	C	...	C	C	R	R	L	C	R	2

PvD

		Period															
		1	2	3	4	5	6	7	...	34	35	36	37	39	39	40	
Player	row	T	B	M	B	T	B	M	...	B	B	B	M	T	M	B	3 Subject
	col	R	C	L	R	C	L	C	...	C	C	R	R	L	C	R	2
Player	row	M	T	M	T	M	T	T	...	B	M	T	T	B	M	T	1 Subject
	col	C	C	L	L	R	C	C	...	L	R	C	R	R	R	C	4

Figure 2.1. Illustration of the sequence matching procedure used in the PvD treatment. In this example, subject 3 (a row player) is matched with the sequence generated by subject 2 (a column player) in the PvP treatment. Likewise, subject 4 (a column player) is matched with the sequence generated by subject 1 (a row player) in the PvP treatment.

2.2.1 Treatments

The experiment consisted of two between-subject treatments: the player-versus-player (PvP) treatment and the player-versus-data (PvD) treatment. The PvP treatment is the primary treatment of interest as it is the one from which I infer subjects' chosen level-k rules. Identification of these rules can be difficult, however, given their co-occurring latency with subjects' beliefs. To remedy this issue, the PvD treatment removes rule selection uncertainty by restricting subjects to the level-1 rule. This helps identify the model's belief learning parameters, and consequently, PvP subjects' beliefs.

In the PvP treatment, 56 subjects played a repeated mixed strategy game for which they were given full payoff information. Before starting the supergame, subjects were informed of their player roles and given two minutes to study the payoff table. This information was then reproduced for them on-screen in every decision-making period of the experiment. The stage game was repeated for 40 iterations with feedback provided after each period. This feedback included the player's own action, the action chosen by his opponent, and the stage

was guaranteed to leave with at least \$5. This lower bound on earnings never threatened to be binding, however, since no subject ever had less than \$10.

game payoffs they received. The opponent's response times, however, were intentionally obscured to prevent players from using them to infer their opponents' levels of reasoning. This was accomplished by instituting a minimum period duration of 30 seconds.[4] The supergame proceeded independently for each pair of players so as to preserve the statistical independence of their decisions. Pairs that finished early were instructed to wait quietly in their seats until everyone had finished the experiment.

The PvD treatment recreated this environment as closely as possible while trying to suppress higher levels of reasoning. Towards this end, another 56 subjects played the same game but with private payoff information. They were told how each pair of actions mapped into their own payoffs, but not the payoffs of the other player. At no point in the experiment were these mappings ever revealed through any feedback or payoff tables. Additionally, each PvD subject was matched against a predetermined sequence of actions instead of another live human player. The sequences they were matched with were taken directly from the action histories of PvP subjects in opposing player roles (see Figure 2.1). Each PvP subject had his sequence of actions matched with exactly one PvD player. PvD subjects were fully informed of this setup and made to understand the differences between the two treatments. More specifically, the following excerpt was read aloud to them from the experimental instructions:

> Before the start of the first period, you will be matched with a sequence of actions that were chosen by a participant in a **previous session**. The participant whose actions you will be matched with has been selected at random. Everyone in today's session will be matched with the actions of a different participant. You will remain matched with the **same** participant's sequence of actions for all 40 periods...The previous participant whose actions you will be matched with made his/her choices in an experiment that was nearly identical to the experiment being conducted today, with the main difference being that he/she was matched with another participant in the same session rather than with a sequence of actions that were chosen by a participant in a previous session. The participant whose actions you are matched with was not informed that his/her actions would be used in another experiment. The choices you make today will have no impact on the earnings of anyone other than yourself...You will only be able to see your own payoffs in this payoff table. These payoffs are the same payoffs that were given to the previous participants who

[4]↑Response times longer than 30 seconds could be observed by the first player in a pair to submit his action, but these only accounted for 9.3 percent of all PvP subjects' response times.

shared your player role, but they were also able to see the other player's payoffs in the payoff table.

By keeping payoff information private, the PvD treatment effectively withheld the opponent's best response function from each PvD player. Knowledge of these functions was needed in order for players to engage in higher levels of reasoning. While it might have been possible for subjects to learn about these functions had they been allowed to interact with other live human players, the sequence matching protocol used in this treatment ensured that they were never given this opportunity. Furthermore, since PvD subjects were not informed of the actions selected by the opponents of their matched PvP players, they were also prevented from using these actions to infer their opponents' level-1 beliefs. In this way, the sequence matching procedure itself acted as another channel to suppress higher levels of reasoning.

2.2.2 Stage Game

The stage game in Figure 2.2 was chosen on the basis of three primary criteria. First, it possesses a unique mixed strategy Nash equilibrium with mixture probabilities that are sufficiently far from uniform random. This creates payoff curvature and incentives for subjects to engage in thoughtful decision making. For the row player, these mixture probabilities are $\frac{42}{94}$, $\frac{25}{94}$, and $\frac{27}{94}$ for actions T (top), M (middle), and B (bottom) respectively, while for the column player they are $\frac{27}{94}$, $\frac{25}{94}$, and $\frac{42}{94}$ for actions L (left), C (center), and R (right). Second, the stage game is a zero sum game with no equitable lag-0 or lag-1 strategies. These features were included to discourage PvP subjects from experimenting with turn-taking strategies. Finally, for the row player in this game, action B is never a best response to any of his opponent's pure strategies.[5] In nearly every instance, however, a player's level-1 rule will prescribe a pure strategy best response. This implies that all higher level rules best respond to one of the opponent's pure strategies. Then in the absence of noise, a row player would

[5]↑Cason et al. (2010) uses a similar strategy to study learning dynamics two rock-paper-scissors style games.

	L	C	R
T	1, -1	-4, 4	2, -2
M	2, -2	5, -5	-4, 4
B	-3, 3	2, -2	1, -1

Figure 2.2. The repeated stage game used in the experiment. Payoffs are listed in "Francs," with a conversion rate of 10 Francs = 1 U.S. Dollar.

never select action B unless he also chose to be level-1.[6] Observations of B can therefore be used to help identify level-1 reasoning.

2.3 Data

This section summarizes the experimental data and compares behavior between the two treatments. The implications of these differences are discussed as they relate to subjects' chosen levels of reasoning. The behavior in both treatments is also compared to the Nash equilibrium for reference. All of the hypothesis testing in this section was conducted using bootstrapped t-tests with $2,000$ first stage replications. Closed form expressions for the standard deviations of the bootstrap estimators do not always exist, so they were estimated using a second stage of bootstrapping. 200 replications per first stage sample were conducted for $400,000$ replications in total. The samples in both stages were constructed by drawing with replacement from the smallest independent unit of observation, which in this case is a pair of matched PvP subjects and the two PvD subjects who were matched with their data. Consequently, in my data set of $4,480$ subject decisions, there are only 28 independent observations.

2.3.1 Action choices

Table 2.1 reports the empirical frequencies with which each action in the experiment was observed. The top half of the table reports the first period action frequencies while the bottom half reports the frequencies over all periods. Looking at the table, we see that

[6]↑In practice of course, there will always be some action selection noise. My structural estimation results indicate that PvP subjects selected action B 15.4 percent of the time while using higher levels of reasoning compared to 25.2 percent of the time when they were level-1. While this separation is not as large as I had hoped, it still helps identify higher levels of reasoning.

behavior in both treatments deviated substantially from the Nash equilibrium. Significant differences were discovered in nearly two thirds of the Nash-versus-treatment comparisons. As one would expect, the deviations from Nash were more pronounced in the supergame's first period. It would also appear that these deviations generally erred in the same directions across time frames and treatments. When there were differences in the magnitudes of the deviations from Nash, they tended to be smaller for PvP players, who (unlike their PvD counterparts) had a "defensive" incentive to adhere to the game's equilibrium strategies.

A number of differences between treatments were also observed for subjects in the row player role. Of particular interest is the first period difference in the frequency of action M. In period 1, action M is uniquely identified as the row player's level-1 action for all $r \leq 3$. The observed direction of this difference is therefore indicative of which treatment initially had more level-1 reasoning. From the table, we see that action M accounted for 42.9 percent of PvP row players' first period actions compared to 82.1 percent of PvD players' (two-sided $p = .006$). Row players were therefore initially more likely to select their level-1 action in the PvD treatment.

Table 2.1. Action frequencies

	Row players								Column players							
		Nash		PvP		PvD		Nash		Nash		PvP		PvD		Nash
Period 1	T	.447	$>$	.286	$\overset{*}{>}$	.071	$\overset{***}{<}$	.447	L	.287	$>$	.250	$>$	.107	$\overset{***}{<}$	.287
				(.086)		(.049)						(.082)		(.057)		
	M	.266	$\overset{*}{<}$	.429	$\overset{***}{<}$	.821	$\overset{***}{>}$	.266	C	.266	$\overset{***}{>}$	.071	$\overset{**}{<}$	.321	$>$	.266
				(.095)		(.074)						(.047)		(.088)		
	B	.287	$>$	.286	$\overset{**}{>}$	.107	$\overset{***}{<}$	.287	R	.447	$\overset{**}{<}$	.679	$>$	.571	$>$	.447
				(.084)		(.059)						(.089)		(.094)		
Periods 1-40	T	.447	$\overset{***}{>}$	.343	$\overset{*}{<}$	.407	$<$	.447	L	.287	$<$	.298	$>$	.285	$<$	.287
				(.016)		(.038)						(.018)		(.031)		
	M	.266	$\overset{***}{<}$	.350	$<$	.413	$\overset{***}{>}$	.266	C	.266	$\overset{***}{>}$	.179	$<$	.188	$\overset{***}{<}$	.266
				(.016)		(.036)						(.012)		(.025)		
	B	.287	$<$	.307	$\overset{***}{>}$	.179	$\overset{***}{<}$	.287	R	.447	$\overset{***}{<}$	.522	$<$	.528	$\overset{**}{>}$	.447
				(.020)		(.026)						(.016)		(.034)		

Bootstrap std. errors in parentheses. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

If we look at the action choices over all periods, however, we find a conflicting result: PvP row players were more likely to select action B which is never a best response to any pure strategy. In the absence of noise, agents using higher levels of reasoning would almost never select this response, and yet PvP row players selected this action in 30.7 percent of their decisions. PvD row players chose this action as well, but they did so only 17.9 percent of the time. This difference is highly significant (two-sided $p = .002$) and suggests that PvP subjects scarcely used higher levels of reasoning.

Table B.1 in Appendix B.2 disaggregates the empirical action frequencies by the lag-1 contexts in which the actions were observed. The table refers to these contexts as "states" and names them by concatenating the lagged actions. State TL, for instance, refers to the state in which actions T and L were selected in the previous period. Subjects' first period responses are excluded from the table since they were not preceded by any action history. Looking at the table, we again see that behavior in both treatments deviated substantially from the Nash equilibrium. Significant differences were discovered in nearly one half of the Nash-versus-treatment comparisons. The deviations from Nash generally erred in the same directions across treatments and states, although there was more variation in the directions of the deviations across states in the PvD treatment. It would therefore appear that PvD subjects departed more strongly from the history independence property of the mixed strategy Nash equilibrium.

The most notable takeaway from Table B.1 is the high level of inertia demonstrated by PvD players. Understand inertia to be the tendency for players to repeat their previous period's action. In the vast majority of states, PvD subjects were more inert than either Nash or PvP players. This was especially true of the states for which a subject's previous action resulted in a positive payoff. If the outcomes preceding these states are considered "wins" and all other outcomes are considered "losses", then PvD subjects (unlike their PvP counterparts) tended to be more inert after a win than a loss. This behavior resembles a well-known adaptive heuristic known as the win-stay-lose-shift strategy.

Table 2.2 further investigates action inertia by aggregating subjects' inert action frequencies over all states. A subject's chosen action is said to be "inert" if he also selected that action in the previous period. The bottom row of the table indicates that PvD subjects were nearly 1.5 times as likely to repeat their previous period's action as either Nash or PvP players (two-sided $p = .000$). This finding could be indicative of PvP subjects switching between different levels of reasoning. To understand why, note that if subjects slowly update their beliefs, the level-k rules will frequently repeat their prescribed actions. The prescriptions across rules will often vary, however, given the opposing nature of players' payoff functions. Switching between rules could therefore result in reduced action inertia.

Table 2.2. Inert action frequencies, periods 2-40

	Nash		PvP		PvD		Nash
Row players	.353	$\overset{**}{>}$	.299	$\overset{***}{<}$	.526	$\overset{***}{>}$	.353
			(.022)		(.033)		
Column players	.353	$\overset{**}{<}$	.403	$\overset{***}{<}$	.521	$\overset{***}{>}$	.353
			(.019)		(.032)		
All players	.353	$>$	.351	$\overset{***}{<}$	.523	$\overset{***}{>}$	.353
			(.017)		(.024)		

Bootstrap std. errors in parentheses. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

Table 2.3. Per-unit change in inert action frequencies following a win, periods 2-40

	Nash		PvP		PvD		Nash
Row players	0	$>$	-.111	$\overset{**}{<}$	.320	$\overset{**}{>}$	0
			(.100)		(.161)		
Column players	0	$>$	-.061	$<$	.133	$>$	0
			(.102)		(.157)		
All players	0	$>$	-.086	$\overset{**}{<}$	.226	$\overset{*}{>}$	0
			(.072)		(.120)		

Bootstrap std. errors in parentheses. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

Table 2.3 further investigates subject adaptivity by comparing inert action frequencies in winning and losing states. The table reports the per-unit change in these frequencies which can be viewed as a simple measure of adaptive behavior. The bottom row of the table reveals that while PvD subjects were 22.6 percent more likely to stay with their actions after a win than a loss, PvP subjects were actually 8.6 percent *less* likely to repeat their actions following a positive payoff. Only the former is significantly different from zero (two-sided

Table 2.4. Avg. abs. value of the z-statistic obtained by conducting a runs test for randomness, periods 1-40

	Nash		PvP		PvD		Nash
Row players	.80	$\overset{***}{<}$	1.26	$\overset{**}{<}$	1.88	$\overset{***}{>}$	.80
			(.17)		(.25)		
Column players	.80	$\overset{***}{<}$	1.15	$<$	1.51	$\overset{***}{>}$	.80
			(.13)		(.26)		
All players	.80	$\overset{***}{<}$	1.21	$\overset{***}{<}$	1.69	$\overset{***}{>}$	.80
			(.12)		(.18)		

Bootstrap std. errors in parentheses. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

p = .073), however, which is the value predicted by the Nash equilibrium. Even so, the treatment difference remains moderately significant (two-sided $p = .011$) and suggests that PvD subjects behaved more adaptively than PvP players. From Section 2, we know that adaptive learning is a hallmark of level-1 behavior, so it stands to reason that this result is consistent with PvP subjects using higher levels of reasoning.

Finally, in Table 2.4 I report an averaged measure of randomness for the action sequences produced by each player. This measure, $|z|$, is the absolute value of the z-statistic obtained by conducting a one-sample runs test for randomness. A runs test compares the number of runs in a sequence to the distribution of runs if the sequence were random.[7] The more extreme the z-statistic is then (i.e., the further from 0), the less likely the sequence is to arise under a random data generating process. From the bottom row of the table, we see that the average $|z|$ was 1.21 for subjects in the PvP treatment compared to 1.69 for PvD players. The average PvP sequence was therefore twice as likely as the average PvD sequence under the null hypothesis of randomness (two-sided $p = .010$). Even so, this sequence was only two-thirds as likely as the average equilibrium sequence (two-sided $p = .000$). Once again, the observed treatment difference could be explained by PvP subjects using higher levels of reasoning, since rule selection adds additional noise to the action selection process.

[7]↑In this context, randomness should be understood to mean that the elements of a sequence are mutually independent. A runs test conditions on the number of times each element appears in a sequence and is therefore not a test to determine whether the data generating process is uniform random.

2.3.2 Response times

As a procedural theory, level-k reasoning specifies the steps of cognition needed to use each level-k rule. Because these steps are iterative, the model is able to make ordinal response time predictions. More specifically, the model posits that each level of reasoning takes longer for an agent to use than the preceding level, since it requires all the same steps of cognition plus one additional best response mapping. If we assume that the time it takes to use each level of reasoning is constant across the two treatments, slower treatment-level response times can be associated with subjects using higher levels of reasoning (Alaoui and Penta, 2022).

Table 2.5 reports subjects' average response times by their chosen actions in the PvP and PvD treatments. Response time is measured as the latency between a subject's button press to start the new period and his button press to submit his chosen action. From the bottom row of the table, we see that the average PvP response time over all actions was seven seconds slower than the average for PvD players (two-sided $p = .000$). A comparison of median response times is also provided in Appendix B.2 and it yields qualitatively similar results (see table B.2). As noted earlier, slower response times could be indicative of subjects using higher levels of reasoning, so these results are consistent with PvP subjects reasoning beyond level-1.

Table 2.5. Average response times by chosen action, periods 2-40

	Row players				Column Players		
	PvP		PvD		PvP		PvD
T	16.9	$\overset{***}{>}$	8.4	L	15.6	$\overset{***}{>}$	10.2
	(2.0)		(.8)		(1.8)		(1.1)
M	18.0	$\overset{***}{>}$	9.1	C	15.0	$\overset{***}{>}$	8.4
	(2.1)		(.7)		(1.7)		(.8)
B	15.9	$\overset{**}{>}$	11.4	R	14.9	$\overset{***}{>}$	8.1
	(1.6)		(1.2)		(1.5)		(.7)
All	17.0	$\overset{***}{>}$	9.2	All	15.1	$\overset{***}{>}$	8.8
	(1.9)		(.7)		(1.5)		(.7)

Bootstrap std. errors in parentheses. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

The average PvP response times were also slower than the average PvD response times for every one of the individual actions. Notably, this also includes action B, which should only be selected when agents are level-1. The presence of action selection noise can account for this difference by allowing action B to sometimes be selected when using higher levels of reasoning. Given the rarity of this event, however, PvP row players should still select action B more quickly on average than they select alternative actions. The average response time for action B relative to the other two actions should also be smaller in the PvP treatment.

Table 2.6 investigates these hypotheses by reporting the per-unit changes in subjects' average response times when they selected each action. From the bottom row of the table, we see that action B was selected 9 percent faster on average than the other two actions in the PvP treatment (two-sided $p = .067$). In the PvD treatment, conversely, action B was selected 30.8 percent *slower* on average than the alternative actions (two-sided $p = .000$). The difference in these percentages is highly significant (two-sided $p = .000$) and in the direction predicted by PvP subjects using higher levels of reasoning.

Table 2.6. Per-unit change in average response times by chosen action, periods 2-40

Row players

			PvP		PvD		
T	0	$>$	-.006	$\overset{*}{>}$	-.139	$\overset{*}{<}$	0
			(.034)		(.074)		
M	0	$\overset{**}{<}$	.097	$\overset{*}{>}$	-.030	$<$	0
			(.042)		(.072)		
B	0	$\overset{*}{>}$	-.090	$\overset{***}{<}$	.308	$\overset{***}{>}$	0
			(.044)		(.104)		

Column players

			PvP		PvD		
L	0	$<$	.045	$\overset{*}{<}$	.242	$\overset{**}{>}$	0
			(.061)		(.109)		
C	0	$>$	-.005	$>$	-.044	$<$	0
			(.084)		(.079)		
R	0	$>$	-.033	$>$	-.147	$\overset{*}{<}$	0
			(.046)		(.078)		

Bootstrap std. errors in parentheses. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

To summarize the main results of this section, PvP subjects were found to be less inert, less adaptive, and less predictable than their PvD counterparts. They were also less likely to select their level-1 action in the first period of the experiment and generally took longer to make their decisions. All of these findings are consistent with the hypothesis that they used higher levels of reasoning. In contrast to these findings, however, PvP row players were also more likely to select action B, an action that is normally reserved for agents who are level-1.

On the surface it would seem that this result should be sufficient to rule out the higher-level reasoning hypothesis.

Unfortunately, none of the aforementioned findings are sufficient to establish the extent to which PvP subjects used higher levels of reasoning, as any one of these treatment differences could be explained by a number of alternative hypotheses. All of the action choice differences, for example, could be explained by the presence of stronger equilibrating forces in the PvP treatment, since all of the PvP treatment's departures from the PvD action choices are in the direction of the Nash equilibrium. Alternatively, these action choice differences could be explained by PvP subjects responding more noisily to make themselves less predictable to opposing players, since all of the PvP treatment's departures from the PvD action choices are also in the direction of uniform randomness. Ultimately then, we will need to a conduct a structural analysis to determine the extent to which PvP subjects used higher levels of reasoning, as this will allow us to control for such factors and evaluate the contribution of each hypothesis.

2.4 Analysis

This section summarizes the structural analysis that is used to infer subjects' chosen levels of reasoning. Towards this end, the model developed in Chapter 1 is parameterized and subsequently estimated. Several related specifications are also considered to evaluate the previous section's alternative hypotheses. In order to assess model fit, subjects' action choices are simulated and plotted alongside their actual behavior. Counterfactual exercises are also conducted to explore the relationship between earnings and chosen level of reasoning.

2.4.1 Econometric specification

The model in Chapter 1 requires us to specify the values of $2r + 14$ unknown parameters. These parameters include the rule learning step-size α, associative learning step-size β, weighting function curvature γ, weighting function elevation δ, utility sensitivity κ, payoff sensitivity λ, action selection biases $\nu(a_\mathrm{i})\ \forall a_\mathrm{i} \in A_\mathrm{i} \setminus \{a'_\mathrm{i}\}$ and $\forall \mathrm{i} \in N$, prior belief strength ξ, initial rule utilities $\boldsymbol{\pi}(k)\ \forall k \in \{1, ..., r\}$, hypothetical reinforcement weight ρ, rule selection

Table 2.7. Summary of model parameters

Parameter	Description	Range	Interpretation
α	rule learning step-size	[0, 1]	$\alpha = 0 \Rightarrow$ no rule learning $\alpha = 1 \Rightarrow$ instant rule learning
β	associative learning step-size	[0, 1]	$\beta = 0 \Rightarrow$ no associative learning $\beta = 1 \Rightarrow$ instant associative learning
γ	weighting function curvature	$[0, \infty)$	$\gamma < 1 \Rightarrow$ over-discriminate between low prob. $\gamma > 1 \Rightarrow$ under-discriminate between low prob.
δ	weighting function elevation	$(0, \infty)$	$\delta < 1 \Rightarrow$ pessimistic over gain domain gambles $\delta > 1 \Rightarrow$ optimistic over gain domain gambles
κ	utility sensitivity (USD)	$[0, \infty)$	$\kappa = 0 \Rightarrow$ rule selection is uniform random $\kappa \to \infty \Rightarrow$ rule selection is deterministic
λ	payoff sensitivity (USD)	$[0, \infty)$	$\lambda = 0 \Rightarrow$ action selection is uniform random $\lambda \to \infty \Rightarrow$ action selection is deterministic
$\nu(a_\mathrm{i})$	action selection bias	$(-\infty, \infty)$	$\nu(a_\mathrm{i}) \to -\infty \Rightarrow$ action a_i is never chosen $\nu(a_\mathrm{i}) \to \infty \Rightarrow$ action a_i is always chosen
ξ	prior belief strength	$(0, \infty)$	$\xi \to 0 \Rightarrow$ beliefs are entirely learned $\xi \to \infty \Rightarrow$ beliefs are entirely priors
$\pi(k)$	initial utility of rule k	$(-\infty, \infty)$	$\pi(k) \to -\infty \Rightarrow$ rule k is never chosen $\pi(k) \to \infty \Rightarrow$ rule k is always chosen
ρ	hypo. reinforcement weight	[0, 1]	$\rho = 0 \Rightarrow$ pure reinforcement learning $\rho = 1 \Rightarrow$ pure belief learning
$\upsilon(k)$	rule selection bias	$(-\infty, \infty)$	$\upsilon(k) \to -\infty \Rightarrow$ rule k is never chosen $\upsilon(k) \to \infty \Rightarrow$ rule k is always chosen
ϕ	memory decay rate	$[0, \infty)$	$\phi = 0 \Rightarrow$ perfect memory $\phi \to \infty \Rightarrow$ no memory
ψ	associative strength scaler	$[0, \infty)$	$\psi = 0 \Rightarrow$ no pattern recognition $\psi \to \infty \Rightarrow$ only pattern recognition
$\omega_\mathrm{i}(a_\mathrm{i})$	attn. weight on own actions	[0, 1]	$\omega_\mathrm{i}(a_\mathrm{i}) = 0 \Rightarrow$ no attn. on own actions $\omega_\mathrm{i}(a_\mathrm{i}) = 1 \Rightarrow$ all attn. on own actions

biases $\upsilon(k)$ $\forall k \in \{2, ..., r\}$, memory decay rate ϕ, associative strength scaler ψ, and attentional weight $\omega_\mathrm{i}(a_\mathrm{i})$. Naturally, we will want to estimate these parameters using Maximum Likelihood Estimation (MLE), but first we need to make several decisions regarding the econometric specification. One such decision concerns our choice of panel data model.

The tension in the choice of panel model is characterized by the well-known bias-variance trade-off. On one end of the model space sits a pooled panel model that assigns all subjects the same set of parameters. The resulting estimators will generally have the lowest variance of any panel estimators but the highest subject-level heterogeneity bias. On the other end of the model space sits a fixed effects model that assigns each subject a unique set of parameters. The resulting estimators will generally have the highest variance of any panel estimators but the lowest subject-level heterogeneity bias. Between these two models sits a random effects model that estimates some pooled and some random parameters. This specification is recommended for adaptive learning models by Wilcox (2006), and so it is the specification that I adopt for this analysis.

The random effects specification allows us to model subject heterogeneity with relatively few additional parameters. This comes at the cost of increased computational complexity, however, since each random effect adds another nonelementary integral to the likelihood function. For this reason, the random effects are reserved for the model's most important and heterogeneous parameters. These include payoff sensitivity $\lambda \sim \text{Lognormal}(\mu_\lambda, \sigma_\lambda^2)$, memory decay rate $\phi \sim \text{Lognormal}(\mu_\phi, \sigma_\phi^2)$, and initial rule utilities $\boldsymbol{\pi}(k) \sim \text{Normal}(\mu_{\boldsymbol{\pi}(k)}, \sigma_{\boldsymbol{\pi}(k)}^2)$ $\forall k \in \{2, ..., r\}$. λ is included as a random effect because it has been previously identified as a large source of heterogeneity bias in a model's belief learning parameters (Wilcox, 2006). ϕ is included because it is generally believed to be a belief learning model's most important parameter. Finally, $\boldsymbol{\pi}(k)$ $\forall k \in \{2, ..., r\}$ are included because the initial distribution of reasoning strongly affects level-k learning dynamics.[8] These parameters are also normalized by defining $\boldsymbol{\pi}^*(k) \equiv \kappa \cdot (\boldsymbol{\pi}(k) - \boldsymbol{\pi}(1))$ $\forall k \in \{2, ..., r\}$ so that they can still be identified in the absence of rule learning.

[8]↑Stahl (1999) also modeled subjects' initial rule utilities as normally distributed random variables, although he used this specification to approximate their utility distributions across different games instead of across different subjects.

In the interest of parsimony and computational feasibility, my main estimation restricts PvP subjects' depth of strategic reasoning to level-3. By virtue of the PvD treatment's design, its subjects are instead restricted to the first level of reasoning. All of the model's belief learning and action choice parameters are assumed to be the same across treatments. The restrictions $\delta = 1$, $\xi = 1$, and $\rho = 1$ are also imposed because these parameters can be difficult to identify. Finally, $\upsilon(k)$ is set to zero $\forall k \in \{2, ..., r\}$ since rule selection is latent. The matrix of random effects $\boldsymbol{\eta} = \begin{bmatrix} \ln(\boldsymbol{\lambda}) & \boldsymbol{\pi}^*(2) & \boldsymbol{\pi}^*(3) & \ln(\boldsymbol{\phi}) \end{bmatrix}$ is drawn from a quadrivariate normal distribution with mean vector $\boldsymbol{\mu} = \begin{bmatrix} \mu_{\ln(\lambda)} & \mu_{\boldsymbol{\pi}^*(2)} & \mu_{\boldsymbol{\pi}^*(3)} & \mu_{\ln(\phi)} \end{bmatrix}$ and covariance vector $\boldsymbol{\sigma} = \begin{bmatrix} \sigma^2_{\ln(\lambda)} & \sigma_{\ln(\lambda),\boldsymbol{\pi}^*(2)} & \sigma^2_{\boldsymbol{\pi}^*(2)} & \sigma_{\ln(\lambda),\boldsymbol{\pi}^*(3)} & \sigma_{\boldsymbol{\pi}^*(2),\boldsymbol{\pi}^*(3)} & \sigma^2_{\boldsymbol{\pi}^*(3)} & \sigma_{\ln(\lambda),\ln(\phi)} & \sigma_{\boldsymbol{\pi}^*(2),\ln(\phi)} & \sigma_{\boldsymbol{\pi}^*(3),\ln(\phi)} & \sigma^2_{\ln(\phi)} \end{bmatrix}$. Let $F(\boldsymbol{\eta}; \boldsymbol{\theta})$ denote the cumulative distribution function of the random effects and $\boldsymbol{\theta} = \begin{bmatrix} \alpha & \beta & \gamma & \kappa & \boldsymbol{\mu} & \nu(M) & \nu(B) & \nu(C) & \nu(R) & \boldsymbol{\pi}(1) & \boldsymbol{\sigma} & \psi & \omega_{\mathrm{i}}(a_{\mathrm{i}}) \end{bmatrix}$ be the vector of free parameters. Then the likelihood function corresponding to my main estimation can be expressed as follows:

$$L(\boldsymbol{\theta}) = \prod_{\mathrm{i}=1}^{112} \iiiint \prod_{t=1}^{40} \sum_{a_{\mathrm{i}} \in A_{\mathrm{i}}} \mathbb{1}^t_{\mathrm{i}}(a_{\mathrm{i}}) \sum_{k=1}^{r_{\mathrm{i}}} p^t_{\mathrm{i}}(k; \boldsymbol{\eta}, \boldsymbol{\theta}) p^t_{\mathrm{i}}(a_{\mathrm{i}} | k; \boldsymbol{\eta}, \boldsymbol{\theta}) dF(\boldsymbol{\eta}; \boldsymbol{\theta}) \tag{2.1}$$

2.4.2 Estimation procedure

The multiple integral in (2.1) cannot be evaluated through ordinary analytical methods. It can, however, be approximated numerically or through random effects simulation. The later is preferred for its superior accuracy when dealing with high dimensional problems. For this reason, the model is estimated using the simulated analogue of Maximum Likelihood Estimation. $4,000$ simulations were conducted to approximate the integral at each step of the optimization. The random effect draws were also seeded identically for every evaluation of the likelihood function.

The likelihood function is optimized using 1000 iterations of Wales and Doye (1997)'s basin-hopping algorithm. Basin-hopping is a global estimation technique that combines the efficiency of local optimization routines with guided exploration of the parameter space induced by random perturbations. The local optimizer I use is a gradient-based method known as Sequential Least Squares Programming (SLSQP) (Kraft, 1988). SLSQP is a constrained optimization routine that in the unconstrained case collapses to Newton's method. Naturally, I constrain the covariance matrix for the four random effects to be positive semidefinite.

Gradient calculation poses another potential problem for parameter estimation, as there are no closed form expressions for the gradient of the likelihood function. Numerical approximations of the gradient can be very time-intensive, and if first-order approximations are used, also inaccurate. For these reasons, I calculate the gradient using an automatic differentiation method. Automatic differentiation records the sequence of elementary operations that are used by your computer when it evaluates the likelihood function. It then computes the derivatives of this expression for you "automatically" through repeated application of the chain rule. The resulting derivatives are calculated quickly and accurately up to machine precision.

Even with the performance improvements brought about by using automatic differentiation, global optimization of the likelihood function still takes considerable time and computational resources. This is owed in large part to the four random effects and the many simulations they require. To make the estimation feasible, I combine GPU accelerated computing with the automatic differentiation method. Together these tools reduce the total estimation time by more than two orders of magnitude. GPU computing is a type of parallel processing that excels at performing element-wise operations. It is conveniently supported by the PyTorch package for Python (Paszke et al., 2019) along with automatic differentiation. Alternatively, GPU's can be programmed in Python using the PyCUDA or PyOpenCL packages (Klöckner et al., 2012).

GPU memory capacity is typically much smaller than it is for CPU's of similar quality. This poses problems for the estimation of my model which requires 16GB of GPU memory. Most of this memory is needed to maintain the computational graph for automatic gradient calculation. 16GB GPU's cost nearly \$3,000, which for a graduate student is prohibitively expensive, so I instead purchased time on Nvidia's Tesla GPU's with a monthly Google Colab subscription. Even with a premium subscription, Colab limits program runtime to a maximum of 24 hours. This is enough time for my estimation, but not nearly enough time for the entirety of my bootstrap procedure. Consequently, I had to conduct several smaller bootstraps to obtain the desired number of bootstrap samples.

2.4.3 Estimation results

All of the single-parameter (multi-parameter) significance testing in this subsection was conducted using bootstrapped t-tests (Wald tests) with $1,000$ bootstrap replications.[9] For the sake of computational feasibility, the standard deviations of the bootstrap estimators themselves were estimated using the jackknife method. As in Section 3, the bootstrap samples were constructed by drawing with replacement from the smallest independent unit of observation, which as mentioned before is a pair of matched PvP subjects and the two PvD subjects who were matched with their data.

Table 2.8 reports the results of five closely related estimations. These estimations vary in what data they use and the restrictions they place on subjects' chosen levels of reasoning. In estimation (1), the model's parameters are estimated using only the PvD data. Estimations (2) and (4) are likewise performed using data from only the PvP treatment. Estimations (3) and (5), conversely, use the data from both treatments to estimate the model's belief learning and action-choice parameters. In all estimations, PvD subjects are restricted to the first level of reasoning. Estimations (2) and (3) also restrict PvP subjects from reasoning beyond level-1. Estimations (4) and (5), conversely, allow PvP subjects to also use level-2 and level-3 reasoning. Estimation (5) is then the primary estimation of interest as it estimates the full level-k model using the data from both treatments. The other specifications are used to show the contribution of level-k reasoning and inter-treatment parameter heterogeneity in explaining the observed treatment differences.

The first items of interest are determining whether PvP players initially used and learned to use different levels of reasoning. Looking at the estimates for the initial rule utility means in Table 2.8, we see that $\mu_{\pi^*(3)}$ is significant in both of the level-k estimations, while $\mu_{\pi^*(2)}$ is only significant in estimation (5). The means are jointly significant in both estimations as seen in Table 2.9. The variances of the initial rule utilities $\sigma^2_{\pi^*(2)}$ and $\sigma^2_{\pi^*(3)}$ are also significant across both estimations, both jointly and at the individual parameter level. The initial rule

[9]↑The asymptotic distributions of the likelihood ratio, Wald, and score test statistics do not follow chi-square distributions under the null hypotheses for many of the significance tests conducted in this section. Instead, they follow complicated mixtures of chi-square distributions whenever multiple null parameters lie on the boundary of the parameter space. Bootstrapping avoids this difficult distributional theory by simulating the distribution of the Wald test statistics directly.

Table 2.8. Parameter estimates

Parameter	Range	Null	(1)	(2)	(3)	(4)	(5)
α	[0, 1]	0				.011*	.018**
						(.007)	(.006)
β	[0, 1]	0	.093**	.165***	.100**	.179*	.087**
			(.039)	(.060)	(.036)	(.039)	(.026)
γ	[0, 10]	1	.492*	.158**	.469**	.363	.542*
			(.127)	(.199)	(.120)	(.116)	(.090)
κ	[0, 10]	0				1.15**	2.88**
						(.057)	(.137)
$\mu_{\ln(\lambda)}$	[-10, ln(10)]	-7	1.97***	2.30***	1.70***	1.96**	1.66**
			(.261)	(.423)	(.243)	(.154)	(.164)
$\mu_{\pi^*(2)}$	[-10, 10]	-7				-7.07	-1.28*
						(.051)	(.848)
$\mu_{\pi^*(3)}$	[-10, 10]	-7				-2.25*	-1.81**
						(.137)	(.355)
$\mu_{\ln(\phi)}$	[-10, ln(10)]	-7	-1.69***	-7.37	-2.83***	-5.92**	-3.02*
			(.934)	(1.82)	(1.02)	(.058)	(.446)
ν(M)	[-10, 10]	0	.383	-.695**	.196	-.077	.333
			(.326)	(.371)	(.220)	(.123)	(.163)
ν(B)	[-10, 10]	0	-.416***	-.209*	-.194*	.028	-.137
			(.167)	(.129)	(.105)	(.102)	(.092)
ν(C)	[-10, 10]	0	.114	.446**	-.020	-.150	-.062
			(.232)	(.275)	(.189)	(.144)	(.143)
ν(R)	[-10, 10]	0	.260**	.150	.325***	.219	.319
			(.129)	(.130)	(.088)	(.104)	(.083)
$\pi(1)$	[-10, 10]	0				-5.20***	-7.45**
						(.014)	(.062)
$\sigma^2_{\ln(\lambda)}$	(0, 10]	0	.801***	.016	.771***	.117	.752*
			(.366)	(.099)	(.223)	(.133)	(.168)
$\sigma_{\ln(\lambda),\pi^*(2)}$	[-10, 10]	0				.697	.221*
						(.109)	(.047)
$\sigma_{\ln(\lambda),\pi^*(3)}$	[-10, 10]	0				.151	.602*
						(.070)	(.130)
$\sigma_{\ln(\lambda),\ln(\phi)}$	[-10, 10]	0	.021	.377*	.512**	.232	.579
			(.453)	(.449)	(.340)	(.070)	(.202)
$\sigma^2_{\pi^*(2)}$	(0, 10]	0				4.25**	.068**
						(.029)	(.016)
$\sigma_{\pi^*(2),\pi^*(3)}$	[-10, 10]	0				.913	.194*
						(.057)	(.041)
$\sigma_{\pi^*(2),\ln(\phi)}$	[-10, 10]	0				1.08*	.252*
						(.011)	(.061)
$\sigma^2_{\pi^*(3)}$	(0, 10]	0				.525 **	.581*
						(.076)	(.176)
$\sigma_{\pi^*(3),\ln(\phi)}$	[-10, 10]	0				-.482**	.885
						(.052)	(.205)
$\sigma^2_{\ln(\phi)}$	(0, 10]	0	2.45**	9.96***	3.29*	5.10**	3.03*
			(1.68)	(1.66)	(1.86)	(.024)	(.489)
ψ	[0, 10]	0	4.18**	4.22	4.05**	2.77**	4.00
			(1.97)	(2.21)	(1.85)	(.094)	(.971)
$\omega_i(a_i)$	[0, 1]	0		.260**	.292**	.231	.318
				(.153)	(.162)	(.138)	(.140)
Data			PvD	PvP	All	PvP	All
Log likelihood			-1962.47	-2295.85	-4296.37	-2289.55	-4278.40

Bootstrap std. errors in parentheses. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

Table 2.9. *p*-values for multiple hypothesis tests

Parameters	Description	(1)	(2)	(3)	(4)	(5)
α, κ, $\boldsymbol{\pi}(1)$	rule learning				.085	.061
$\mu_{\boldsymbol{\pi}^*(2)}$, $\mu_{\boldsymbol{\pi}^*(3)}$	initial rule utility means				.086	.094
$\sigma^2_{\boldsymbol{\pi}^*(2)}$, $\sigma^2_{\boldsymbol{\pi}^*(3)}$	initial rule utility variances				.064	.079
$\boldsymbol{\sigma}^*$	initial rule utility covariances				.356	.350
$\Uparrow$	all of the above				.421	.277
β, ψ, ω	pattern recognition	.025	.059	.024	.122	.127
$\boldsymbol{\theta}^*_{(x)} - \boldsymbol{\theta}_{(\mathbf{1})}$	treatment differences		.607		.484	

$\boldsymbol{\sigma}^* \equiv \begin{bmatrix}\sigma_{\ln(\lambda),\boldsymbol{\pi}^*(2)} & \sigma_{\ln(\lambda),\boldsymbol{\pi}^*(3)} & \sigma_{\boldsymbol{\pi}^*(2),\boldsymbol{\pi}^*(3)} & \sigma_{\boldsymbol{\pi}^*(2),\ln(\phi)} & \sigma_{\boldsymbol{\pi}^*(3),\ln(\phi)}\end{bmatrix}$, $\boldsymbol{\theta}^*_{(x)} \equiv \boldsymbol{\theta}_{(x)} \cap \boldsymbol{\theta}_{(\mathbf{1})}\ \forall x \in \{2,3\}$

utility covariances, however, are jointly insignificant in estimations (4) and (5). Even so, the significance of the mean and variance parameters provides some evidence that PvP subjects initially used (and varied in using) higher levels of reasoning. Turning our attention to the rule learning parameters, we see that the rule learning step-size α, rule selection sensitivity parameter λ, and initial rule utility $\boldsymbol{\pi}(1)$ are individually and jointly significant in both of the level-k estimations. These results suggest that PvP subject learned to used different levels of reasoning. Finally, a joint test of all the level-k parameters reveals that they are jointly insignificant in estimations (4) and (5). It is largely the inclusion of the covariance parameters that leads to the model being over-specified.

Figure 2.3 shows the simulated distribution of subjects' rule frequencies using the estimated parameters. From these distributions, we see that nearly all subjects overwhelmingly used level-1 reasoning. In fact, over 93 percent of subjects decisions were made using level-1 reasoning in both estimations (4) and (5). From Figure 2.4 however, we see this wasn't always the case, as players learned to increasingly use level-1 reasoning. In estimation (5), the initial proportion of level-k rules in the PvP treatment was 67.9 percent, 18.9 percent, and 13.3 percent for the level-1, level-2, and level-3 rules respectively. By the end of the supergame, however, level-1 rule use had risen to 97.9 percent and the rest of the rules had fallen to negligible levels.

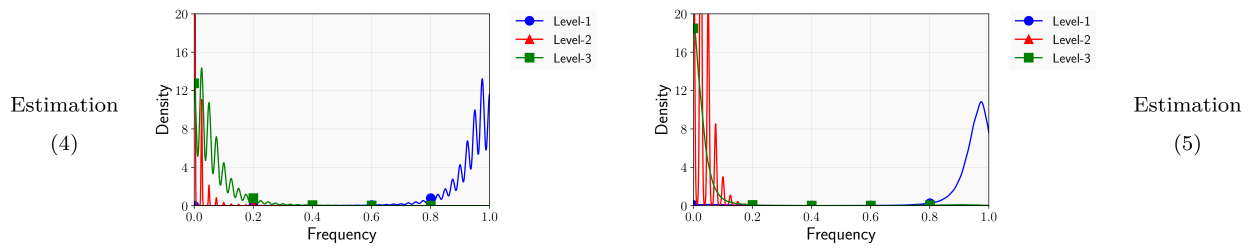


Figure 2.3. Simulated distribution of rule frequencies, periods 1-40

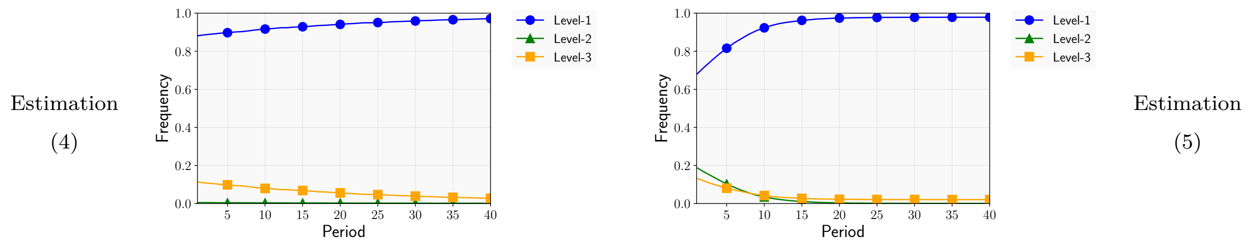


Figure 2.4. Simulated rule frequencies, periods 1-40

Another question of interest is whether subjects in different treatments had different belief learning and action-choice parameters. This can be determined by comparing the point estimates reported in estimation (1) with those reported in estimations (2) and (4). There are several significant differences in the action selection bias parameters and variance parameters, but the most interesting difference is the much faster mean logged rate of memory decay in the PvD treatment as evidenced by the $\mu_{\ln(\phi)}$ parameter. This difference decreases the adaptability of PvP subjects in favor of making their behavior more random. As noted before, randomness reduces the degree to which players can be exploited in repeated mixed strategy games, so there may exist an incentive for subjects to behave more noisily in a player-versus-player environment. Looking at Table 2.9 however, we see that the joint tests for differences between all of the free parameters in estimation (1) with their corresponding values in estimation (2) and (4) finds them to be jointly insignificant. This finding suggests that we should prefer the estimations that pool their estimates of these parameters across treatments (i.e., estimations (3) and (5)).

A final question of interest is whether the ACT-R extensions to weighted fictitious play learning informed subjects' level-1 beliefs. This can be determined by comparing the pattern recognition and probability weighting parameters in Table 2.8 to their nested values under the null hypotheses. Looking at the pattern recognition parameters first, we see from the non-zero values of β and ψ across most estimations that pattern recognition picked up some traction in the belief learning model. The values of these parameters indicate that new action associations were weighted more heavily than an agent's memory of his opponent's previous action. The non-zero values of the attentional weight $w_{\text{i}}(a_{\text{i}})$ suggest that subjects conditioned their beliefs on both their own and their opponent's previous actions, though $w_{\text{i}}(a_{\text{i}}) < .5$ implies that they weighted the evidence of their opponent's previous action more heavily. All three of these parameters are jointly significant in the level-1 estimations, but not in estimations (4) and (5). Turning now to the only free probability weighting parameter, we see that γ is significantly less than in one four of the five estimations. As noted before, $\gamma < 1$ indicates that subjects over-discriminated (under-discriminated) between the likelihoods of low/high (intermediate) probability actions.

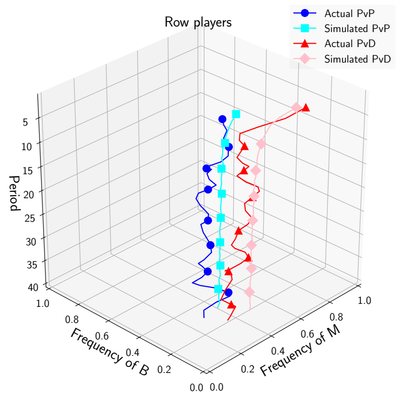
2.4.4 Simulations

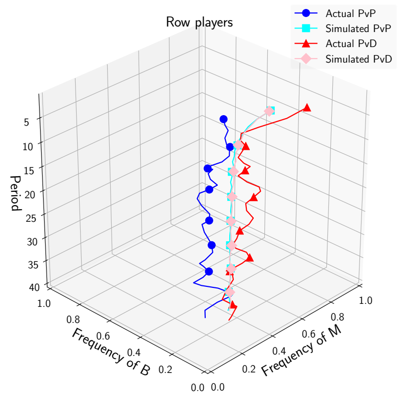
The purpose of this subsection is to asses model fit and see how well the level-k model explains the observed treatment differences. Figures 2.5 through 2.10 illustrate the fit of the simulated data from each estimation against the observed data from the actual experiment. All time series plots of the observed data are smoothed using a Gaussian filter with a standard deviation of one. 50,000 simulations were conducted within each player role to generate the simulated data. The simulated PvD subjects responded to the simulated PvP data so as to mirror the matching protocol of the actual experiment.

Figure 2.5 depicts the observed and simulated action frequencies in each one of the supergame's 40 periods. On this dimension the simulations do a decent job matching matching the actual data. From the right half of the figure, however, we see that estimations (3) and (5) both struggle to reproduce the observed treatment differences. These failures indicate that the differences are poorly explained by equilibrium dynamics and level-k reasoning respectively. Looking at the left half of the figure, we see that estimations (1) and (2) and estimations (1) and (4) provide a noticeably better fit to the data. This suggests that inter-treatment parameter heterogeneity is at least partially responsible for some of the observed

Estimations

(1) & (2)



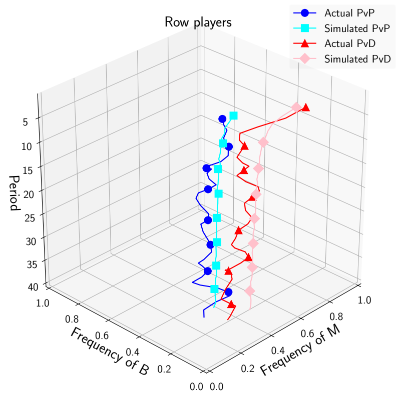


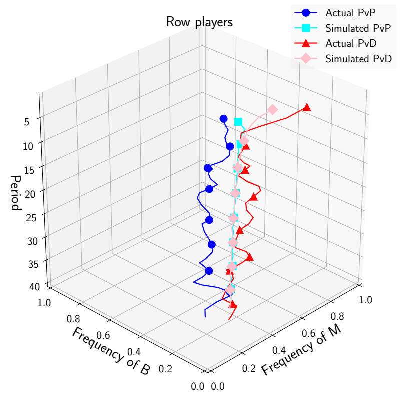
Estimation

(3)

Estimations

(1) & (4)



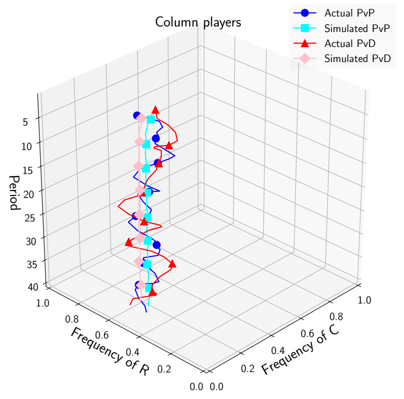


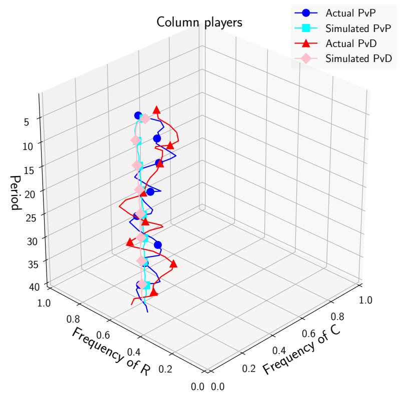
Estimation

(5)

Estimations

(1) & (2)



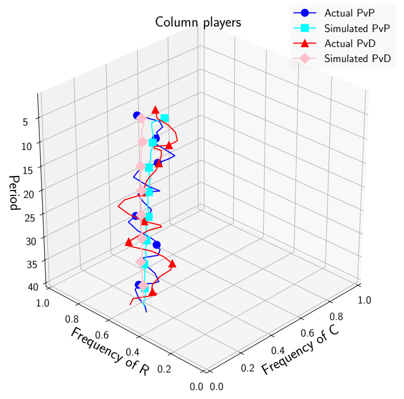


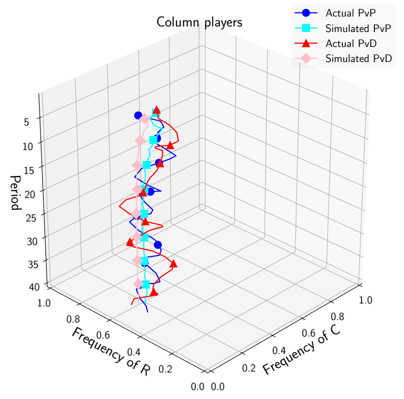
Estimation

(3)

Estimations

(1) & (4)





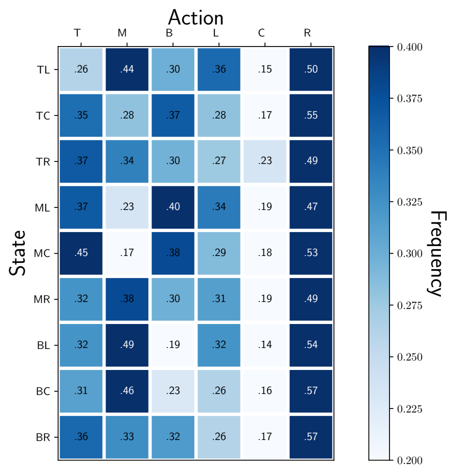
Estimation

(5)

Figure 2.5. Action frequencies, periods 1-40

Actual

PvP

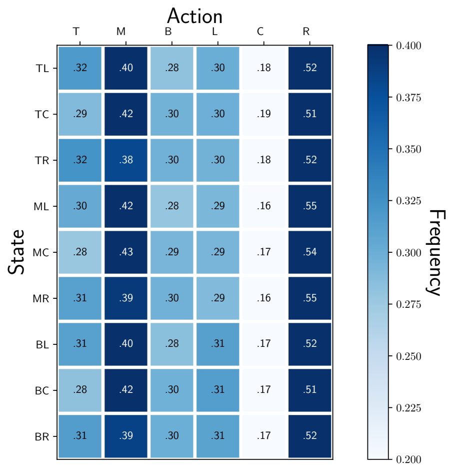


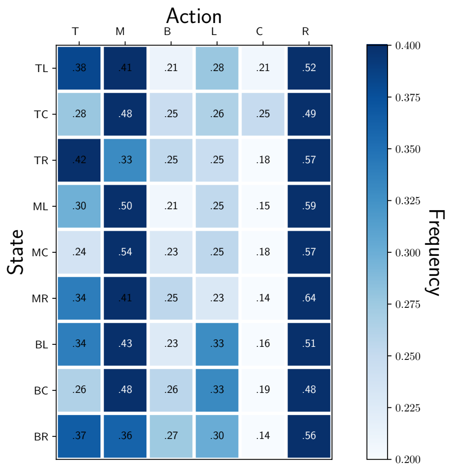
Actual
PvP

Simulated
PvP

Estimations

(1) & (2)





Simulated
PvP

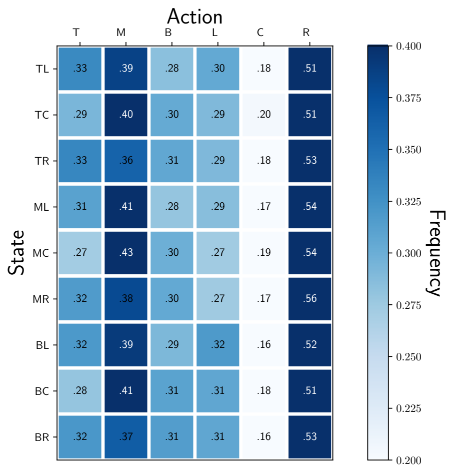
Estimation

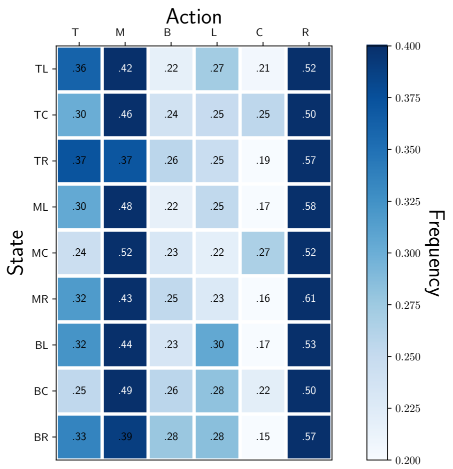
(3)

Simulated
PvP

Estimations

(1) & (4)





Simulated
PvP

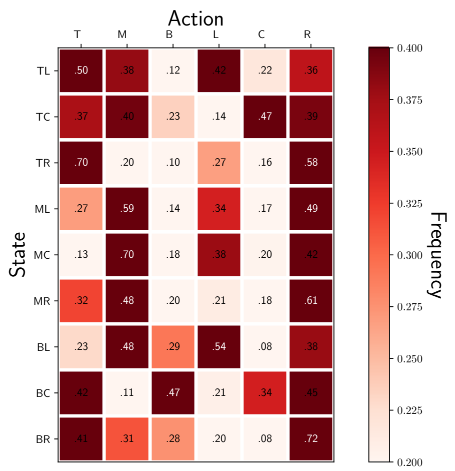
Estimation

(5)

Figure 2.6. PvP action frequencies by state, periods 2-40

Actual

PvD



State	T	M	B	L	C	R
TL	.50	.38	.12	.42	.22	.36
TC	.37	.40	.23	.14	.47	.39
TR	.70	.20	.10	.27	.16	.58
ML	.27	.59	.14	.34	.17	.49
MC	.13	.70	.18	.38	.20	.42
MR	.32	.48	.20	.21	.18	.61
BL	.23	.48	.29	.54	.08	.38
BC	.42	.11	.47	.21	.34	.45
BR	.41	.31	.28	.20	.08	.72

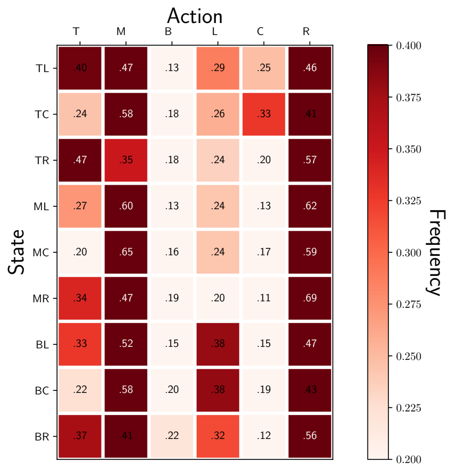
Actual

PvD

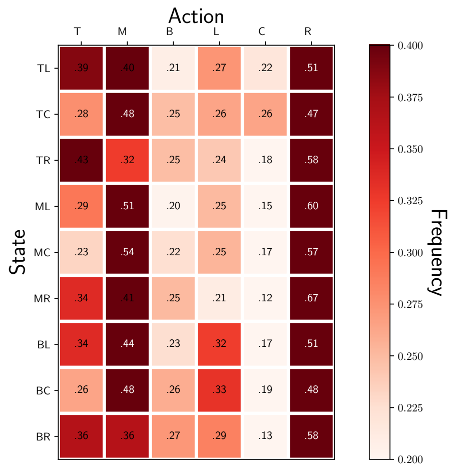
Simulated
PvD

Estimations

(1) & (2)



State	T	M	B	L	C	R
TL	.40	.47	.13	.29	.25	.46
TC	.24	.58	.18	.26	.33	.41
TR	.47	.35	.18	.24	.20	.57
ML	.27	.60	.13	.24	.13	.62
MC	.20	.65	.16	.24	.17	.59
MR	.34	.47	.19	.20	.11	.69
BL	.33	.52	.15	.38	.15	.47
BC	.22	.58	.20	.38	.19	.43
BR	.37	.41	.22	.32	.12	.56



State	T	M	B	L	C	R
TL	.39	.40	.21	.27	.22	.51
TC	.28	.48	.25	.26	.26	.47
TR	.43	.32	.25	.24	.18	.58
ML	.29	.51	.20	.25	.15	.60
MC	.23	.54	.22	.25	.17	.57
MR	.34	.41	.25	.21	.12	.67
BL	.34	.44	.23	.32	.17	.51
BC	.26	.48	.26	.33	.19	.48
BR	.36	.36	.27	.29	.13	.58

Simulated
PvD

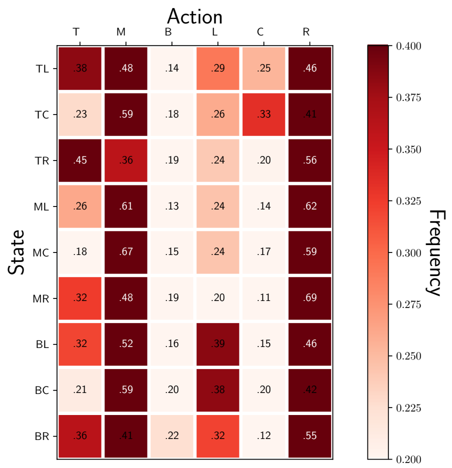
Estimation

(3)

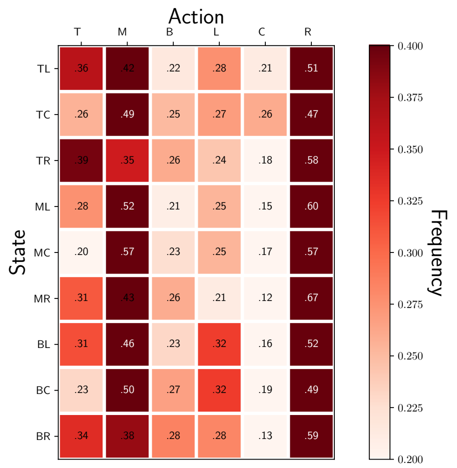
Simulated
PvD

Estimations

(1) & (4)



State	T	M	B	L	C	R
TL	.38	.48	.14	.29	.25	.46
TC	.23	.59	.18	.26	.33	.41
TR	.45	.36	.19	.24	.20	.56
ML	.26	.61	.13	.24	.14	.62
MC	.18	.67	.15	.24	.17	.59
MR	.32	.48	.19	.20	.11	.69
BL	.32	.52	.16	.39	.15	.46
BC	.21	.59	.20	.38	.20	.42
BR	.36	.41	.22	.32	.12	.55



State	T	M	B	L	C	R
TL	.36	.42	.22	.28	.21	.51
TC	.26	.49	.25	.27	.26	.47
TR	.39	.35	.26	.24	.18	.58
ML	.28	.52	.21	.25	.15	.60
MC	.20	.57	.23	.25	.17	.57
MR	.31	.43	.26	.21	.12	.67
BL	.31	.46	.23	.32	.16	.52
BC	.23	.50	.27	.32	.19	.49
BR	.34	.38	.28	.28	.13	.59

Simulated
PvD

Estimation

(5)

Figure 2.7. PvD action frequencies by state, periods 2-40

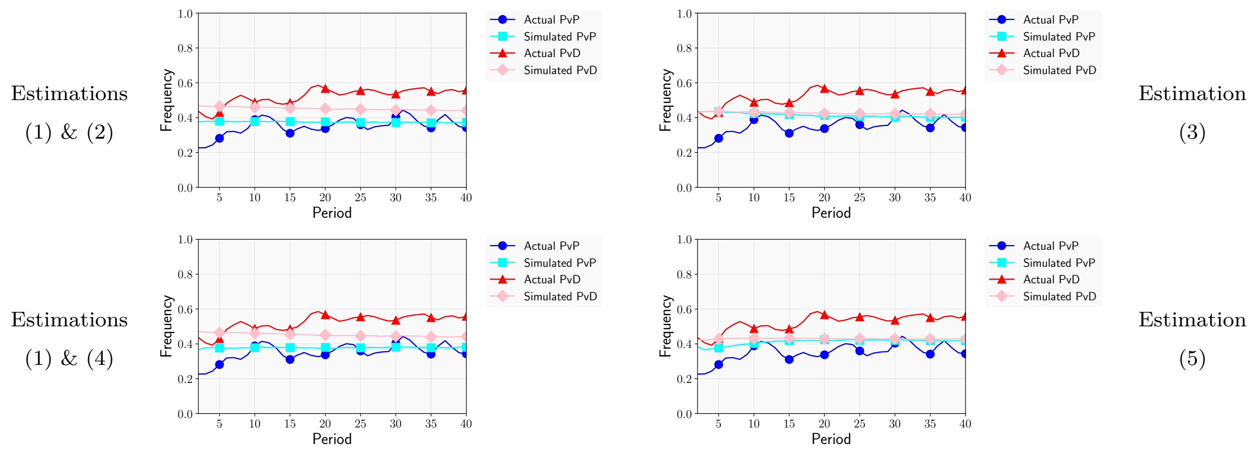


Figure 2.8. Inert action frequencies, periods 2-40

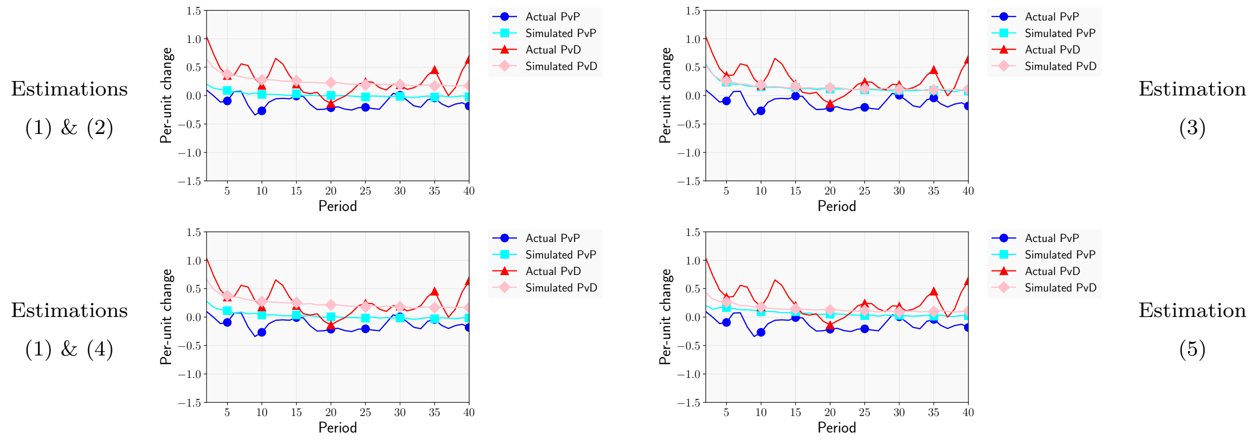


Figure 2.9. Per-unit change in inert action frequencies following a win, periods 2-40

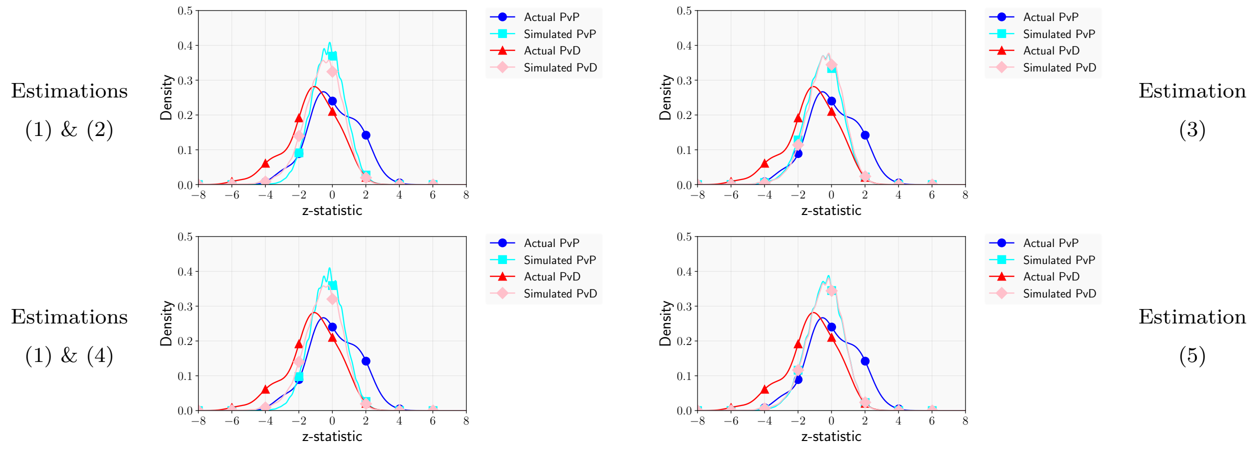


Figure 2.10. Distribution of z-statistics, periods 1-40

differences between the two treatments. It is also worth noting that estimations (2) and (4) provide a nearly identical fit to the observed PvP data.

Figures 2.6 and 2.7 compare the observed and simulated state-dependent action frequencies in periods 2-40 of the experiment. These frequencies are presented as heat maps for the reader to facilitate visual comparisons. In both figures, the simulations do a reasonably good job emulating the actual data. There are no noticeable differences between estimations in their ability to match the observed data from either treatment.

Figure 2.8 depicts the observed and simulated inert action frequencies in each one of the supergame's non-initial periods. Figure 2.9 shows how these inert action frequencies change in the period immediately following a win. From the right halves of these figures, we see that estimations (3) and (5) once again struggle to reproduce the observed treatment differences. This finding further suggests that the differences are poorly explained by equilibrium dynamics and level-k reasoning. Looking at the left halves of the figures, we see that estimations (1) and (2) and estimations (1) and (4) provide a marginally better fit to the data. Estimations (2) and (4) are nearly identical, however, in their fit to the observed data from the PvP treatment.

Finally, Figure 2.10 compares the observed and simulated distributions of the z-statistic obtained by conducting a runs test for randomness. As in Figures 5 and 6, there are no noticeable differences in the goodness of fit between the four sets of estimations. The simulated distributions all do a good job matching the modes of the actual data, but they understate the spread of the observed distributions, indicating that there may be some unmodeled dimensions of subject heterogeneity. This is to be expected though given the computational constraints that require us to use many pooled estimators.

2.4.5 Counterfactuals

To see whether subjects left any money on the table by stopping at the first level of reasoning, I simulate the earnings of each level-k rule using the model's estimated parameters. Figure 2.11 presents several time series of the average simulated earnings for each level-k rule against various types of level-k players. The graphs in the left column show their earnings

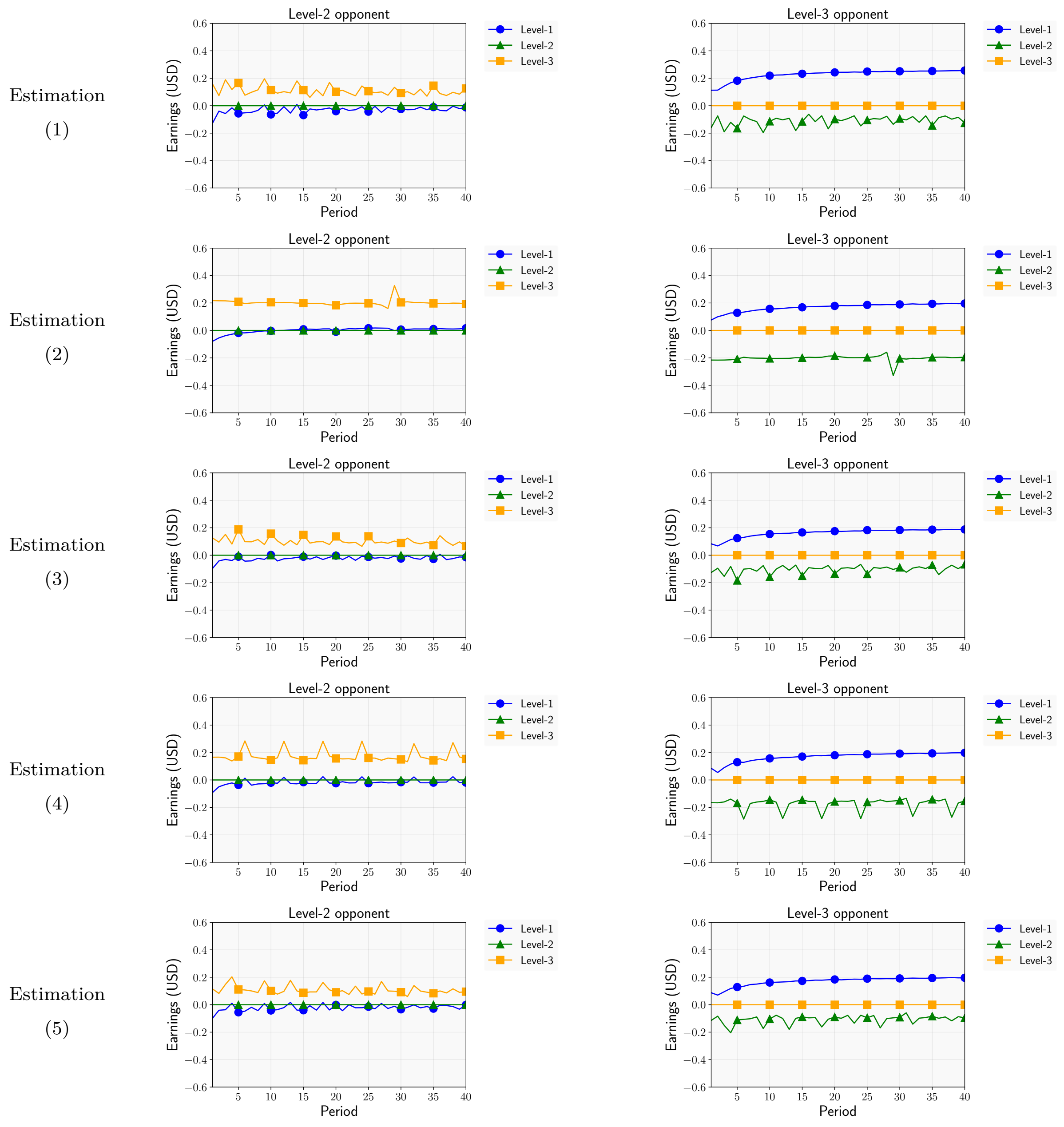


Figure 2.11. Average earnings by rule, periods 1-40

against a level-2 agent, while the graphs in the right column shows their earnings against a level-3 player. From the graphs on the left, we see that level-3 agents earn a large premium playing against level-2 players. Level-1 agents, conversely, suffer small losses initially but become too noisy to exploit after only a couple periods. From the graphs on the right, we see that level-1 agents start with a small earnings premium but gradually learn to more fully exploit the level-3 players. By the end of the supergame, they are extracting a rent of at least \$0.20 per period across all of the estimations. With these results in mind, it easy to understand why subjects learned to use level-1 reasoning: the level-1 rule is adaptive enough to exploit strategies that do not directly respond to it, yet noisy enough not to be exploited itself. Consequently, it is never too far behind the average earnings of any other rule that the law of effect is able to dominate the law of exercise.

2.5 Conclusion

This paper investigated the extent to which human agents used level-k reasoning in a repeated mixed strategy game. Towards this end, the model developed in Chapter 1 was estimated using data from a novel experiment with two between-subject treatments. In both treatments, subjects played a repeated version of a modified rock-paper-scissors game. The treatments varied the availability of information needed to use higher levels of reasoning. In one treatment, the information provided was sufficient to use any level of reasoning, while in the other treatment subjects were only provided with enough information to be level-1. A random effects model was estimated using the data from both treatments to identify the model's belief learning parameters.

Using the results of my estimation to simulate subjects' latent rule selections, I find that the initial proportion of level-k rules in the PvP treatment was 67.9 percent, 18.9 percent, and 13.3 percent for the level-1, level-2, and level-3 rules respectively. By the end of the supergame, however, level-1 rule use had risen to 97.9 percent and the rest of the rules had fallen to negligible levels. Over the course of the entire supergame, PvP subjects used level-1 reasoning to make over 93 percent of their decisions, so it should come as no surprise that the level-k model does a poor job of explaining most long-run treatment differences. To see whether subjects left any money on the table by stopping at the first level of reasoning, I simulate the earnings of each level-k rule using the model's estimated parameters. In contrast to several related works that study level-k reasoning in repeated pure strategy games (Stahl, 1999, 2000, 2001, and 2003; Gill and Prowse, 2016), I find that earnings could not be appreciably improved by reasoning beyond level-1. This was because level-1 behavior became too unpredictable to exploit after only a couple of periods.

Taken together, these results reject the chapter's motivating folk wisdom and validate the use of adaptive learning models in mixed strategy games. They also differ from a couple of key findings from the level-k learning literature, namely, that players learn to use and earn more money by using higher levels of reasoning (Stahl, 1996, 1999, 2000, 2001, and 2003; Danz, Fehr, and Kübler, 2012; Ho and Su, 2012 and 2020; Gill and Prowse, 2016). One explanation for these discrepancies is that level-1 agents may have weaker preferences between their actions in repeated mixed strategy games. This leads to noisier behavior which in turn lowers the incentive for players to engage in higher levels of reasoning. If this explanation is correct, then my results should generalize to other repeated mixed strategy games. Further investigation of level-k reasoning in these environments will be needed to confirm the scope of my findings.

REFERENCES

Alaoui, L., & Penta, A. (2022). Cost-Benefit Analysis in Reasoning. Journal of Political Economy, forthcoming.

Anderson, J. (1996). ACT: A simple theory of complex cognition. American Psychologist, 51(4), 355-365.

Anderson, J. (2007). How can the human mind occur in the physical universe? Oxford:Oxford University Press.

Anderson, J. R., Bothell, D., Lebiere, C., & Matessa, M. (1998). An Integrated Theory of List Memory. Journal of Memory and Language, 38(4), 341-380.

Anderson, J., & Lebiere, C. (1998).The Atomic Components of Thought. Routledge.

Brown, G.W. (1951). Iterative Solutions of Games by Fictitious Play. Activity Analysis of Production and Allocation.

Camerer, C., & Ho, T. H. (1999). Experience-weighted Attraction Learning in Normal Form Games. Econometrica, 67(4), 827-874.

Camerer, C. F., Ho, T.H., & Chong, J.K. (2002). Sophisticated experience-weighted attraction learning and strategic teaching in repeated games. Journal of Economic Theory, 104(1), 137–188.

Cason, T. N., Friedman, D., & Hopkins, E. (2010). Testing the TASP: An experimental investigation of learning in games with unstable equilibria. Journal of Economic Theory, 145(6), 2309–2331.

Cheung, Y., & Friedman, D. (1997). Individual Learning in Normal Form Games: Some Laboratory Results. Games and Economic Behavior, 19(1), 46-76.

Crawford, V. P., Costa-Gomes, M. A., & Iriberri, N. (2013). Structural Models of Nonequilibrium Strategic Thinking: Theory, Evidence, and Applications. Journal of Economic Literature, 51(1), 5-62.

Danz, D. N., Fehr, D., & Kübler, D. (2012). Information and beliefs in a repeated normal-form game. Experimental Economics, 15(4), 622-640.

Feng, J., & Wang, X. (2019). Dynamic level-K and cognitive hierarchy models of repeated normal-form games: A note. SSRN Electronic Journal.

Fischbacher, Urs. z-Tree: Zurich Toolbox for Ready-made Economic Experiments, Experimental Economics 10(2), 171-178.

Gill, D., & Prowse, V. L. 2016. Cognitive Ability, Character Skills, and Learning to Play Equilibrium: A Level-k Analysis. Journal of Political Economy.

Goldstein, W. M., & Einhorn, H. J. (1987). Expression theory and the preference reversal phenomena. Psychological Review, 94(2), 236-254.

Gonzalez, C., Lerch, J. F., & Lebiere, C. (2003). Instance-based learning in dynamic decision making. Cognitive Science, 27(4), 591-635.

Greiner, B. (2015). Subject Pool Recruitment Procedures: Organizing Experiments with ORSEE. Journal of the Economic Science Association 1 (1), 114-125.

Ho, T. H., Camerer, C. F., & Chong, J. (2007). Self-tuning experience weighted attraction learning in games. Journal of Economic Theory, 133(1), 177-198.

Ho, T., & Su, X. (2012). A Dynamic Level-k Model in Sequential Games. Management Science, 59(2), 452-469.

Ho, T., Park, S., & Su, X. (2020). A Bayesian Level-k Model in n-Person Games. Management Science.

Ioannou, C. A., & Romero, J. (2014). A generalized approach to belief learning in repeated games. Games and Economic Behavior, 87, 178-203.

Karmarkar, U. S. (1978). Subjectively weighted utility: A descriptive extension of the expected utility model. Organizational Behavior and Human Performance, 21(1), 61-72.

Karmarkar, U. S. (1979). Subjectively weighted utility and the Allais Paradox. Organizational Behavior and Human Performance, 24(1), 67-72.

Klöckner, A., Pinto, N., Lee, Y., Catanzaro, B., Ivanov, P., & Fasih, A. (2012) PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation, Parallel Computing, Volume 38, Issue 3, Pages 157-174.

Kraft, D. A software package for sequential quadratic programming. 1988. Tech. Rep. DFVLR-FB 88-28, DLR German Aerospace Center – Institute for Flight Mechanics, Koln, Germany.

Lattimore, P., Baker, J., & Witte, A. D. (1992). The Influence Of Probability on Risky Choice: A parametric Examination.

Lebiere, C., Wallach, D., & West, R. L. (2000). A memory-based account of the prisoner's dilemma and other 2x2 games. In Proceedings of International Conference on Cognitive Modeling, 185-193. NL: Universal Press.

Lindner, F., & Sutter, M. (2013). Level-k reasoning and time pressure in the 11-20 money request game. Economic Letters, 120, 542-545.

Mookherjee, D., & Sopher, B. (1994). Learning Behavior in an Experimental Matching Pennies Game. Games and Economic Behavior, 7(1), 62-91.

Nagel, R. (1995). Unraveling in Guessing Games: An Experimental Study. American Economic Review, 85, 1313-1326.

Nyarko, Y., & Schotter, A. (2002). An Experimental Study of Belief Learning Using Elicited Beliefs. Econometrica, 70(3), 971-1005.

Ochs, J. (1995). Games with Unique, Mixed Strategy Equilibria: An Experimental Study. Games and Economic Behavior, 10(1), 202-217.

Paszke, A., et al.(2019) Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems, 8026-8037

Rescorla, R.A. & Wagner, A.R. (1972) A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement, Classical Conditioning II, A.H. Black & W.F. Prokasy, Eds., pp. 64–99. Appleton-Century-Crofts.

Roth, A. E., & Erev, I. (1995). Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term. Games and Economic Behavior, 8(1), 164-212.

Rutström, E., & Wilcox, N. (2009). Stated beliefs versus inferred beliefs: A methodological inquiry and experimental test. Games and Economic Behavior, 67, 616-632.

Scott, D. (1992). "Multivariate Density Estimation: Theory, Practice, and Visualization", John Wiley & Sons, New York, Chicester.

Spiliopoulos, L. (2012). Pattern recognition and subjective belief learning in a repeated constant-sum game. Games and Economic Behavior, (75), 921-935.

Spiliopoulos, L. (2013a). Beyond fictitious play beliefs: Incorporating pattern recognition and similarity matching. Games and Economic Behavior, 81, 69-85.

Spiliopoulos, L. (2013b). Strategic adaptation of humans playing computer algorithms in a repeated constant-sum game. Auton Agent Multi-Agent Syst Autonomous Agents and Multi-Agent Systems, 27, 131-160.

Stahl, D. O. (1996). Boundedly Rational Rule Learning in a Guessing Game. Games and Economic Behavior, 16(2), 303-330.

Stahl, D. (1999). Evidence based rules and learning in symmetric normal-form games. International Journal of Game Theory, 28, 111-130.

Stahl, D. (2000). Rule Learning in Symmetric Normal-Form Games: Theory and Evidence. Games and Economic Behavior, 32, 105-138.

Stahl, D. (2001). Population rule learning in symmetric normal-form games: Theory and evidence. Journal of Economic Behavior & Organization, 45, 19-35.

Stahl, D. (2003). Action-Reinforcement Learning Versus Rule Learning. SSRN Electronic Journal SSRN Journal.

Stahl, D., & Wilson, P. W. (1994). Experimental evidence on players' models of other players. Journal of Economic Behavior & Organization, 25(3), 309-327

Stahl, D., & Wilson, P. W. (1995). On Players Models of Other Players: Theory and Experimental Evidence. Games and Economic Behavior, 10(1), 218-254.

Thomson, R., Bennati, S., & Lebiere, C. (2014). Extending theInfluence of Contextual Information in ACT-R using Buffer Decay. In Proceedings of the Conference of the Cognitive Science Society. Austin, TX: Cognitive Science Society

Thomson, R. & Lebiere, C. (2013). A Balanced Hebbian Algorithm for Associative Learning in ACT-R. In Proceedings of the International Conference on Cognitive Modeling.

Wales, D J, and Doye J P K, Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms. Journal of Physical Chemistry A, 1997, 101, 5111.

Weerd, H., Diepgrond, D., & Verbrugge, R. (2018). Estimating the use of higher-order theory of mind using computational agents. The B.E. Journal of Theoretical Economics, 18(2).

Wilcox, N. T. (2006). Theories of Learning in Games and Heterogeneity Bias. Econometrica, 74(5), 1271-1292.

A. CHAPTER 1 APPENDIX

A.1 ACT-R framing

ACT-R is comprised of a number of specialized modules that are responsible for the different components of cognition. There are multiple perceptual, manual, and control modules along with a centralized production system. Of the eight core modules that are included in the most recent version of ACT-R (see Figure A.1), only two require an in depth discussion for the development of my game-playing model: the procedural module, which houses the architecture's production system, and the declarative module, which houses declarative memory.

A.1.1 Procedural module

The procedural module coordinates the actions of all other modules through its production system. Productions are *if-then* statements that map the current state of the architecture into the next set of cognitive actions. The architecture's procedural knowledge is represented with utility values that measure the usefulness of each production rule. When two or more productions are in conflict (that is, their *if* conditions are all satisfied), the winner is selected probabilistically on the basis of their relative utilities. As in Chapter 1, rule selection is determined by a simple logit choice function. Utilities are also updated over time according to Chapter 1's rule learning mechanism. Since the production system is governed by the same set of equations as Chapter 1's rule learning model, we can simply frame the level-k rules as ACT-R productions to obtain our desired framing in ACT-R.

A.1.2 Declarative module

In ACT-R, declarative knowledge is represented with schema-like structures called "chunks". Each chunk is made up of one or more ordered slots, and each slot holds a single symbol. Symbols can be used to represent any information and may even be chunks themselves. As knowledge accumulates, new chunks are added to an agent's declarative memory. Whenever a chunk that is identical to an existing chunk (i.e., a chunk representing previously learned

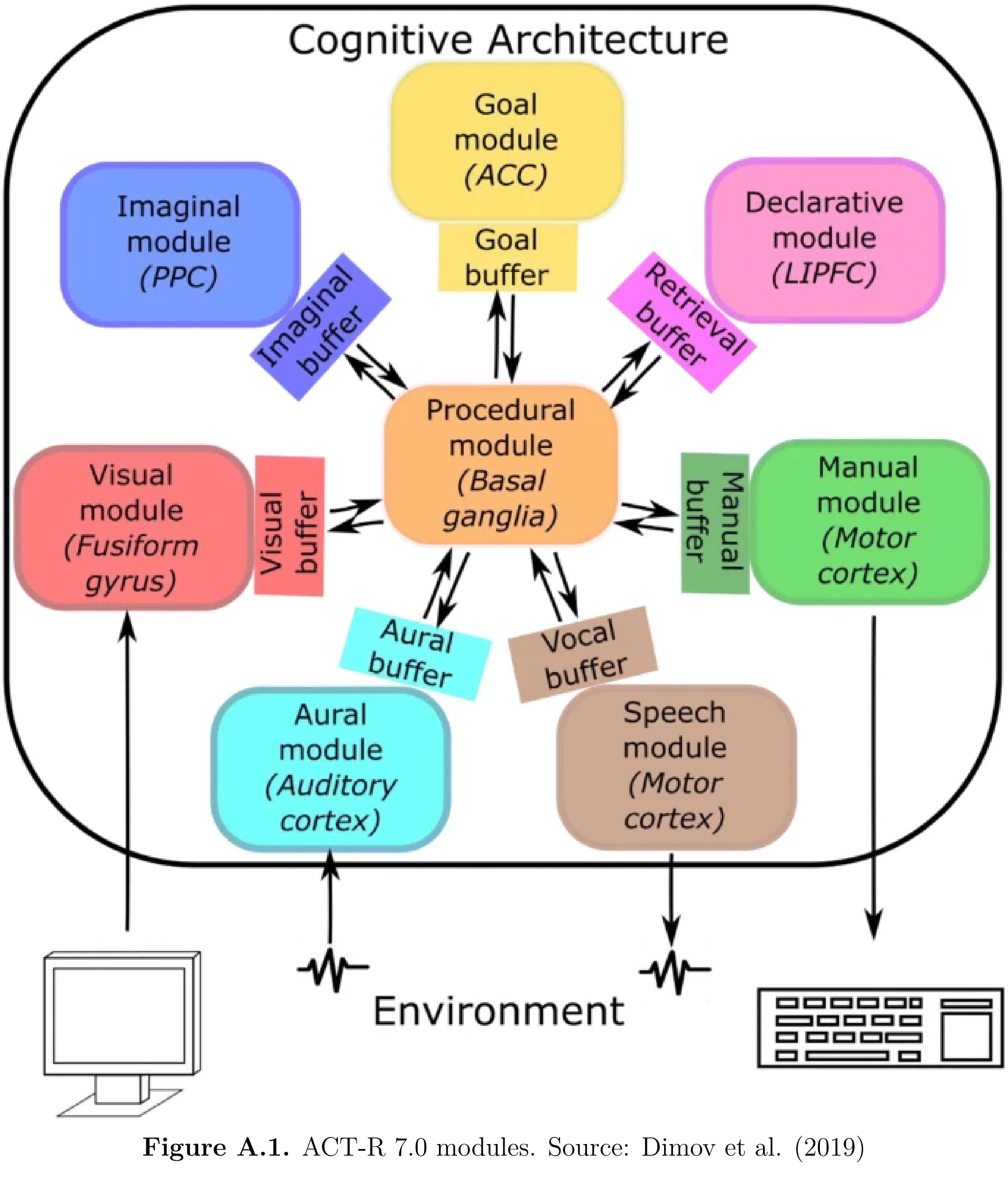

Cognitive Architecture
Goal module (ACC)
Goal buffer
Imaginal module (PPC)
Imaginal buffer
Declarative module (LIPFC)
Retrieval buffer
Procedural module (Basal ganglia)
Visual module (Fusiform gyrus)
Visual buffer
Manual buffer
Manual module (Motor cortex)
Aural buffer
Vocal buffer
Aural module (Auditory cortex)
Speech module (Motor cortex)
Environment

Figure A.1. ACT-R 7.0 modules. Source: Dimov et al. (2019)

information) tries to enter declarative memory, it is instead merged with the old chunk resulting in an increase in the chunk's level of activation. Activation measures how easily a chunk can be retrieved and is determined by the sum of two quantities: base activation and associative activation.

Base activation expresses the frequency and recency with which a chunk has appeared in declarative memory. This component is analogous to the weighted fictitious play part of Section 2.2's belief learning model. Before starting the supergame, agents add each one of their opponents' available actions to their declarative memory. This creates a total of $\sum_{\mathrm{j} \neq \mathrm{i} \in N} |A_{\mathrm{j}}|$ chunks to initialize the model. Each chunk begins with a base-level reinforcement of ξ, and thereafter receives a normalized reinforcement of one every time the opponent plays the chunk's corresponding action. The activation boost from these reinforcements slowly decays over time according to a power function.

Associative activation reflects the degree to which the need for a chunk can be predicted by the presence of contextual cues. This component is analogous to the pattern recognition part of Section 2.2's belief learning model. After observing his opponent's action, an agent associates the action's corresponding chunk with the actions each player selected in the previous period. Associations continue to be updated over time according to the previously discussed difference learning mechanism. Let $C_{\mathrm{i}}(a)$ denote the set of all player i chunks containing action $a \in A_{\mathrm{j}}$. Then the total activation of chunk $c \in C_{\mathrm{i}}(a)$ can be formally defined as

$$\mathbb{A}_{\mathrm{i}}^{t}(c) = \ln(\sum_{t=0}^{t-1} \mathbb{1}_{\mathrm{j}}^{t}(a)(t-t)^{-\phi}) + \psi \cdot \sum_{q \in Q^{t}} S_{\mathrm{i}}^{t}(a,q) \tag{A.1}$$

where $\ln(\sum_{t=0}^{t-1} \mathbb{1}_{\mathrm{j}}^{t}(a)(t-t)^{-\phi})$ and $\psi \cdot \sum_{q \in Q^{t}} S_{\mathrm{i}}^{t}(a,q)$ are the base-level and associative activations respectively.

To form their beliefs, agents create a composite chunk through a process known as "blending." Blending is a type of least squares interpolation that outputs an activation-weighted average of all the values held in each chunk's specified slot. In ACT-R, the weight assigned to each chunk is typically determined by a logit weighting function. Here I use a generalization of this function that adds the δ parameter:

$$w_{\mathrm{i}}^t(c) = \frac{\delta \cdot \exp(\gamma \cdot \mathbb{A}_{\mathrm{i}}^t(c))}{\delta \cdot \sum_{c \in C(a)} \exp(\gamma \cdot \mathbb{A}_{\mathrm{i}}^t(c)) + \sum_{c \in C(a) | a \neq a} \exp(\gamma \cdot \mathbb{A}_{\mathrm{i}}^t(c))} \tag{A.2}$$

Note that the actions stored in player i's chunks can be represented numerically with a series of action choice indicators. Blending these indicators is equivalent to summing the weights of the chunks that contain a given action. Then player i's belief that player j $\neq$ i $\in N$ will choose action $a \in A_{\mathrm{j}}$ in period t is $\mathfrak{b}_{\mathrm{i}}^t(a) = \sum_{c \in C(a)} w_{\mathrm{i}}^t(c)$.

Theorem A.1.1. *The ACT-R framing and the fictitious play framing of the belief learning model are equivalent, that is,* $\mathfrak{b}_{\mathrm{i}}^t(a) = b_{\mathrm{i}}^t(a)$.

Proof. First, note that $b_{\mathrm{i}}^t(a)$ can be rewritten as

$$b_{\mathrm{i}}^t(a) = \frac{\delta \cdot [\sum_{t=0}^{t-1} \mathbb{1}_{\mathrm{j}}^t(a)(t-t)^{-\phi} \cdot \exp(\psi \cdot \sum_{q \in Q^t} S_{\mathrm{i}}^t(a,q))]^\gamma}{\begin{array}{l}\delta \cdot [\sum_{t=0}^{t-1} \mathbb{1}_{\mathrm{j}}^t(a)(t-t)^{-\phi} \cdot \exp(\psi \cdot \sum_{q \in Q^t} S_{\mathrm{i}}^t(a,q))]^\gamma \\ \quad + \sum_{a \neq a \in A_{\mathrm{j}}} [\sum_{t=0}^{t-1} \mathbb{1}_{\mathrm{j}}^t(a)(t-t)^{-\phi} \cdot \exp(\psi \cdot \sum_{q \in Q^t} S_{\mathrm{i}}^t(a,q))]^\gamma\end{array}}$$

Next, substitute equation (A.1) in for $\mathbb{A}_{\mathrm{i}}^t(c)$ in equation (A.2) so that the expression for $w_{\mathrm{i}}^t(c)$ can be rewritten as

$$w_{\mathrm{i}}^t(c) = \frac{\delta \cdot \exp(\gamma \cdot (\ln(\sum_{t=0}^{t-1} \mathbb{1}_{\mathrm{j}}^t(a)(t-t)^{-\phi}) + \psi \cdot \sum_{q \in Q^t} S_{\mathrm{i}}^t(a,q)))}{\begin{array}{l}\delta \cdot \sum_{c \in C(a)} \exp(\gamma \cdot (\ln(\sum_{t=0}^{t-1} \mathbb{1}_{\mathrm{j}}^t(a)(t-t)^{-\phi}) + \psi \cdot \sum_{q \in Q^t} S_{\mathrm{i}}^t(a,q))) \\ \quad + \sum_{c \in C(a) | a \neq a \in A_{\mathrm{j}}} \exp(\gamma \cdot (\ln(\sum_{t=0}^{t-1} \mathbb{1}_{\mathrm{j}}^t(a)(t-t)^{-\phi}) + \psi \cdot \sum_{q \in Q^t} S_{\mathrm{i}}^t(a,q)))\end{array}}$$

$$= \frac{\delta \cdot [\sum_{t=0}^{t-1} \mathbb{1}_{\mathrm{j}}^t(a)(t-t)^{-\phi} \cdot \exp(\psi \cdot \sum_{q \in Q^t} S_{\mathrm{i}}^t(a,q))]^\gamma}{\begin{array}{l}\delta \cdot \sum_{c \in C(a)} [\sum_{t=0}^{t-1} \mathbb{1}_{\mathrm{j}}^t(a)(t-t)^{-\phi} \cdot \exp(\psi \cdot \sum_{q \in Q^t} S_{\mathrm{i}}^t(a,q))]^\gamma \\ \quad + \sum_{c \in C(a) | a \neq a \in A_{\mathrm{j}}} [\sum_{t=0}^{t-1} \mathbb{1}_{\mathrm{j}}^t(a)(t-t)^{-\phi} \cdot \exp(\psi \cdot \sum_{q \in Q^t} S_{\mathrm{i}}^t(a,q))]^\gamma\end{array}}$$

Finally, note that $\sum_{c \in C(a)} w_{\mathrm{i}}^t(c) = w_{\mathrm{i}}^t(c)$ and

$$w_{\mathrm{i}}^t(c) = \frac{\delta \cdot [\sum_{t=0}^{t-1} \mathbb{1}_{\mathrm{j}}^t(a)(t-t)^{-\phi} \cdot \exp(\psi \cdot \sum_{q \in Q^t} S_{\mathrm{i}}^t(a,q))]^\gamma}{\begin{array}{l}\delta \cdot [\sum_{t=0}^{t-1} \mathbb{1}_{\mathrm{j}}^t(a)(t-t)^{-\phi} \cdot \exp(\psi \cdot \sum_{q \in Q^t} S_{\mathrm{i}}^t(a,q))]^\gamma \\ \quad + \sum_{a \neq a \in A_{\mathrm{j}}} [\sum_{t=0}^{t-1} \mathbb{1}_{\mathrm{j}}^t(a)(t-t)^{-\phi} \cdot \exp(\psi \cdot \sum_{q \in Q^t} S_{\mathrm{i}}^t(a,q))]^\gamma\end{array}}$$

since $C_{\mathrm{i}}(a)$ is a singelton set in my model. Then $\mathfrak{b}_{\mathrm{i}}^t(a) = \sum_{c \in C(a)} w_{\mathrm{i}}^t(c) = w_{\mathrm{i}}^t(c) = b_{\mathrm{i}}^t(a)$.

■

B. CHAPTER 2 APPENDIX

B.1 Experimental instructions

PvP treatment

GENERAL INSTRUCTIONS

This is an experiment in the economics of strategic decision making. Purdue University has provided funds for this research. Everyone in today's experiment will earn at least $5. If you follow the instructions and make appropriate decisions, you can earn even more money. The currency used in today's experiment will be francs. Your francs will be converted to U.S. dollars at a rate of **10 francs to 1 U.S. dollar** at the end of the experiment. At the end of today's session, you will be paid in private and in cash.

It is important that you remain silent and do not look at other people's work during the experiment. If you have a question, or need assistance of any kind, please raise your hand and an experimenter will come to you. If you talk, laugh, exclaim out loud, etc., you will be asked to leave the session and you will not be paid. Additionally, we ask that you now turn off your cellphones and any other electronic devices so that there are no distractions during the experiment. We expect and appreciate your cooperation.

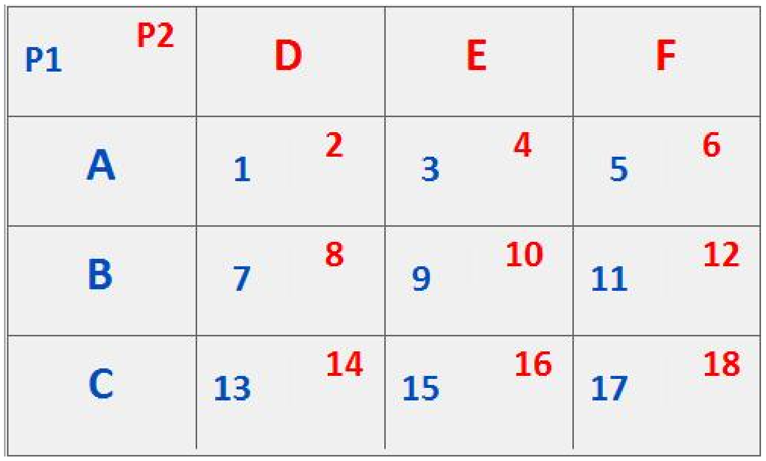
OVERVIEW

Today's experiment consists of **40** decision making periods. Before the start of the first period, you will be randomly and anonymously matched with another participant in this room. You will remain matched with the **same** participant for all 40 periods.

In each period you will be asked to select **one of three** available actions. The actions that you and the person you are matched with select will determine your earnings each period. The way that these actions determine each player's earnings is given by a payoff table that will be presented to you shortly. Before viewing the table, however, we will first look at some examples that illustrate how to read a payoff table.

In **figure 1** there is an example of a payoff table that is different from the one that will be used to determine your earnings. Player 1 (**P1**) can choose any of the actions in **blue**

(**A**, **B**, or **C**), and player 2 (**P2**) can choose any of the actions in **red** (**D**, **E**, or **F**). To find the payoffs that each player will earn from any action pair, you only need to look at the cell corresponding to the intersection of their two actions. Player 1's payoff is listed first in **blue**, and player 2's payoff is listed second in **red**.



P1 P2	D	E	F
A	1 2	3 4	5 6
B	7 8	9 10	11 12
C	13 14	15 16	17 18

Figure B.1. Sample payoff table

As an example, suppose player 1 chose action **A** and player 2 chose action **E**. In this case, player 1 would earn **3 francs** and player 2 would earn **4 francs** because 3 and 4 are the blue and red payoffs found at the intersection of row **A** and column **E**. Alternatively, suppose player 1 chose action **C** and player 2 chose action **D**. In this case, player 1 would earn **13 francs** and player 2 would earn **14 francs**.

At the beginning of today's session, you will receive **150 francs** for your participation in this experiment. Any gains (losses) that you acquire over the course of the experiment will be added to (subtracted from) this amount. In a few minutes, you will be given the actual payoff table that will be used to determine your earnings. Before viewing the table, however, let's first take a look at the interface you will use to select your actions in the experiment.

Each period begins on the screen depicted in **figure 2** below. Click the red button labeled "**Display**" to reveal the additional information shown in **figure 3**. The payoff table will be displayed in the box labeled "**a**" located in the upper-left hand corner. In box "**b**" you will find the player role (either **P1** or **P2**) that you will have for all 40 periods. In box "**c**" you will see the actions that you and the person you are matched with picked in the previous period and each of your earnings in that period. Below that in box "**d**" there will be three radio buttons with labels that correspond to your three possible actions. Click the button next to the appropriate label to indicate your action choice. Once you have selected your action, you may submit your choice by clicking the red button labeled "**Submit Action**" found in box "**e**". This button will not appear until you select one of the radio buttons. After you and the person you are matched with have both submitted your actions, the next period will begin. Note however that the program will not advance to the next period until at least 30 seconds have passed in the current period. You may take as much time as you need to select your actions choices.

Once you have completed all 40 periods, a gray button labeled "**Continue**" will appear in the bottom right hand corner of your screen. **Do not** click this button, but instead please wait quietly for everyone to finish with the experiment. No use of cellphones or any other electronic devices will be permitted during this time.

QUIZ

You will now be asked to answer 2 short quiz questions to test you understanding of the experiment. The experiment will not begin until everyone has answered both of the quiz questions correctly. Everyone will answer the same 2 questions.

If there are no further questions, you will now be given **2 minutes** to study the payoff table that will be used to determine your earnings. Your role (either **P1** or **P2**) will be listed below the payoff table. The participant you are matched with will be assigned to the other role. As a reminder, you will be matched with the **same** participant for all 40 periods. After you have had 2 minutes to study the payoff table, the experiment will begin.

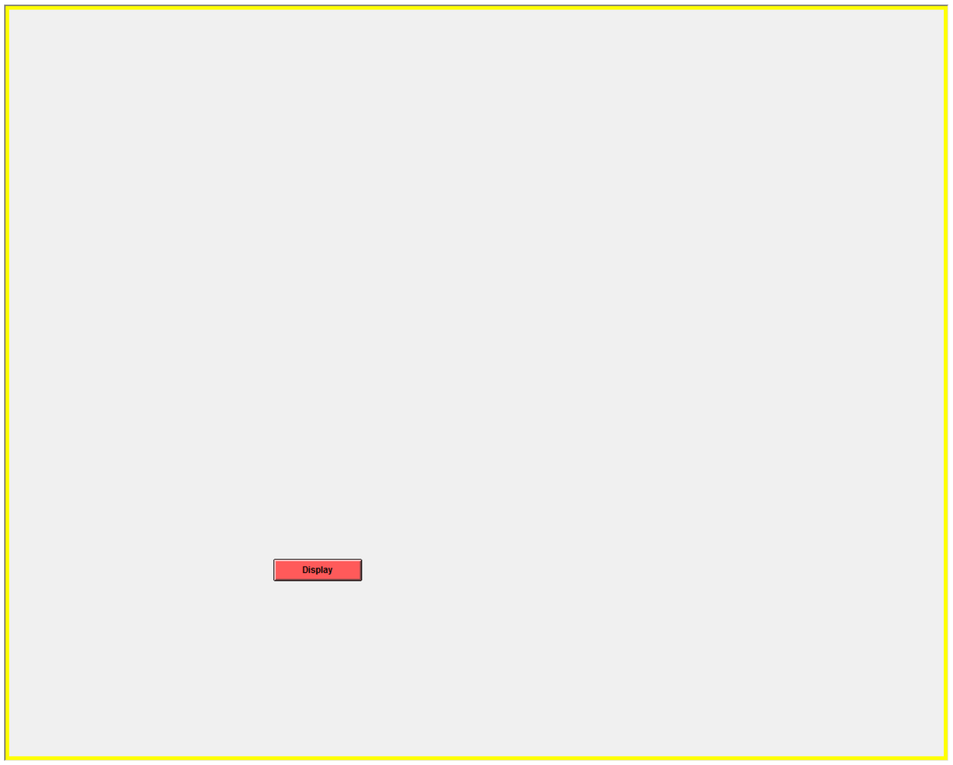
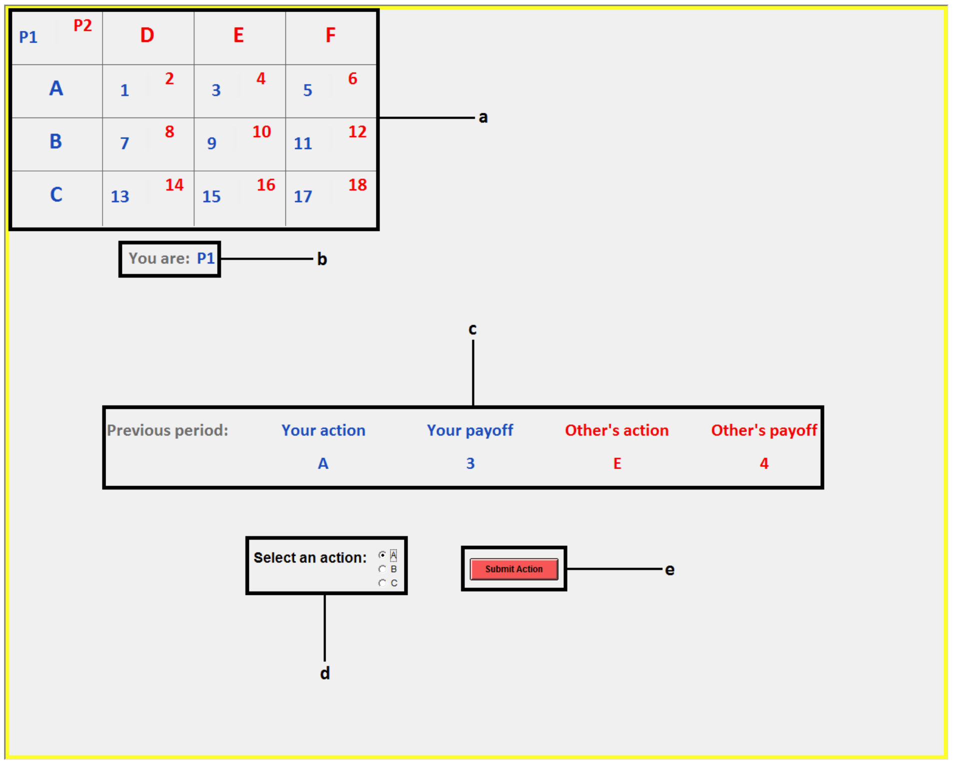


Figure B.2. Sample screen



P1 P2	D	E	F
A	1 2	3 4	5 6
B	7 8	9 10	11 12
C	13 14	15 16	17 18

You are: P1

Previous period:	Your action	Your payoff	Other's action	Other's payoff
	A	3	E	4

Select an action: A B C

Figure B.3. Sample screen

PvD treatment

GENERAL INSTRUCTIONS

This is an experiment in the economics of strategic decision making. Purdue University has provided funds for this research. Everyone in today's experiment will earn at least $5. If you follow the instructions and make appropriate decisions, you can earn even more money. The currency used in today's experiment will be francs. Your francs will be converted to U.S. dollars at a rate of **10 francs to 1 U.S. dollar** at the end of the experiment. At the end of today's session, you will be paid in private and in cash.

It is important that you remain silent and do not look at other people's work during the experiment. If you have a question, or need assistance of any kind, please raise your hand and an experimenter will come to you. If you talk, laugh, exclaim out loud, etc., you will be asked to leave the session and you will not be paid. Additionally, we ask that you now turn off your cellphones and any other electronic devices so that there are no distractions during the experiment. We expect and appreciate your cooperation.

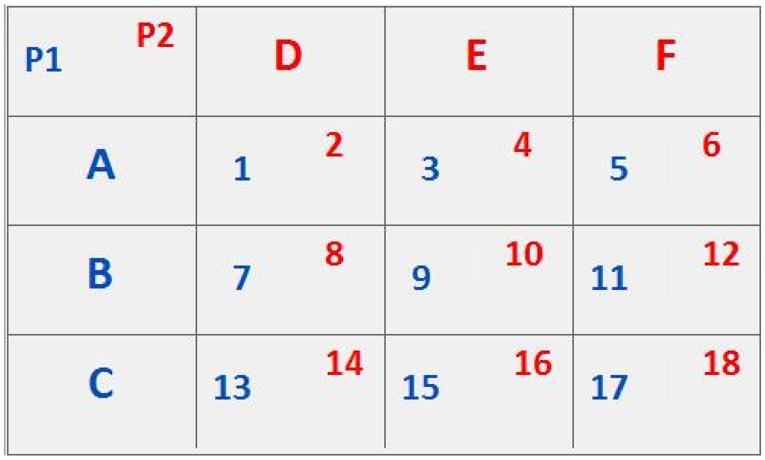
OVERVIEW

Today's experiment consists of **40** decision making periods. Before the start of the first period, you will be matched with a sequence of actions that were chosen by a participant in a **previous session**. The participant whose actions you will be matched with has been selected at random. Everyone in today's session will be matched with the actions of a different participant. You will remain matched with the **same** participant's sequence of actions for all 40 periods.

In each period you will be asked to select **one of three** available actions. The actions that you select and the actions given by the sequence you are matched with will determine your earnings each period. The way that these actions determine your earnings is given by a payoff table that will be presented to you shortly, but first we will discuss in greater detail the nature of the previous session.

The previous participant whose actions you will be matched with made his/her choices in an experiment that was nearly identical to the experiment being conducted today, with the main difference being that he/she was matched with another participant in the same session rather than with a sequence of actions that were chosen by a participant in a previous session. The participant whose actions you are matched with was not informed that his/her actions would be used in another experiment. The choices you make today will have no impact on the earnings of anyone other than yourself. In a few minutes, you will be given the actual payoff table that will be used to determine your earnings. Before viewing the table, however, let's first take a look at some examples that illustrate how to read a payoff table.

In **figure 1** there is an example of a payoff table that is different from the one that will be used to determine your earnings. Player 1 (**P1**) can choose any of the actions in **blue** (**A**, **B**, or **C**), and player 2 (**P2**) can choose any of the actions in **red** (**D**, **E**, or **F**). To find the payoffs that each player will earn from any action pair, you only need to look at the cell corresponding to the intersection of their two actions. Player 1's payoff is listed first in **blue**, and player 2's payoff is listed second in **red**.



P1 P2	D	E	F
A	1 2	3 4	5 6
B	7 8	9 10	11 12
C	13 14	15 16	17 18

Figure B.1. Sample payoff table

As an example, suppose player 1 chose action **A** and player 2 chose action **E**. In this case, player 1 would earn **3 francs** and player 2 would earn **4 francs** because 3 and 4 are the blue and red payoffs found at the intersection of row **A** and column **E**. Alternatively, suppose player 1 chose action **C** and player 2 chose action **D**. In this case, player 1 would earn **13 francs** and player 2 would earn **14 francs**.

At the beginning of today's session, you will receive **150 francs** for your participation in this experiment. Any gains (losses) that you acquire over the course of the experiment will be added to (subtracted from) this amount. In a few minutes, you will be given the actual payoff table that will be used to determine your earnings. You will only be able to see your own payoffs in this payoff table. These payoffs are the same payoffs that were given to the previous participants who shared your player role, but they were also able to see the other player's payoffs in the payoff table. Before viewing the table, let's first take a look at the interface you will use to select your actions in the experiment.

Each period begins on the screen depicted in **figure 2** below. Click the red button labeled "**Display**" to reveal the additional information shown in **figure 3**. The payoff table will be displayed in the box labeled "**a**" located in the upper-left hand corner. In box "**b**" you will find the player role (either **P1** or **P2**) that you will have for all 40 periods. In box "**c**" you will see the actions that you and the sequence of actions you are matched with picked in the previous period and your earnings in that period. Below that in box "**d**" there will be three radio buttons with labels that correspond to your three possible actions. Click the button next to the appropriate label to indicate your action choice. Once you have selected your action, you may submit your choice by clicking the red button labeled "**Submit Action**" found in box "**e**". This button will not appear until you select one of the radio buttons. After you have submitted your action, the next period will begin. Note however that the program will not advance to the next period until at least 30 seconds have passed in the current period. You may take as much time as you need to select your actions choices.

Once you have completed all 40 periods, a gray button labeled "**Continue**" will appear in the bottom right hand corner of your screen. **Do not** click this button, but instead please wait quietly for everyone to finish with the experiment. No use of cellphones or any other electronic devices will be permitted during this time.

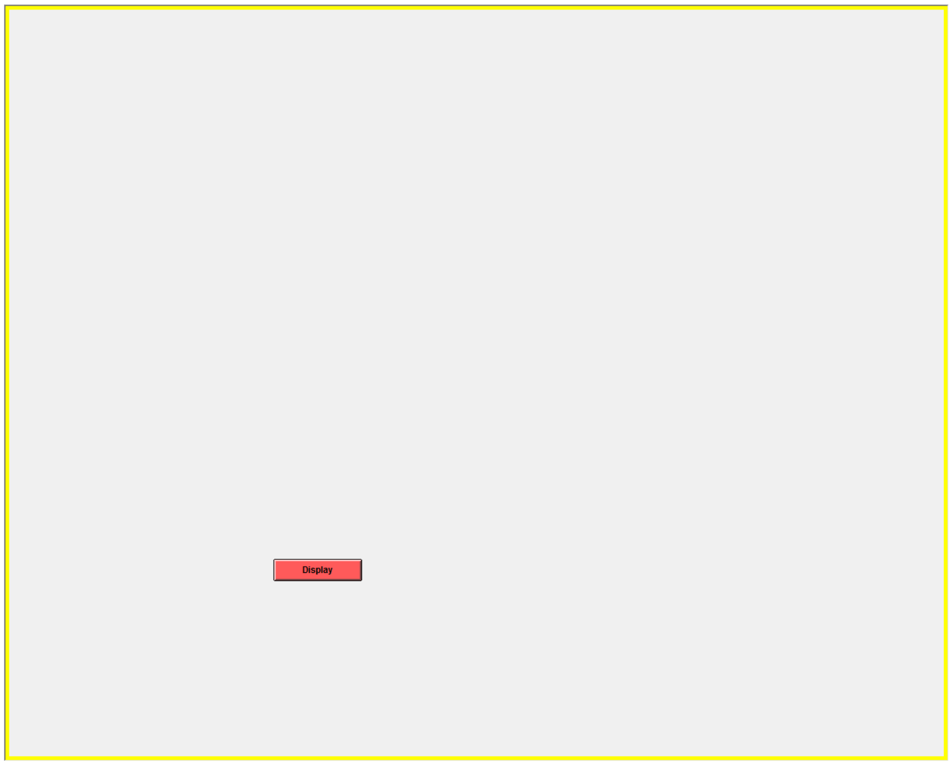
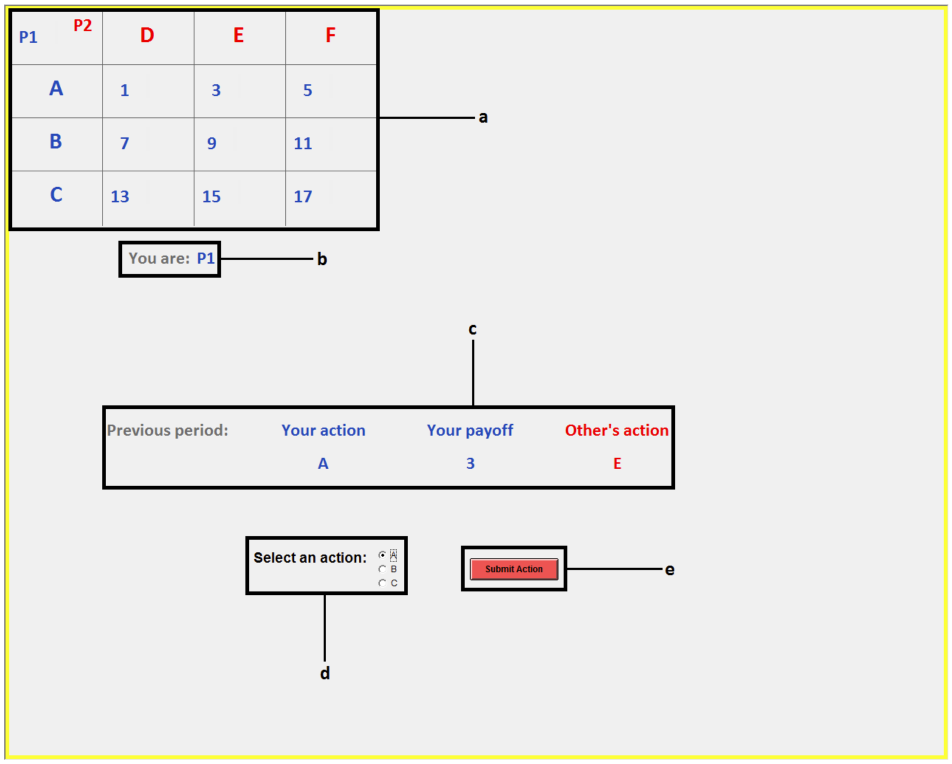


Figure B.2. Sample screen



P1 P2	D	E	F
A	1	3	5
B	7	9	11
C	13	15	17

You are: P1

Previous period:	Your action	Your payoff	Other's action
	A	3	E

Select an action: [✓ A] B C

Figure B.3. Sample screen

QUIZ

You will now be asked to answer 2 short quiz questions to test you understanding of the experiment. The experiment will not begin until everyone has answered both of the quiz questions correctly. Everyone will answer the same 2 questions.

If there are no further questions, you will now be given **2 minutes** to study the payoff table that will be used to determine your earnings. Your role (either **P1** or **P2**) will be listed below the payoff table. The previous participant whose sequence of actions you are matched with was assigned to the other role. As a reminder, you will remain matched with the **same** participant's sequence of actions for all 40 periods. After you have had 2 minutes to study the payoff table, the experiment will begin.

B.2 Supplementary tables and figures

Table B.1. Action frequencies by state, periods 2-40. Previous period's action in gray.

		Row players										Column players						
		Nash		PvP		PvD		Nash			Nash		PvP		PvD		Nash	
	T	.447	$\overset{***}{>}$	.262	$\overset{**}{<}$	.500	$>$	.447		L	.287	$\overset{*}{<}$	.355	$<$	.425	$\overset{***}{>}$	.287	
				(.051)		(.072)							(.034)		(.045)			
State TL (Win)	M	.266	$\overset{***}{<}$	.439	$>$	.381	$\overset{*}{>}$	.266		C	.266	$\overset{***}{>}$	.150	$<$	.217	$<$	.266	State TL (Loss)
				(.045)		(.060)							(.032)		(.034)			
	B	.287	$<$	.299	$\overset{***}{>}$	.119	$\overset{***}{<}$	.287		R	.447	$<$	.495	$>$	.358	$<$	.447	
				(.056)		(.035)							(.052)		(.057)			
	T	.447	$>$	.352	$<$	.370	$<$	.447		L	.287	$<$	.342	$<$	.344	$>$	.287	
				(.066)		(.066)							(.044)		(.058)			
State TC (Loss)	M	.266	$<$	.282	$<$	.395	$\overset{**}{>}$	.266		C	.266	$\overset{**}{>}$	.192	$>$	.167	$\overset{*}{<}$	.266	State ML (Loss)
				(.063)		(.056)							(.031)		(.048)			
	B	.287	$<$	.366	$>$	.235	$<$	.287		R	.447	$<$	.467	$<$	.490	$>$	.447	
				(.067)		(.051)							(.047)		(.050)			
	T	.447	$\overset{*}{>}$	.368	$\overset{***}{<}$	.705	$\overset{***}{>}$	.447		L	.287	$<$	.323	$\overset{**}{<}$	.537	$\overset{**}{>}$	.287	
				(.041)		(.054)							(.056)		(.091)			
State TR (Win)	M	.266	$\overset{*}{<}$	.337	$\overset{**}{>}$	.198	$<$	.266		C	.266	$\overset{***}{>}$	.135	$>$	.083	$\overset{**}{<}$	.266	State BL (Win)
				(.034)		(.040)							(.030)		(.036)			
	B	.287	$<$	.295	$\overset{***}{>}$	.097	$\overset{***}{<}$	.287		R	.447	$\overset{*}{<}$	.542	$\overset{*}{>}$	.380	$<$	.447	
				(.041)		(.027)							(.055)		(.082)			
	T	.447	$\overset{*}{>}$	.367	$\overset{*}{>}$	.268	$\overset{**}{<}$	.447		L	.287	$>$	.282	$\overset{**}{>}$	.135	$\overset{**}{<}$	.287	
				(.041)		(.054)							(.065)		(.049)			
State ML (Win)	M	.266	$>$	.233	$\overset{***}{<}$	.591	$\overset{***}{>}$	.266		C	.266	$>$	.169	$\overset{***}{<}$	.473	$\overset{*}{>}$	.266	State TC (Win)
				(.057)		(.065)							(.051)		(.114)			
	B	.287	$\overset{**}{<}$	.400	$\overset{***}{>}$	.141	$\overset{**}{<}$	.287		R	.447	$\overset{*}{<}$	.549	$>$	.392	$<$	.447	
				(.042)		(.038)							(.061)		(.097)			
	T	.447	$<$	.455	$\overset{***}{>}$	.127	$\overset{***}{<}$	.447		L	.287	$<$	.288	$<$	.378	$>$	.287	
				(.072)		(.043)							(.053)		(.073)			
State MC (Win)	M	.266	$>$	.167	$\overset{***}{<}$	.696	$\overset{***}{>}$	.266		C	.266	$>$	.182	$<$	.203	$<$	.266	State MC (Loss)
				(.056)		(.088)							(.050)		(.046)			
	B	.287	$<$	.379	$\overset{**}{>}$	.177	$<$	.287		R	.447	$<$	.530	$>$	.419	$<$	.447	
				(.077)		(.062)							(.085)		(.063)			
	T	.447	$\overset{***}{>}$	.323	$>$	.320	$\overset{**}{<}$	.447		L	.287	$>$	.262	$>$	.207	$<$	.287	
				(.032)		(.044)							(.057)		(.070)			
State MR (Loss)	M	.266	$\overset{***}{<}$	.379	$<$	.484	$\overset{***}{>}$	.266		C	.266	$>$	.164	$\overset{*}{<}$	.345	$>$	.266	State BC (Loss)
				(.035)		(.059)							(.053)		(.074)			
	B	.287	$<$	.298	$\overset{**}{>}$	.196	$\overset{**}{<}$	.287		R	.447	$\overset{*}{<}$	.574	$>$	.448	$>$	.447	
				(.041)		(.036)							(.071)		(.062)			
	T	.447	$\overset{**}{>}$	.323	$>$	.229	$\overset{**}{<}$	.447		L	.287	$>$	.275	$>$	.267	$<$	.287	
				(.046)		(.070)							(.037)		(.036)			
State BL (Loss)	M	.266	$\overset{***}{<}$	.490	$>$	.479	$\overset{**}{>}$	.266		C	.266	$>$	.233	$>$	.157	$\overset{**}{<}$	.266	State TR (Loss)
				(.050)		(.078)							(.032)		(.032)			
	B	.287	$\overset{**}{>}$	.188	$<$	.292	$>$	.287		R	.447	$<$	.492	$<$	.576	$\overset{**}{>}$	.447	
				(.039)		(.083)							(.033)		(.047)			
	T	.447	$\overset{**}{>}$	.311	$<$	.421	$<$	.447		L	.287	$<$	.313	$\overset{*}{>}$	.210	$<$	.287	
				(.056)		(.106)							(.038)		(.050)			
State BC (Win)	M	.266	$\overset{***}{<}$	.459	$\overset{***}{>}$	.105	$<$	.266		C	.266	$\overset{*}{>}$	.192	$>$	.182	$<$	.266	State MR (Win)
				(.048)		(.059)							(.031)		(.056)			
	B	.287	$>$	.230	$\overset{*}{<}$	.474	$\overset{*}{>}$	.287		R	.447	$<$	.495	$<$	.607	$\overset{**}{>}$	.447	
				(.053)		(.114)							(.047)		(.072)			
	T	.447	$\overset{**}{>}$	.356	$<$	.413	$<$	.447		L	.287	$>$	.261	$>$	.199	$\overset{*}{<}$	.287	
				(.040)		(.044)							(.036)		(.040)			
State BR (Win)	M	.266	$<$	.328	$>$	.312	$>$	.266		C	.266	$\overset{*}{>}$	.167	$\overset{***}{>}$	.082	$\overset{***}{<}$	.266	State BR (Loss)
				(.044)		(.053)							(.037)		(.027)			
	B	.287	$<$	.317	$>$	.275	$<$	.287		R	.447	$\overset{***}{<}$	.572	$\overset{**}{<}$	.719	$\overset{***}{>}$	.447	
				(.038)		(.050)							(.043)		(.043)			

Bootstrap std. errors in parentheses. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

Table B.2. Median response times by chosen action, periods 2-40

	Row players				Column Players		
	PvP		PvD		PvP		PvD
T	12.0	$\overset{***}{>}$	6.5	L	12.1	$\overset{***}{>}$	7.8
	(1.0)		(.7)		(1.4)		(.9)
M	13.7	$\overset{***}{>}$	6.7	C	11.9	$\overset{***}{>}$	6.4
	(1.1)		(.6)		(1.2)		(.8)
B	12.8	$\overset{***}{>}$	8.8	R	11.2	$\overset{***}{>}$	5.9
	(1.0)		(1.0)		(.9)		(.5)
All	12.8	$\overset{***}{>}$	6.9	All	11.5	$\overset{***}{>}$	6.5
	(1.0)		(.5)		(1.0)		(.6)

Bootstrap std. errors in parentheses. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

Table B.3. Avg. earnings gain from best response to opponent's empirical action distribution, periods 1-40

			PvP		PvD		
Row players	0	$\overset{***}{<}$	2.71	$>$	2.37	$\overset{***}{>}$	0
			(.35)		(.38)		
Column players	0	$\overset{***}{<}$	2.24	$>$	1.77	$\overset{***}{>}$	0
			(.29)		(.44)		
All players	0	$\overset{***}{<}$	2.48	$>$	2.07	$\overset{***}{>}$	0
			(.17)		(.26)		

Bootstrap std. errors in parentheses. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

B.3 Estimation code

B.3.1 estimate.py

```
import numpy as np
import pandas as pd
import torch
import time
import mylib as my
from scipy import special, optimize

# Basic settings
treat = ('pvp', 'pvd')
dat_file = ('wagner_pvp.csv', 'wagner_pvd.csv')  # Data files
mat_file = ('wagner_mat.csv', 'wagner_mat.csv')  # Matrix files
num_sub = (56, 56)  # Number of subjects
num_sim = (4000, 1000)  # Number of simulations
num_per = (40, 40)  # Number of periods
num_act = (3, 3)  # Number of actions
num_rule = (3, 1)  # Number of rules
opt = {}  # Options
opt['p_k'] = 'smax'  # Rule selection
opt['p_ak'] = 'smax'  # Action selection
opt['U'] = 'diff'  # Rule learning
opt['v'] = 'lin'  # Declarative utility
opt['w'] = 'ratio'  # Probability weighting
opt['b_1'] = 'smax'  # Action blending
opt['b_k'] = 'hmax'  # Lower-level response
opt['B'] = 'pow'  # Base-level forgetting
opt['S'] = 'spread'  # Associative memory
opt['P'] = 'none'  # Similarity matching
opt['R_a'] = opt['v']  # Procedural utility
opt['R_c'] = 'belief'  # Base-level learning
opt['S_q'] = 'diff'  # Associative learning
est = {}  # Estimators
est['alpha'] = 'pool'  # Rule learning, rate parameter
```

```
est['beta'] = 'pool'  # Associative learning, rate parameter
est['gamma'] = 'pool'  # Probability weighting, curvature parameter
est['delta'] = None  # Probability weighting, elevation parameter
est['iota'] = None  # Lower-level response, sensitivity parameter
est['kappa'] = 'pool'  # Rule selection, sensitivity parameter
est['lam'] = 'rand'  # Action selection, sensitivity  parameter
est['nu_M'] = 'pool'  # Action selection, bias parameter
est['nu_B'] = 'pool'  # Action selection, bias parameter
est['nu_C'] = 'pool'  # Action selection, bias parameter
est['nu_R'] = 'pool'  # Action selection, bias parameter
est['xi'] = None  # Similarity matching, penalty parameter
est['pi'] = None  # Base-level learning, prior parameter
est['rho'] = None  # Base-level learning, reinforcement parameter
est['upsilon1'] = 'pool'  # Rule learning, prior parameter
est['ups_star2'] = 'rand'  # Rule learning, prior parameter
est['ups_star3'] = 'rand'  # Rule learning, prior parameter
est['phi'] = 'rand'  # Base-level forgetting, rate parameter
est['psi'] = 'pool'  # Associative learning, strength parameter
est['omega'] = 'pool'  # AL & SM, saliency parameter

# Advanced settings
seed = 7  # Random seed
cuda = True  # GPU acceleration
scale = 1e-2
tol = 1e-6 * scale  # Minimizer tolerance
temp = 1. * scale  # Basin-hopping temperature
step = .1  # Basin-hopping stepsize
niter = 1000  # Number of basin-hopping iterations
dist = {}  # RE distributions
dist['alpha'] = 'logit_norm'
dist['beta'] = 'logit_norm'
dist['gamma'] = 'log_norm'
dist['delta'] = 'log_norm'
dist['iota'] = 'log_norm'
dist['kappa'] = 'log_norm'
dist['lam'] = 'log_norm'
```

```
dist['nu_M'] = 'normal'
dist['nu_B'] = 'normal'
dist['nu_C'] = 'normal'
dist['nu_R'] = 'normal'
dist['xi'] = 'log_norm'
dist['pi'] = 'log_norm'
dist['rho'] = 'logit_norm'
dist['upsilon1'] = 'normal'
dist['ups_star2'] = 'normal'
dist['ups_star3'] = 'normal'
dist['phi'] = 'log_norm' if opt['B'] == 'pow' else 'logit_norm'
dist['psi'] = 'log_norm'
dist['omega'] = 'logit_norm'
xset = {}  # Parameter settings (x0, xmin, xmax)
xset['alpha'] = {'pool':([.1], [0], [1]), 'rand':([-1], [-10], [10])}
xset['beta'] = {'pool':([.1], [0], [1]), 'rand':([-1], [-10], [10])}
xset['gamma'] = {'pool':([1.], [0], [10]), 'rand':([0], [-10], [5])}
xset['delta'] = {'pool':([1.], [1e-3], [10]), 'rand':([0], [-10], [5])}
xset['iota'] = {'pool':([1.], [0], [10]), 'rand':([0], [-10], [5])}
xset['kappa'] = {'pool':([1.], [0], [10]), 'rand':([0], [-10], [5])}
xset['lam'] = {'pool':([1.], [0], [10]), 'rand':([0], [-10], [np.log(10)])}
xset['nu_M'] = {'pool':([0.], [-10], [10]), 'rand':([0], [-10], [10])}
xset['nu_B'] = {'pool':([0.], [-10], [10]), 'rand':([0], [-10], [10])}
xset['nu_C'] = {'pool':([0.], [-10], [10]), 'rand':([0], [-10], [10])}
xset['nu_R'] = {'pool':([0.], [-10], [10]), 'rand':([0], [-10], [10])}
xset['xi'] = {'pool':([1.], [0], [10]), 'rand':([0], [-10], [5])}
xset['pi'] = {'pool':([1.], [1e-3], [10]), 'rand':([0], [-10], [5])}
xset['rho'] = {'pool':([.5], [0], [1]), 'rand':([0], [-10], [10])}
xset['upsilon1'] = {'pool':([0.], [-10], [10]), 'rand':([0], [-10], [10])}
xset['ups_star2'] = {'pool':([0.], [-10], [10]), 'rand':([0], [-10], [10])}
xset['ups_star3'] = {'pool':([0.], [-10], [10]), 'rand':([0], [-10], [10])}
pow_set = {'pool':([1.], [0], [10]), 'rand':([0], [-10], [np.log(10)])}
exp_set = {'pool':([.9], [0], [1]), 'rand':([2], [-10], [10])}
xset['phi'] = pow_set if opt['B'] == 'pow' else exp_set
xset['psi'] = {'pool':([1.], [0], [10]), 'rand':([0], [-10], [5])}
xset['omega'] = {'pool':([.5], [0], [1]), 'rand':([0], [-10], [10])}
```

```
np.random.seed(seed)
ex0 = (None,)
ex02 = (None, slice(None), None)
ex1 = (slice(None), None)
ex2 = (slice(None), slice(None), None)
ex23 = (slice(None), slice(None), None, None)
ex3 = (slice(None), slice(None), slice(None), None)
dtype = torch.cuda.FloatTensor if cuda == True else torch.FloatTensor
torch.set_default_tensor_type(dtype)
opt['R_a'] = opt['v']
match = int(opt['P'] != 'none')
belief = int(opt['R_c'] == 'belief')
model = my.Model(opt)
data = []
for t in range(len(treat)):
    N, S, T, A, K = (num_sub[t], num_sim[t],
                     num_per[t], num_act[t], num_rule[t])
    Q = 2 * A
    X = 1 + match*A**2
    M = A ** (1-belief)
    C = M * A * X
    I = min(K, 2)
    df = pd.read_csv(dat_file[t])
    np_dat = df.to_numpy()
    raw_dat = torch.from_numpy(np_dat).type(dtype)
    df = pd.read_csv(mat_file[t])
    np_mat = np.array(df.to_numpy()[1:,2:], dtype=float).reshape(2, A, A)
    matrix = torch.from_numpy(np_mat).type(dtype)
    null = torch.full((N,), float(A ** 2))
    map_a = my.npeat(torch.arange(float(A)), X).repeat(M)
    map_x = torch.arange(float(X)).repeat(M * A)
    map_q1 = map_x // A
    map_q2 = map_x % A + A*map_x.eq(A ** 2).type(dtype)
    hypo = torch.empty(N, I * C, T)
    hist = torch.empty(N, I * C, T)
```

```
act = torch.empty(N, I * C, T)
cue = torch.empty(N, I * C, Q, T)
sim = torch.empty(N, I * C, Q, T)
mat = torch.empty(N, I * A, A)
for i in range(I):
    ixA = (slice(None), slice(i * A, (i+1) * A))
    ixC = (slice(None), slice(i * C, (i+1) * C))
    own_act = raw_dat[:,8+i].reshape(N, T)
    opp_act = raw_dat[:,8+1-i].reshape(N, T)
    player = (raw_dat[:,6] if i == 0 else 1 - raw_dat[:,6]).reshape(N, T)
    context = torch.cat((null[ex1], A*own_act[:,:-1] + opp_act[:,:-1]), 1)
    own_cue = context // A
    opp_cue = context % A + A*context.eq(A ** 2).type(dtype)
    history = (1-belief)*A*X*own_act + X*opp_act + match*context
    act[ixC] = opp_act[ex1].eq(map_a[ex02]).type(dtype)
    play = player[:,0][ex1].eq(torch.arange(2.)[ex0]).type(dtype)
    con = context[ex1].eq(map_x[ex02]).type(dtype)
    hypo[ixC] = act[ixC] * con if match == 1 else act[ixC]
    hist[ixC] = history[ex1].eq(torch.arange(float(C))[ex02]).type(dtype)
    cue1 = own_cue[ex1].eq(torch.arange(float(A))[ex02]).type(dtype)
    cue2 = opp_cue[ex1].eq(torch.arange(float(A))[ex02]).type(dtype)
    cue[ixC] = torch.cat((cue1, cue2), 1)[ex1].expand(-1, C, -1, -1)
    sim1 = -cue1[ex1] * own_cue[ex1].ne(map_q1[ex02]).type(dtype)[ex2]
    sim2 = -cue2[ex1] * opp_cue[ex1].ne(map_q2[ex02]).type(dtype)[ex2]
    sim[ixC] = torch.cat((sim1, sim2), 2)
    mat[ixA] = (play[ex23] * matrix[ex0]).sum(1)
play = play if I == 1 else 1 - play
own_act, opp_act = (own_act, opp_act) if I == 1 else (opp_act, own_act)
obs = own_act[ex1].eq(torch.arange(float(A))[ex02]).type(dtype)
out = opp_act[ex1].eq(torch.arange(float(A))[ex02]).type(dtype)
pay = (out[ex1] * mat[:,:A][ex3]).sum(2)
str_key = ('treat',)
int_key = ('N', 'S', 'T', 'K', 'A', 'Q', 'X', 'M','C', 'I')
ten_key = ('obs', 'hypo', 'hist',
           'act', 'cue', 'sim', 'mat', 'pay', 'play')
str_val = (treat[t],)
```

```
    int_val = (N, S, T, K, A, Q, X, M, C, I)
    ten_val = (obs, hypo, hist, act, cue, sim, mat, pay, play)
    dat = dict(zip(str_key+int_key+ten_key, str_val+int_val+ten_val))
    dat['hypo'] = torch.cat((torch.zeros(hypo.size()[:-1] + (1,)), hypo), -1)
    dat['hypo'][:,X-1::X, 0] = 1
    dat['hist'] = torch.cat((torch.zeros(hist.size()[:-1] + (1,)), hist), -1)
    dat['hist'][:,X-1::X, 0] = 1
    dat['act'] = torch.cat((torch.zeros(act.size()[:-1] + (1,)), act), -1)
    dat['cue'] = torch.cat((torch.zeros(cue.size()[:-1] + (1,)),cue), -1)
    dat['sim'] = torch.cat((torch.zeros(sim.size()[:-1] + (1,)), sim), -1)
    dat['mat'] = my.npeat(mat, (S,) + (1,) * (mat.dim()-1))
    dat['play'] = my.npeat(play, (S,) + (1,) * (play.dim()-1))
    data.append(dat)
R = 0
x0 = []
xmin = []
xmax = []
for (key, value) in est.items():
    if value == None:
        est[key] = xset[key]['pool'][0]
    elif type(value) == str:
        R += int(value == 'rand')
        x0 += xset[key][value][0]
        xmin += xset[key][value][1]
        xmax += xset[key][value][2]
fact = special.factorial(R-1, True)
x0 += R*[1] + int(R > 1)*fact*[0]
xmin += R*[1e-3] + int(R > 1)*fact*[-10]
xmax += R*[10] + int(R > 1)*fact*[10]
var = slice(-R - fact, -fact)
cov = slice(-fact, None)
bnds = [(xmin[x], xmax[x]) for x in range(len(x0))]
cons = [{'type': 'ineq', 'fun': my.con, 'args': (var, cov)}] if R > 1 else ()
dic = {'normal': my.Utility.lin,
       'log_norm': torch.exp, 'logit_norm': my.Probability.smax}
trans = {key: dic[value] for (key, value) in dist.items()}
```

```
stepsize = step * np.subtract(xmax, xmin)
take_step = my.Step(xmin, xmax, est, trans,
                    model, data, scale, niter, stepsize, cons)
arg = {'args':(cons, est, trans, model, data, scale), 'method': 'SLSQP',
       'jac': True, 'bounds': bnds, 'constraints': cons, 'tol': tol}
start = time.time()
res = optimize.basinhopping(my.fun, x0, niter=niter, T=temp, stepsize=stepsize,
                            take_step=take_step, minimizer_kwargs=arg)
end = time.time()
sec = end - start
print()
print('time: %dh:%dm:%ds'%(sec // 60**2 % 60, sec // 60 % 60, sec % 60))
```

B.3.2 mylib.py

```
import numpy as np
import torch
import copy
from scipy import linalg
from torch.distributions.multivariate_normal import MultivariateNormal

def npeat(x, rep):
    dim = int(np.argmax(rep))
    size1 = x.size()[:dim] + (int(np.max(rep)),) + x.size()[dim:]
    size2 = tuple(np.multiply(rep, x.size()))
    return x.repeat(rep).reshape(size1).transpose(dim, dim + 1).reshape(size2)

def con(x, var, cov):
    size = len(x[var])
    di = np.diag_indices(size)
    tril = np.tril_indices(size, -1)
    triu = np.triu_indices(size, 1)
    sigma = np.empty((size, size))
    sigma[di] = x[var]
    sigma[tril] = x[cov]
    sigma[triu] = sigma.T[triu]
    return np.real(linalg.eigvals(sigma))

def log_like(theta, model, data):
    N, S, T = (data['N'], data['S'], data['T'])
    p_a = torch.ones(N * S).type(torch.cuda.FloatTensor)
    model.start(theta, data)
    for t in range(1, T + 1):
        model.update(data, t)
        p_a = p_a * (npeat(data['obs'][:,:,t-1], (S, 1)) * model.p_a).sum(1)
    ll = p_a.reshape(N, S).mean(1).log().sum()
    return ll


def fun(x, cons, est, trans, model, data, scale):
    for con in cons:
        valid = np.all(con['fun'](x, *con['args']) > 0)
        if valid == False: break
    if valid == False:
        ll = float('nan')
        grad = np.empty(len(x))
        grad[:] = np.nan
        return (-ll, -grad)
    else:
        torch.manual_seed(7)
        ex0 = (None,)
        ex1 = (slice(None), None)
        x = torch.tensor(x.astype(np.float32), requires_grad=True)
        ll = 0
        for d in range(len(data)):
            N, S, K = (data[d]['N'], data[d]['S'], data[d]['K'])
            p = 0
            mu = torch.tensor([])
            re = []
            theta = {}
            for (key, val) in est.items():
                if type(val) != str:
```

```
        theta[key] = torch.tensor(val)
    elif val == 'pool':
        theta[key] = x[p][ex0]
        p += 1
    elif val == 'rand':
        mu = torch.cat((mu, x[p][ex0]))
        p += 1
        re.append(key)
R = len(re)
di = tuple([torch.tensor(ind, dtype=torch.long)
            for ind in np.diag_indices(R)])
tril = tuple([torch.tensor(ind, dtype=torch.long)
              for ind in np.tril_indices(R, -1)])
triu = tuple([torch.tensor(ind, dtype=torch.long)
              for ind in np.triu_indices(R, 1)])
sigma = torch.empty(R, R)
sigma[di] = x[p:p+R]
sigma[tril] = x[p+R:]
sigma[triu] = sigma.t()[triu]
mu = mu.type(torch.cuda.FloatTensor)
sigma = sigma.type(torch.cuda.FloatTensor)
try:
    sample = MultivariateNormal(mu, sigma).rsample((N * S,))
except (RuntimeError, ValueError):
    ll = float('nan')
    grad = np.empty(len(x))
    grad[:] = np.nan
    return (-ll, -grad)
theta.update({re[r]: trans[re[r]](sample[:,r]) for r in range(R)})
theta['upsilon'] = torch.zeros(N * S)[ex1]
if K > 1:
    theta['upsilon'] = torch.cat((theta['upsilon'],
                                  theta['ups_star2'][ex1]), 1)
if K > 2:
    theta['upsilon'] = torch.cat((theta['upsilon'],
                                  theta['ups_star3'][ex1]), 1)
            theta['upsilon'] = (theta['upsilon']
                                /theta['kappa'] + theta['upsilon1'])
            if data[d]['treat'] == 'pvd': theta['omega'] = torch.zeros(1)
            ll = ll + log_like(theta, model, data[d])
        ll = ll * scale
        ll.backward()
        grad = np.array(x.grad.cpu())
        return (-ll.item(), -grad)


class Step(object):
    def __init__(self, xmin, xmax, est, trans,
                 model, data, scale, niter, stepsize=0.5, cons=()):
        self.xmin = xmin
        self.xmax = xmax
        self.stepsize = stepsize
        self.cons = cons
        self.est = est
        self.trans = trans
        self.model = model
        self.data = data
        self.scale = scale
        self.niter = niter
        self.iter = 0

    def __call__(self, x):
        valid = False
        while valid == False:
            eps = np.random.uniform(-self.stepsize, self.stepsize)
            xnew =  np.clip(x + eps, self.xmin, self.xmax)
            ll, grad = fun(xnew, self.cons, self.est,
                           self.trans, self.model, self.data, self.scale)
            valid_ll = not np.isnan(ll) and ll != -np.inf and ll != np.inf
            valid_grad = (not np.isnan(grad).any() and np.not_equal(grad,
                          -np.inf).any() and np.not_equal(grad, np.inf).any())
            valid = valid_ll and valid_grad
        self.iter += 1
        print("********* Iteration %d *********"%self.iter)
        if self.iter == self.niter:
            np.save('new_x0.npy', x)
            np.save('new_step.npy', self.stepsize)
        return xnew


class Utility:

    def lin(c):
        return c

    class Learning:

        def none(U, *args):
            return U

        def diff(U, R, I, alpha):
            return U + I*alpha*(R-U)


class Probability:

    def rand(x, dim=None):
        size = x.size()
        return torch.full(size, 1 / size[dim])

    def hmax(x, dim=None):
        intermed = x.eq(x.max(dim, True)[0]).type(x.type())
        return intermed/intermed.sum(dim, keepdim=True)

    def smax(x, alpha = 0.0, lam=1.0, dim=None):
        return torch.nn.functional.softmax(alpha+lam*x, dim)

    class Weighting:
        def lin(p):
            return p

        def linlog(p, gamma, delta):
            pstar = delta * p**gamma
            return pstar / (pstar+(1-p)**gamma)

        def ratio(p, gamma, delta, dim=None):
            pstar = p ** gamma
            return delta * pstar / (pstar.sum(dim, True)-pstar+delta*pstar)


class Activation:

    class Base:

        def none(B, R, eps=1e-10, **kwargs):
            return (R + B.exp() + eps).log()

        def exp(B, R, gamma, eps=1e-10, **kwargs):
            return (R + gamma*B.exp() + eps).log()

        def pow(t, R_j, d, eps=1e-10, dim=None, **kwargs):
            return ((R_j*t**-d).sum(dim) + eps).log()

        class Learning:

            def none(**kwargs):
                return

            def belief(attn, hypo, **kwargs):
                return attn * hypo

            def reinf(attn, hist, **kwargs):
                return attn * hist
            def attrac(attn, hypo, hist, delta):
                return attn * (delta*hypo+(1-delta)*hist)

    class Associative:

        def none(S_j, *args):
            return S_j

        def spread(S_j, W, dim=None):
            return(W * S_j).sum(dim)

        class Learning:

            def none(S_j, *args):
                return S_j

            def diff(S_j, S, I_i, I_j, S_max, alpha):
                return S_j + I_j*alpha*(I_i*S_max-S)

    class Matching:

        def none(*args):
            return 0

        def part(Sim, MP, dim=None):
            return (MP * Sim).sum(dim)


class Model:

    def __init__(self, opt):
        fun = {'p_k': {'rand': Probability.rand,
                       'hmax': Probability.hmax,
                       'smax': Probability.smax},
               'p_ak': {'rand': Probability.rand,
                        'hmax': Probability.hmax,
                        'smax': Probability.smax},
               'U' : {'none': Utility.Learning.none,
                      'diff': Utility.Learning.diff},
               'v': {'lin': Utility.lin},
               'w': {'lin': Probability.Weighting.lin,
                     'linlog': Probability.Weighting.linlog,
                     'ratio': Probability.Weighting.ratio},
               'b_1': {'rand': Probability.rand,
                       'hmax': Probability.hmax,
                       'smax': Probability.smax},
               'b_k': {'rand': Probability.rand,
                       'hmax': Probability.hmax,
                       'smax': Probability.smax},
               'B': {'none': Activation.Base.none,
                     'exp': Activation.Base.exp,
                     'pow': Activation.Base.pow},
               'S': {'none': Activation.Associative.none,
                     'spread': Activation.Associative.spread},
               'P': {'none': Activation.Matching.none,
                     'part': Activation.Matching.part},
               'R_a': {'lin': Utility.lin},
               'R_c': {'none': Activation.Base.Learning.none,
                       'belief': Activation.Base.Learning.belief,
                       'reinf': Activation.Base.Learning.reinf,
                       'attrac': Activation.Base.Learning.attrac},
               'S_q': {'none': Activation.Associative.Learning.none,
                       'diff': Activation.Associative.Learning.diff}}
        self.fun = {key: fun[key][val] for (key, val) in opt.items()}
        self.opt = opt

    def start(self, theta, dat):
        ex0 = (None,)
        ex1 = (slice(None), None)
        ex12 = (slice(None), None, None)
        ex2 = (slice(None),slice(None), None)
```

```
omega = theta['omega'][ex1].expand(-1, dat['mat'].size()[-1])
W = torch.cat((omega, 1 - omega), 1)
xi = W * theta['xi'][ex1]
beta = W * theta['beta'][ex1]
if len(theta['nu_M']) == 1:
    nu1 = torch.cat((torch.tensor([0.]), theta['nu_M'], theta['nu_B']))
    nu2 = torch.cat((torch.tensor([0.]), theta['nu_C'], theta['nu_R']))
    nu = (dat['play'][ex1]
            * torch.cat((nu1[ex1], nu2[ex1]), 1)[ex0]).sum(2)
else:
    nu1 = torch.cat((torch.zeros(theta['nu_M'].shape[0])[ex1],
                                 theta['nu_M'][ex1],
                                 theta['nu_B'][ex1]), 1)
    nu2 = torch.cat((torch.zeros(theta['nu_C'].shape[0])[ex1],
                                 theta['nu_C'][ex1],
                                 theta['nu_R'][ex1]), 1)
    nu = (dat['play'][ex1]
            * torch.cat((nu1[ex2], nu2[ex2]), 2)).sum(2)
arg = {'p_k': {'rand': (1,),
               'hmax': (1,),
               'smax': (0., theta['kappa'][ex1], 1)},
       'p_ak': {'rand': (1,),
                'hmax': (1,),
                'smax': (nu[ex2], theta['lam'][ex12], 1)},
       'U': {'none': (),
             'diff': (theta['alpha'][ex1],)},
       'v': {'lin': (dat['mat'],)},
       'w': {'lin': (),
             'linlog': (theta['gamma'][ex12], theta['delta'][ex12]),
             'ratio': (theta['gamma'][ex12], theta['delta'][ex12], 2)},
       'b_1': {'rand': (2,),
              'hmax': (2,),
              'smax': (0., 1, 2)},
       'b_k': {'rand': (2,),
              'hmax': (2,),
              'smax': (0., theta['iota'][ex12], 2)},
            'B': {'cons': {},
                  'exp': {'gamma': theta['phi'][ex1]},
                  'pow': {'d': theta['phi'][ex12], 'dim': 2}},
            'S': {'none': (),
                  'spread': (2,)},
            'P': {'none': (),
                  'part': (xi[ex1], 2)},
            'R_a': {'lin': ()},
            'R_c': {'none': {},
                    'belief': {},
                    'reinf': {},
                    'attrac': {'delta': theta['rho'][ex1]}},
            'S_q': {'none': (),
                    'diff': (theta['psi'][ex12], beta[ex1])}}
        self.arg = {key: arg[key][val] for (key, val) in self.opt.items()}
        self.theta = theta
        self.p_k = torch.zeros((1, 1))
        self.p_ak = torch.zeros(1)
        self.p_a = torch.zeros((1, 1))
        self.U = theta['upsilon']
        self.v = self.fun['v'](*self.arg['v'])
        self.B = torch.zeros(1).log()
        self.S = torch.zeros(1)[ex1]
        self.R_ct = torch.zeros(0)
        self.S_q = torch.zeros(1)[ex1]

    def update(self, dat, t):
        old = (Ellipsis, t - 1)
        new = (Ellipsis, t)
        ex1 = (slice(None), None)
        ex2 = (slice(None), slice(None), None)
        N, S, A, X, M, C, K, I = (dat['N'], dat['S'], dat['A'], dat['X'],
                                  dat['M'], dat['C'], dat['K'], dat['I'])
        hypo, hist, act, cue, sim, pay, obs = (dat['hypo'], dat['hist'],
                                               dat['act'], dat['cue'],
                                               dat['sim'], dat['pay'],
                                        dat['obs'])
hypo_old = npeat(hypo[old], (S,)+(1,)*(hypo[old].dim()-1))
hist_old = npeat(hist[old], (S,)+(1,)*(hist[old].dim()-1))
act_old = npeat(act[old], (S,)+(1,)*(act[old].dim()-1))
cue_old = npeat(cue[old], (S,)+(1,)*(cue[old].dim()-1))
cue_new = npeat(cue[new], (S,)+(1,)*(cue[new].dim()-1))
sim_new = npeat(sim[new], (S,)+(1,)*(sim[new].dim()-1))
pay_old = npeat(pay[...,t-2], (S,)+(1,)*(pay[...,t-2].dim()-1))
obs_old = npeat(obs[...,t-2], (S,)+(1,)*(obs[...,t-2].dim()-1))
fun, arg = (self.fun, self.arg)
R_a = fun['R_a'](pay_old, *arg['R_a'])
R = (R_a*obs_old).sum(1)
numerator = self.p_k * (self.p_ak*obs_old[ex2]).sum(1) + (1e-10)/K
denominator = (self.p_a * obs_old).sum(1, keepdim=True) + 1e-10
p_ka = numerator / denominator
self.U = fun['U'](self.U, R[ex1],  p_ka if t > 1 else 0, *arg['U'])
attn = 1 if t > 1 else self.theta['pi'][ex1]
R_c = fun['R_c'](attn, hypo=hypo_old, hist=hist_old, **arg['R_c'])
self.R_ct = torch.cat((self.R_ct, R_c[ex2]), 2)
time = torch.arange(float(t), 0, -1)
self.S_q = fun['S_q'](self.S_q, self.S[ex2],
                      act_old[ex2], cue_old, *arg['S_q'])
self.B = fun['B'](B=self.B, t=time, R=R_c, R_j=self.R_ct, **arg['B'])
self.S = fun['S'](self.S_q, cue_new, *arg['S'])
P = fun['P'](sim_new, *arg['P'])
A_c = (self.B + self.S + P).reshape(N * S, I * M, C // M)
b_1 = fun['b_1'](A_c, *arg['b_1']).reshape(N * S, I * M, A, X).sum(3)
w = npeat(fun['w'](b_1, *arg['w']), (1, A // M, 1))
V_k = (w * self.v).sum(2)[ex2]
for k in range(1, K):
    own = (slice(None), slice(0, A), k - 1)
    opp = (slice(None), slice(A, 2 * A), k - 1)
    V = torch.cat((V_k[opp], V_k[own]), 1).reshape(N * S, 2, A)
    b_k = npeat(fun['b_k'](V, *arg['b_k']), (1, A, 1))
    V_k = torch.cat((V_k, (b_k * self.v).sum(2)[ex2]), 2)
self.p_k = fun['p_k'](self.U, *arg['p_k'])
```

```
        self.p_ak = fun['p_ak'](V_k[:,:A], *arg['p_ak'])
        self.p_a = (self.p_k[ex1] * self.p_ak).sum(2)
```

B.4 Analysis of Rutström and Wilcox (2009)

The experiment conducted in Rutström and Wilcox (2009) consisted of three between-subject treatments: the "no beliefs" (NB) treatment (80 subjects), the "scoring rule" (SR) treatment (92 subjects), and the "expected choice" (EC) treatment (92 subjects). The subjects in all three treatments were matched against a fixed human opponent for 36 periods in the mixed strategy game shown in Figure B.4. In the NB treatment, subjects played the game normally with no belief elicitation. In the SR treatment, a quadratic scoring rule was used to incentivize belief elicitation. Finally, in the EC treatment, subjects were asked to guess their opponent's next action without any reward for accuracy.

The most notable feature of the game in Figure B.4 is the large payoff that the row player receives when actions T and L are chosen. While this sort of sharp payoff separation could help identify level-k rule learning and incentivize the use of higher level rules, analyzing this game comes with several challenges. First, there is the problem with the game's dimensionality, as it is only a 2×2 game. In an action space of this size, there will always be overlap in the actions prescribed by the first three level-k rules. Second, the row player's TL payoff is so large that risk aversion may be a relevant factor. In the analysis that follows, I continue to assume that subjects have linear utility, but that assumption may not be appropriate for this data. Finally, the empirical action choices of of row players in this game are somewhat difficult to explain. Although column players chose action L more than *six times* as often as they should in the mixed strategy Nash equilibrium, row players continued to play action B with a surprisingly large frequency. Consequently, the authors' belief learning model estimates the average subject to have an initial bias against playing action T that reduces its latent utility by more than \$2.00. The enormity of this bias suggests that there may be latent utility factors (e.g., risk aversion, joy of winning) that are important in this environment that are completely outside the scope of my model. With these caveats in mind, I will now proceed to summarizing the results of my analysis as they relate to the analysis in Chapter 2.

As in Chapter 2, I find that the initial rule utility means and variances are generally significant across all estimations, both jointly and at the individual parameter level. Again however, the initial rule utility covariances and all of the level-k parameters are jointly insignificant. The evidence on rule learning is a little more mixed however. In some treatments, I find the rule learning step-size parameter α to be individually significant, but in others it is not. In estimaion (8), which uses the data from all three treatments, α is insignificant. The estimates of α are also generally smaller than those of Chapter 2. Looking at the time series of simulated rule frequencies in Figure B.6, we see that rule learning is muted or non-existent in most estimations, similar to the way it is in my estimation of only the PvP data.[1] As in that estimation, we also see that the level-1 rule has the highest frequency, followed by the level-3 rule, followed by the level-2 rule which is used very infrequently. Given the similarities that these simulations share with the PvP estimation, I suspect that these estimations also struggle to clearly identify level-k reasoning. Unfortunately in this case, however, there is no PvD treatment to help identify the model's belief learning parameters.

The simulated action frequencies, conditional action frequencies, inert action frequencies, per-unit changes in inert action frequencies, and distributions of z-statistics all fit the data reasonably well as they do in Chapter 2. As in those simulations, the estimated level-k models do not provide a noticeably better fit to the data. In contrast to the Chapter 2 estimations, however, the simulated rule performances for row players largely depends on on whether the level-1 or level-k model is specified. In the former case, level-2 reasoning consistently performs the best against level-1 and level-2 players. Counter-intuitively, it also ties the level-1 rule and beats the level-3 rule in its performance against level-3 players. I believe this happens because the column players' action selection biases make them frequently choose action L, which keeps level-3 row players choosing action B, but I have yet to confirm this. For the estimated level-k models, row players' simulated rule earnings are largely the same

[1]↑Interestingly, there appears to be some level-1 rule learning in the SR treatment. This finding is consistent with Rutström and Wilcox's observation that SR subjects adhered more closely to the predictions of their belief learning model.

	L	R
T	19, 0	0, 1
B	0, 1	1, 0

Figure B.4. The repeated stage game used in Rutström and Wilcox (2009). Payoffs are listed in experimental currency units with a conversion rate of 1 experimental currency units = .2 U.S. Dollars.

Table B.4. Parameter estimates, estimations (1) - (4)

Parameter	Range	Null	(1)	(2)	(3)	(4)
β	[0, 1]	0	.496***	.202**	.486**	.361***
			(.124)	(.072)	(.214)	(.089)
γ	[0, 10]	1	.105***	.099***	.155***	.130***
			(.057)	(.041)	(.035)	(.022)
$\mu_{\ln(\lambda)}$	[-10, ln(10)]	-7	1.61***	1.57***	1.60***	1.59***
			(.140)	(.100)	(.085)	(.042)
$\mu_{\ln(\phi)}$	[-10, ln(10)]	-7	-2.33**	-2.70**	-3.22*	-2.56***
			(1.59)	(.985)	(1.15)	(.799)
ν(B)	[-10, 10]	0	10.0***	10.0***	10.0***	10.0***
			(.764)	(.528)	(.502)	(.235)
ν(R)	[-10, 10]	0	-.643***	-.782***	-.618***	-.682***
			(.151)	(.143)	(.127)	(.082)
$\sigma^2_{\ln(\lambda)}$	(0, 10]	0	.002	.012**	.004*	.005***
			(.004)	(.008)	(.004)	(.002)
$\sigma_{\ln(\lambda),\ln(\phi)}$	[-10, 10]	0	.027	.349**	.019	.034
			(.050)	(.140)	(.111)	(.079)
$\sigma^2_{\ln(\phi)}$	(0, 10]	0	3.15	10.0***	10.0***	6.24
			(1.92)	(2.67)	(3.31)	(2.73)
ψ	[0, 10]	0	.625	3.05*	.397	.738**
			(.704)	(1.80)	(.746)	(.414)
$\omega_i(a_i)$	[0, 1]	0	.396*	.194*	.694**	.402**
			(.198)	(.156)	(.212)	(.170)
Data			NB	SR	EC	All
Log likelihood			-1806.58	-1952.26	-2060.88	-5840.71

Bootstrap std. errors in parentheses. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

except for a dramatic increase in the profitability of level-1 reasoning. Level-1 reasoning becomes so profitable in fact that it surpasses level-2 reasoning as the most profitable rule. I believe this happens because the action selection bias on action B is much smaller in the level-k estimations, giving the level-1 player more freedom to best respond to the actual action history. Finally, for column players, regardless of whether the level-1 or level-k model is specified, the level-1 rule earns the most against level-1 and level-3 players. The level-3 rule performs better against level-2 players as we would expect because it best responds to that level of reasoning.

Table B.5. Parameter estimates, estimations (5) - (8)

Parameter	Range	Null	(5)	(6)	(7)	(8)
α	[0, 1]	0	.000	.003**	.038*	.005
			(.035)	(.001)	(.019)	(.002)
β	[0, 1]	0	.296*	.296*	.438**	.438*
			(.042)	(.105)	(.044)	(.025)
γ	[0, 10]	1	3.04	.350**	.289**	.231**
			(.121)	(.089)	(.030)	(.006)
κ	[0, 10]	0	.062*	2.90***	.024**	.470*
			(.030)	(.061)	(.006)	(.025)
$\mu_{\ln(\lambda)}$	[-10, ln(10)]	-7	.573**	1.62***	1.77**	1.83*
			(.227)	(.117)	(.041)	(.027)
$\mu_{\pi^*(2)}$	[-10, 10]	-7	-4.17*	-2.50**	-5.14**	-5.09*
			(.105)	(.289)	(.049)	(.010)
$\mu_{\pi^*(3)}$	[-10, 10]	-7	-1.13*	-.952**	-.471**	-.400*
			(.157)	(.309)	(.044)	(.017)
$\mu_{\ln(\phi)}$	[-10, ln(10)]	-7	-2.96*	-.342***	-1.30**	-.266***
			(.145)	(.313)	(.035)	(.032)
ν(B)	[-10, 10]	0	1.85	10.0***	3.91**	5.87**
			(.194)	(.067)	(.075)	(.009)
ν(R)	[-10, 10]	0	-1.23	-1.19*	-1.26*	-1.32
			(.104)	(.134)	(.038)	(.029)
π(1)	[-10, 10]	0	4.40**	-10.0***	3.94***	-8.20*
			(.009)	(.014)	(.006)	(.005)
$\sigma^2_{\ln(\lambda)}$	(0, 10]	0	5.25**	.194*	2.00**	1.10**
			(.079)	(.077)	(.040)	(.023)
$\sigma_{\ln(\lambda),\pi^*(2)}$	[-10, 10]	0	-3.33	.787	2.95**	.924
			(.057)	(.193)	(.026)	(.016)
$\sigma_{\ln(\lambda),\pi^*(3)}$	[-10, 10]	0	2.69	.561**	.371	.356
			(.094)	(.100)	.037	(.015)
$\sigma_{\ln(\lambda),\ln(\phi)}$	[-10, 10]	0	3.95	.116	.195*	-.089
			(.085)	(.132)	(.025)	(.015)
$\sigma^2_{\pi^*(2)}$	(0, 10]	0	7.73**	3.51**	8.34**	8.99**
			(.026)	(.385)	(.017)	(.003)
$\sigma_{\pi^*(2),\pi^*(3)}$	[-10, 10]	0	-.699	2.62**	2.45**	-.205
			(.063)	(.181)	(.040)	(.009)
$\sigma_{\pi^*(2),\ln(\phi)}$	[-10, 10]	0	-5.07	-.453	-2.37*	-2.55*
			(.044)	(.268)	(.024)	(.004)
$\sigma^2_{\pi^*(3)}$	(0, 10]	0	2.03*	2.00**	1.19**	.647 **
			(.116)	(.195)	(.035)	(.018)
$\sigma_{\pi^*(3),\ln(\phi)}$	[-10, 10]	0	2.10*	-.664*	-1.09*	-.691
			(.086)	(.187)	(.037)	(.018)
$\sigma^2_{\ln(\phi)}$	(0, 10]	0	9.97**	2.89**	4.70**	7.35***
			(.023)	(.271)	(.015)	(.006)
ψ	[0, 10]	0	7.95*	1.12	7.22**	8.64**
			(.106)	(.393)	(.008)	(.002)
$\omega_i(a_i)$	[0, 1]	0	.000	.270	.502**	.203*
			(.047)	(.181)	(.035)	(.020)
Data			NB	SR	EC	All
Log likelihood			-1746.98	-1876.31	-2002.46	-5643.01

Bootstrap std. errors in parentheses. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$

Table B.6. p-values for multiple hypothesis tests

Parameters	Description	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
α, κ, $\boldsymbol{\pi}(1)$	rule learning					.085	.011	.076	.075
$\mu_{\pi^*(2)}$, $\mu_{\pi^*(3)}$	initial rule utility means					.095	.041	.051	.063
$\sigma^2_{\pi^*(2)}$, $\sigma^2_{\pi^*(3)}$	initial rule utility variances					.076	.058	.037	.039
$\boldsymbol{\sigma}^*$	initial rule utility covariances					.277	.226	.204	.171
$\Uparrow$	all of the above					.248	.134	.180	.111
β, ψ, ω	pattern recognition	.054	.039	.057	.006	.108	.164	.047	.043

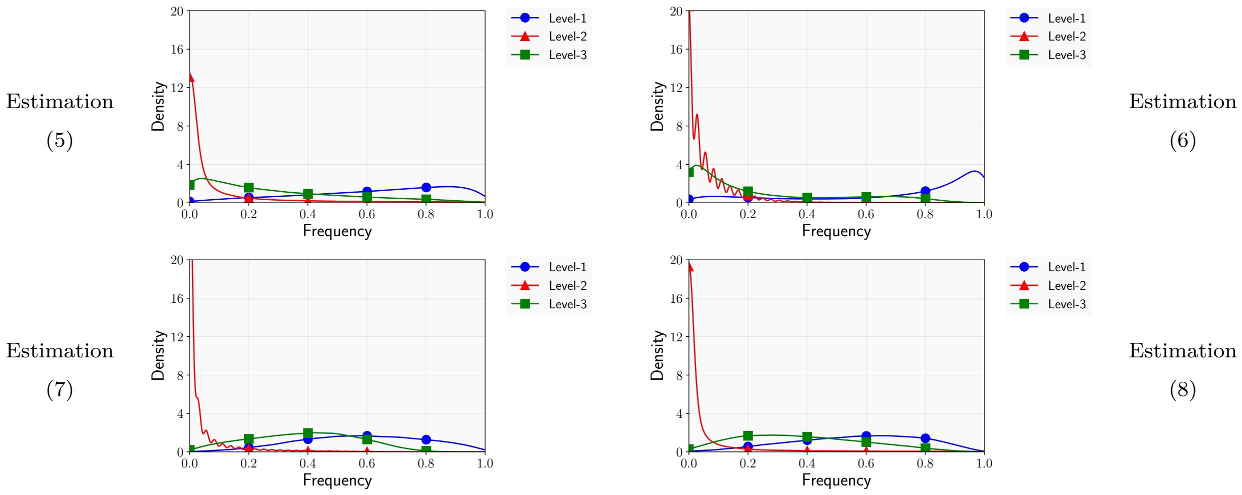


Figure B.5. Simulated distribution of rule frequencies, periods 1-36

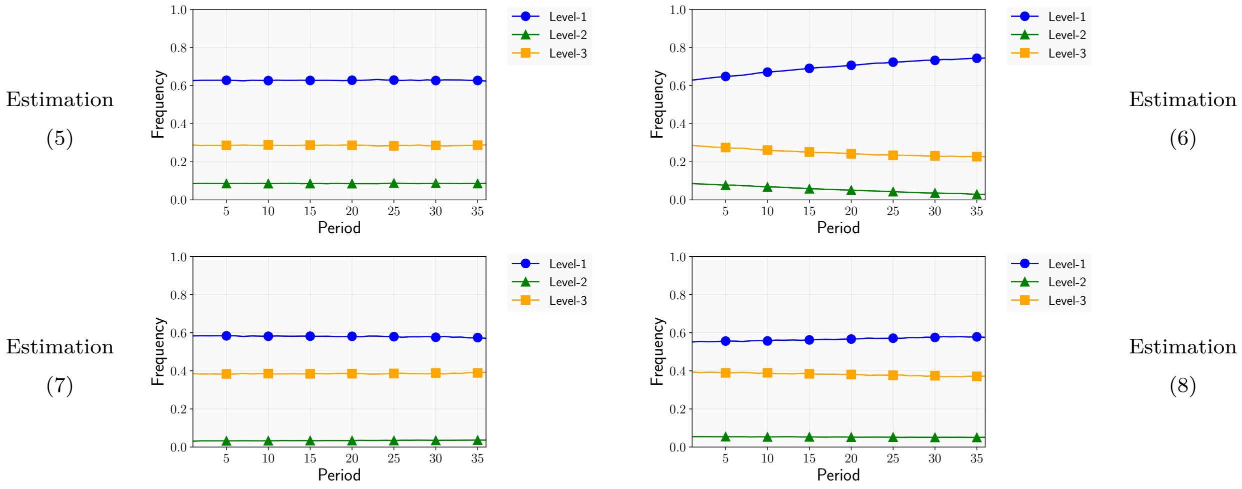


Figure B.6. Simulated rule frequencies, periods 1-36

Estimations (1) & (5) | Estimations (2) & (6)

Estimations (3) & (7) | Estimations (4) & (8)

Estimations (1) & (5) | Estimations (2) & (6)

Estimations (3) & (7) | Estimations (4) & (8)

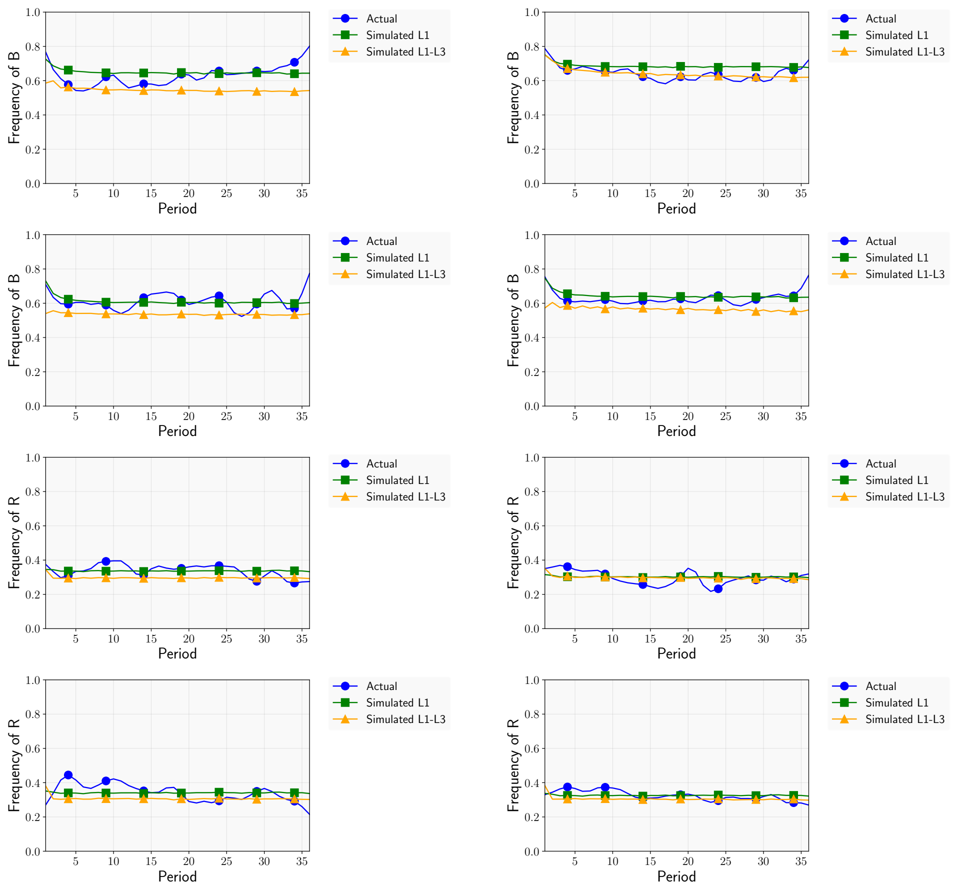
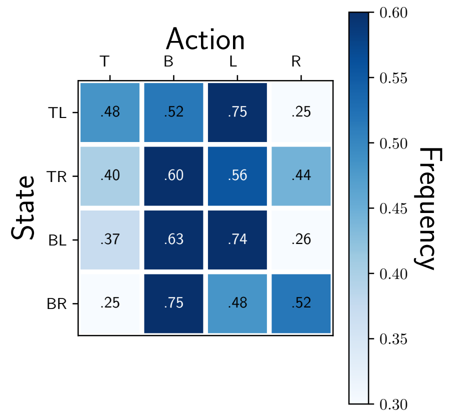
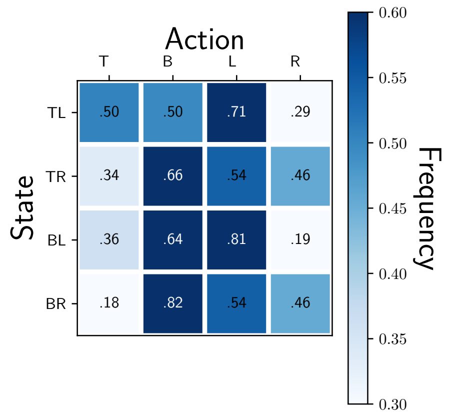


Figure B.7. Action frequencies, periods 1-36

Actual
NB



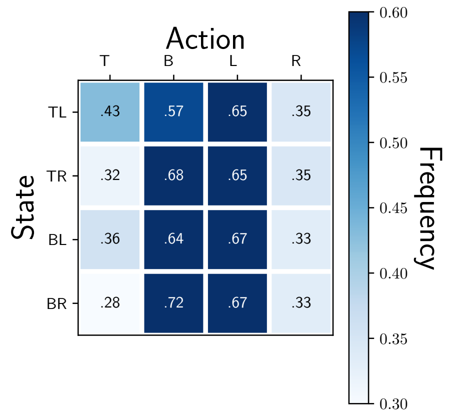


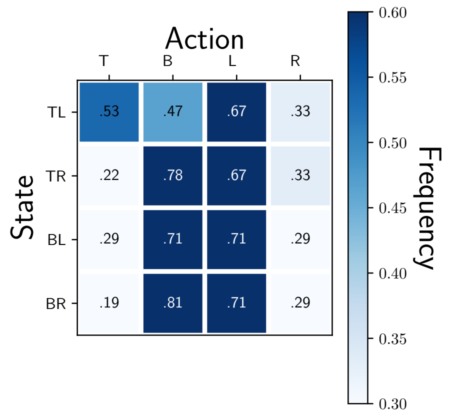
Actual
SR

Simulated
NB

Estimation

(1)





Simulated
SR

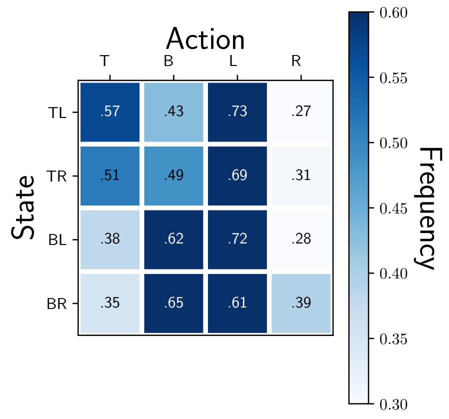
Estimation

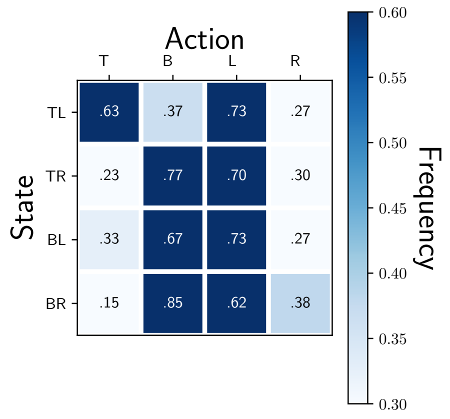
(2)

Simulated
NB

Estimation

(5)





Simulated
SR

Estimation

(6)

Figure B.8. NB and SR action frequencies by state, periods 2-36

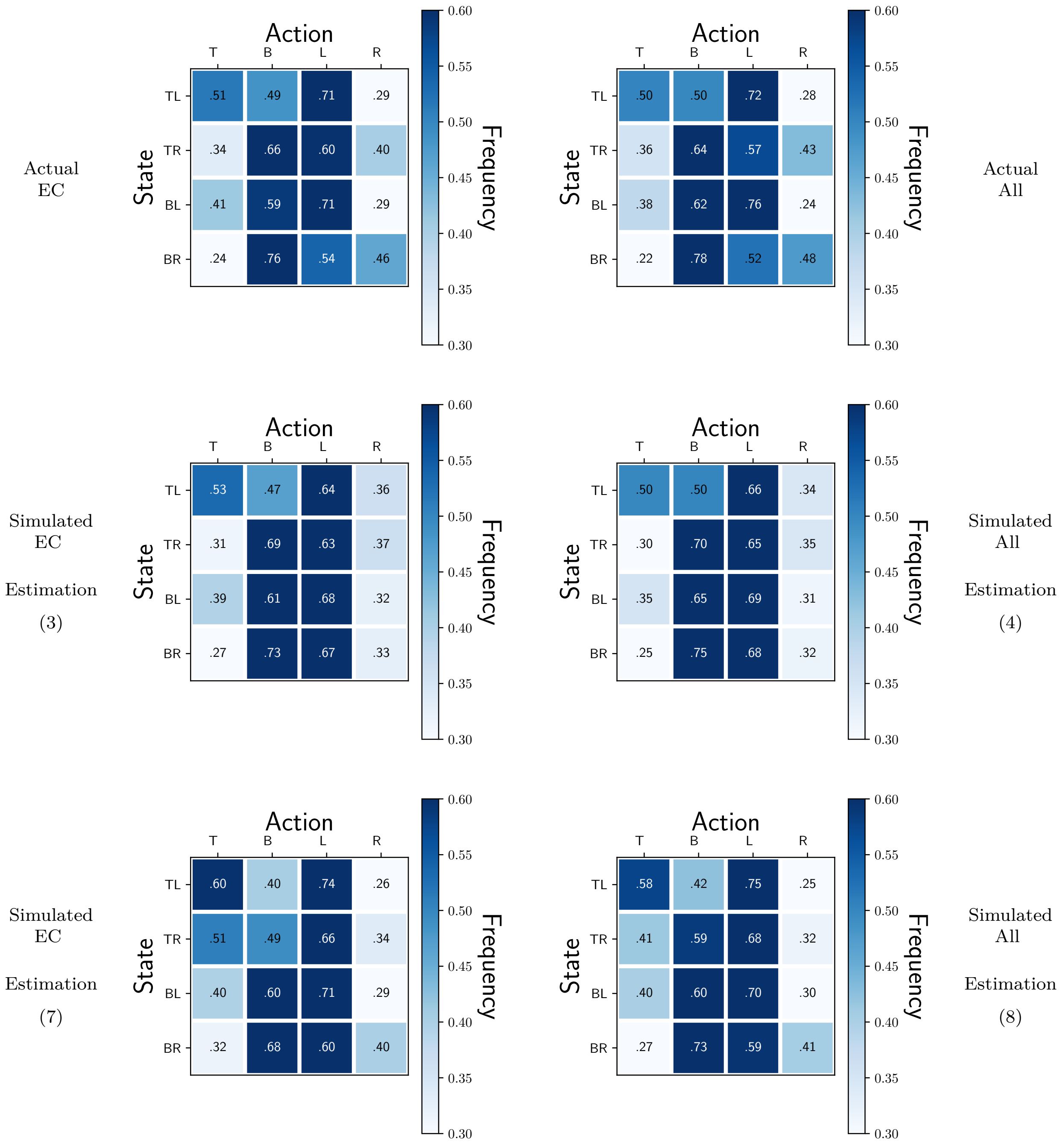


Figure B.9. EC and All action frequencies by state, periods 2-36

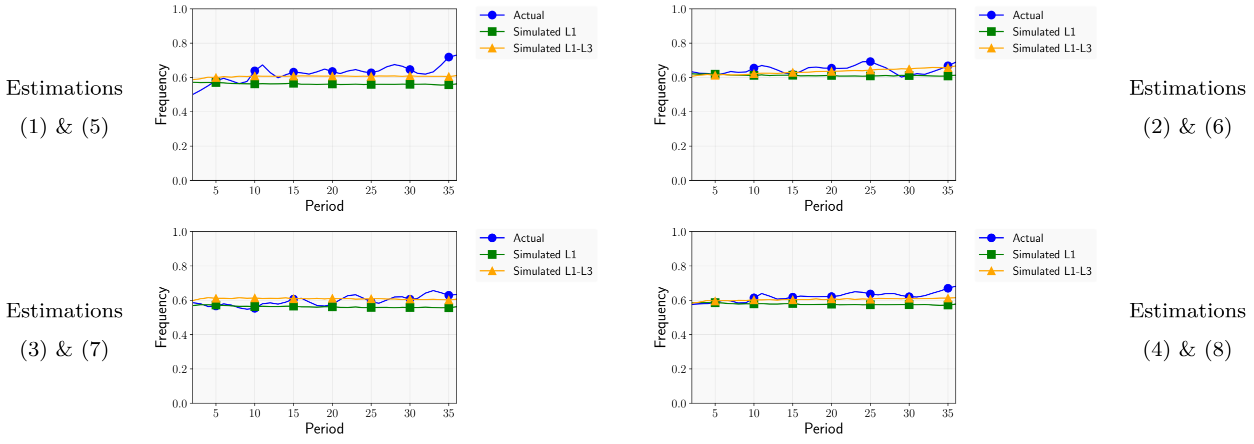


Figure B.10. Inert action frequencies, periods 2-36

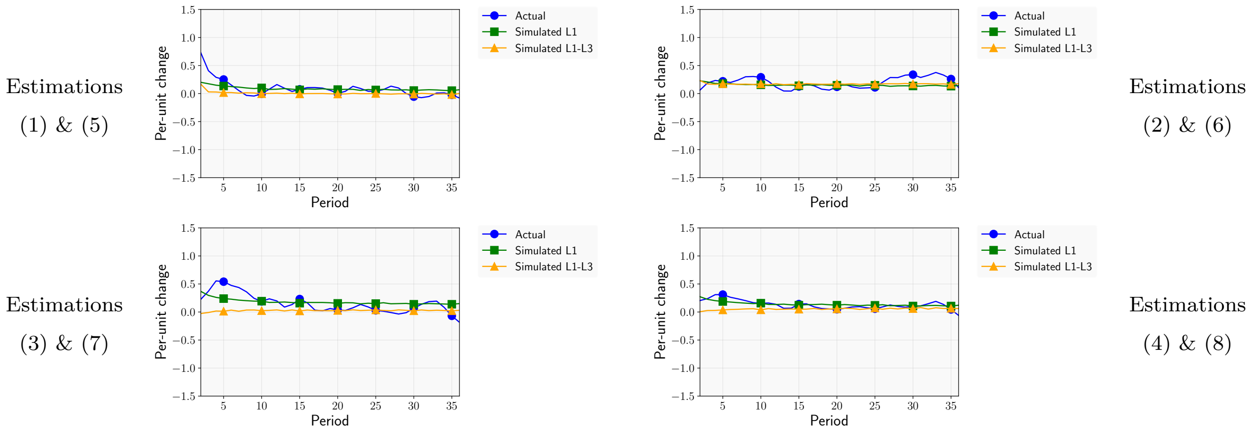


Figure B.11. Per-unit change in inert action frequencies following a win, periods 2-36

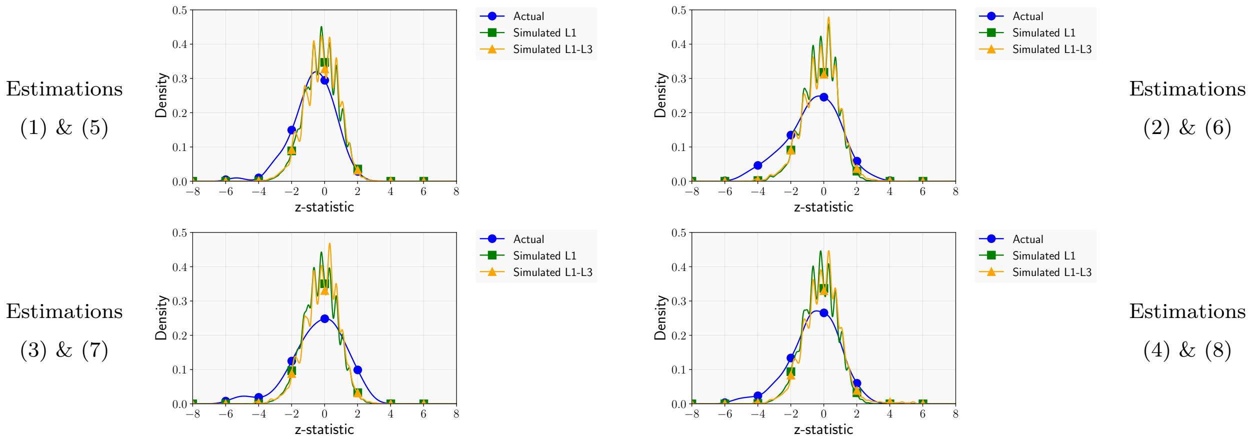


Figure B.12. Distribution of z-statistics, periods 1-36

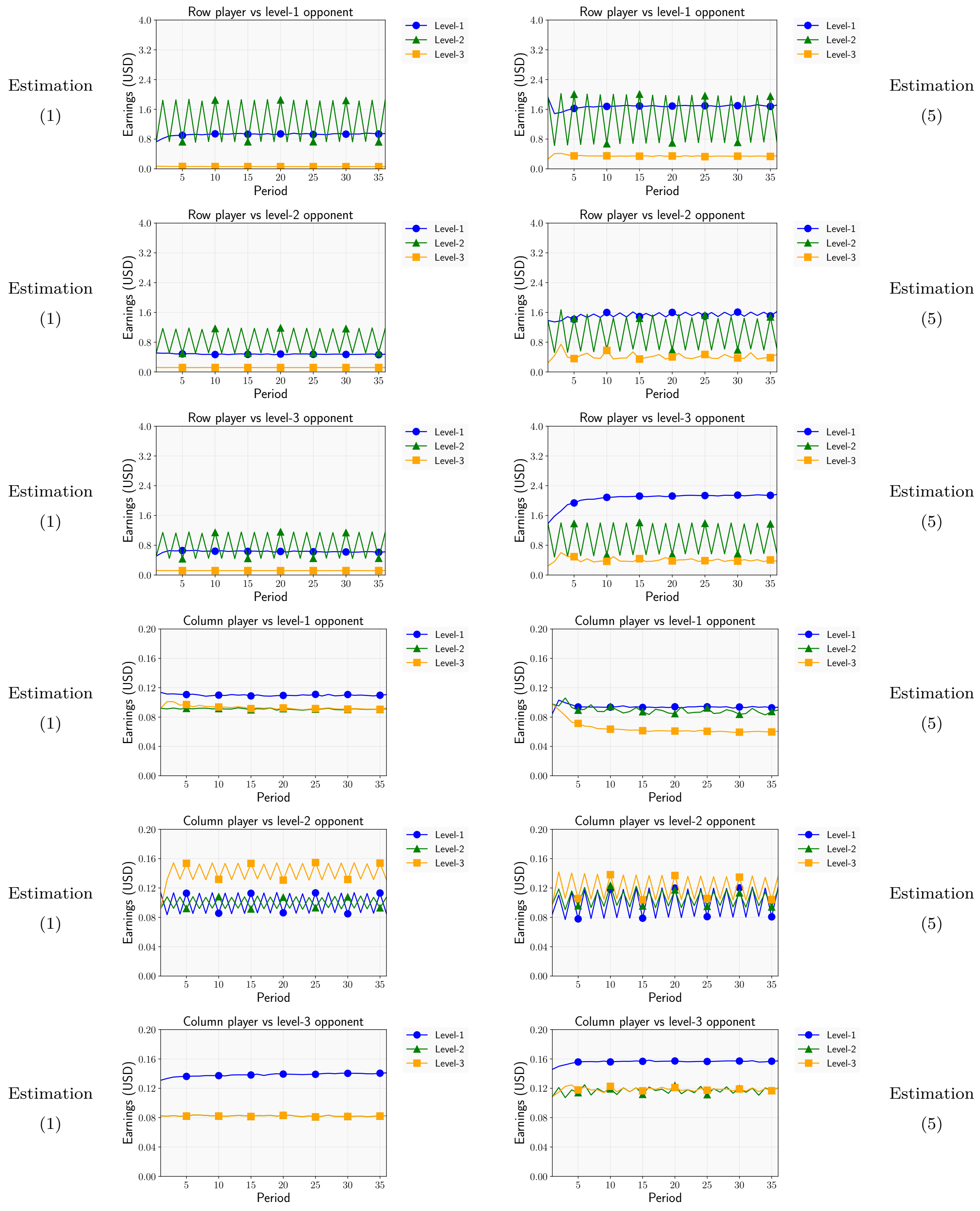


Figure B.13. Average earnings by rule, estimations (1) and (5)

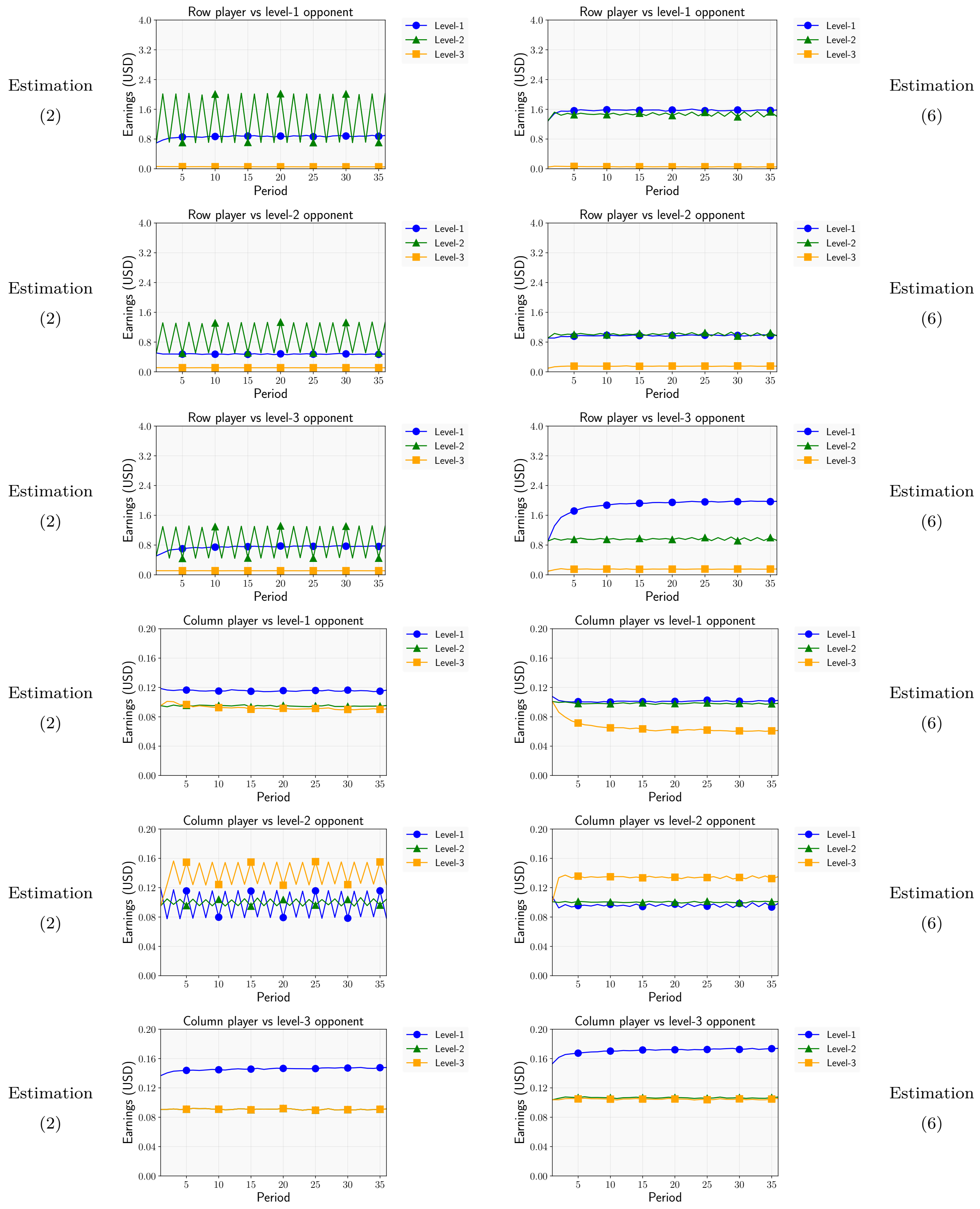


Figure B.14. Average earnings by rule, estimations (2) and (6)

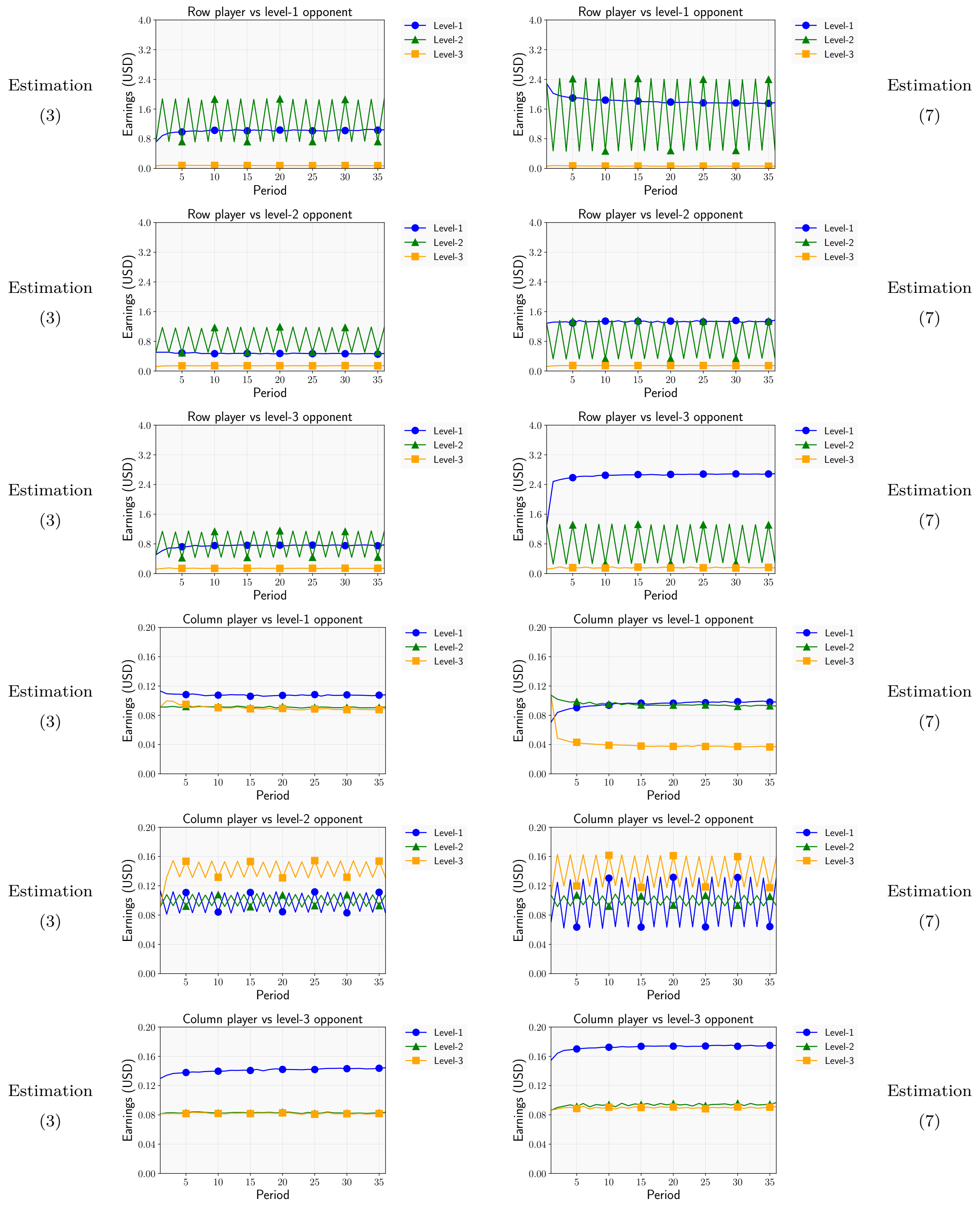


Figure B.15. Average earnings by rule, estimations (3) and (7)

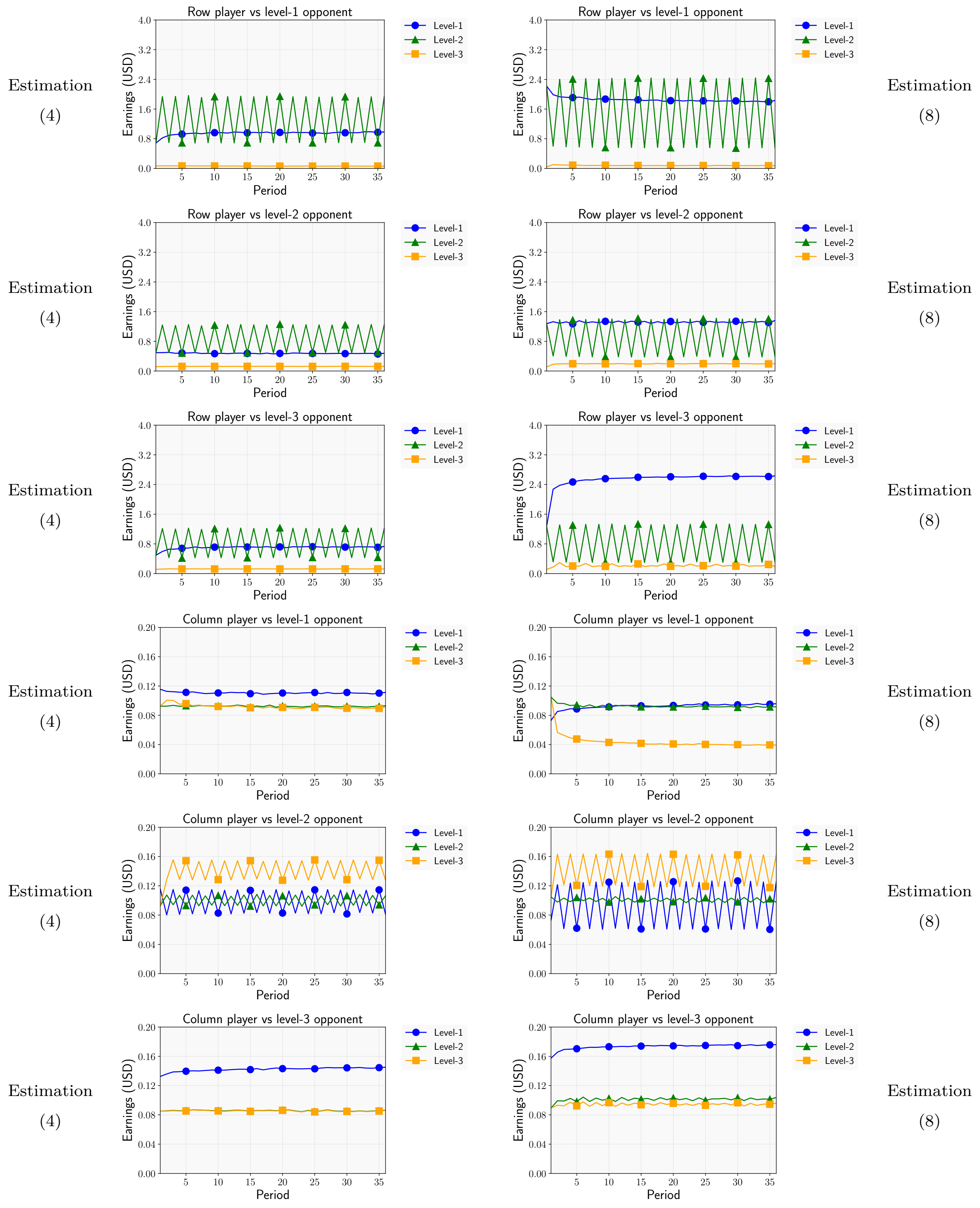


Figure B.16. Average earnings by rule, estimations (4) and (8)