Predicting Behavior in Games

DISSERTATION

zur Erlangung des akademischen Grades

Dr. Rer. Pol.
im Fach Wirtschaftswissenschaften

eingereicht an der
Universität Erfurt
Staatswissenschaftliche Fakultät
von
Wasilios Hariskos

urn:nbn:de:gbv:547-201400607

Dekan der Staatswissenschaftliche Fakultät:
Prof. Dr. André Brodocz

Gutachter:
1. Prof. Dr. Manfred Königstein
2. Prof. Dr. Guido Mehlkop

Eingereicht am: 19.12.2012
Tag der mündlichen Prüfung: 16.05.2013

Contents

List of Figures

List of Tables

1 Introduction

> "[T]he scientific gold standard is prediction. It is perfectly acceptable to propose a theory that fits existing experimental data and then to use the data to calibrate the parameters of the model. But, before using the theory in applied work, the vital next step is to state the proposed domain of application of the theory, and to make specific predictions that can be tested with data that wasn't used either in formulating the theory or in calibrating its parameters" (Binmore and Shaked, 2010).

My thesis contains four articles. Each article investigates the predictive accuracy of one or more behavioral game theory models for different games. Section 2 is based on a joint publication with Johannes Leder and Kinneret Teodorescu (Hariskos et al., 2011). The article is on learning models for predicting decisions from experience in forty market entry games. Each market entry game is a normal form game in which four individuals interact repeatedly over fifty trials. In each trial, each individual can either enter a risky market or stay out; her payoff depends on her own choice, the choices of the other three individuals (the more enter the lower is her payoff from entering the market) and the outcome of a binary gamble that is either high or low. The participants of the experiment do not know the payoff rule but receive after each trial complete individual feedback about their obtained payoff and their forgone payoff.

This prediction set of market entry games involves strategic and environmental uncertainty, complete feedback and minimizes the role of other factors besides experience that affect behavior in repeated games (e.g. framing, fairness, reciprocation and reputation); it was used for testing the predictive power of twenty-five different learning models that were submitted to the market entry prediction competition organized by Erev et al. (2010). The organizers of the competition provided moreover an estimation set of another forty market entry games and six baseline models that could be extended by the researchers that participated in the competition. The baseline models included reinforcement learning (Roth and Erev, 1995; Erev and Roth, 1998), normalized reinforcement learning (Erev et al., 1999), stochastic fictitious play (Goeree and Holt, 1999; Cheung and Friedman, 1997; Cooper et al., 1997; Fudenberg and Levine, 1998), normalized fictitious play (Ert and Erev, 2007), experience weighted attraction (Camerer and Ho, 1999) and the inertia, sampling and weighting model (Nevo and Erev, 2012).

We submitted three models to the market entry competition. All three models are based on the inertia, sampling and weighting (I-SAW) model. The I-SAW model assumes a fixed tendency to explore, a reliance on small experience samples, and strong inertia when the recent payoffs are not surprising. It is explicitly designed to capture eight well known behavioral tendencies from previous investigations of market entry games and individual decisions from experience that were replicated for the estimation set of market entry games (see Erev et al., 2010, and the literature cited therein): (1) the payoff variability effect, (2) high sensitivity to forgone payoffs, (3) excess entry, (4) underweighting of rare events, (5) surprise-triggers-change, (6) the very recent effect, (7) strong inertia and (8) individual differences.

Our models introduce four new assumptions: (A) a higher tendency for exploration at the beginning that decreases over time; (B) surprise as a factor influencing the weight of a trial in the sampling procedure; (C) the exclusion of unreliable experiences gained in early trials; and (D) the revision of a reasonable alternative that led to a very low payoff in the previous trial. We estimate the relative effect of each assumption and carry out a robustness test in order to clarify the usefulness of each assumption. Our main results can be summarized as follows: assumption A improves the predictions and appears to be robust, even beyond market entry games; assumption B improves the predictions and is in line with the von-Restorff-Effect and with animal research on the disruptive effect of surprising events on memory recall; assumption C improves the predictions and could be a result of "doubt about experiences in very early trials" or from memory limitation; whereas assumption D does not find supportive evidence.

In Section 3 I investigate and extend the predictions of the social preference model of Fehr and Schmidt (1999) for the public goods game. The public goods game is a normal form game in which two or more individuals interact. Each individual owns the same endowment and decides how much of it to contribute to the public good. The sum of contributions to the public good are multiplied by a factor greater one representing the positive externalities of the public good and are divided evenly among all individuals.

Although it is collectively best for everyone to contribute her complete endowment to the public good it is a dominant strategy for money-maximizing individuals not to contribute to the public good, i.e., free-riding is the unique Nash equilibrium (Nash, 1951). The result of free-riding is that nobody benefits from the positive externalities of the public good. This prediction is not very useful for predicting behavior of subjects in one-shot public goods experiments as, on average, about half of the endowment is contributed; with most subjects contributing either nothing or their complete endowment to the public good (Ledyard, 1995).

Fehr and Schmidt (1999) improve the Nash equilibrium prediction by assuming that individuals do care not only about their own income but also dislike unequal income distributions. They prove for the public goods game the existence of cooperation equilibria where all inequality averse individuals contribute symmetrically, i.e., the same amount of money, if the number of free riders is not too high.

Behind this background I show by means of a numerical example that cooperation equilibria exist where inequality averse individuals contribute different positive amounts of money to the public good (asymmetric cooperation). Thereby I demonstrate that the combination of the Nash equilibrium concept with the social preference model of inequality aversion predicts more heterogeneity in behavior for the public goods game than was thought previously.

Section 4 is joint work with Konstantinos Katsikopoulos and Gerd Gigerenzer. The article is on bargaining models for predicting decisions in one-shot ultimatum bargaining games. In a one-shot ultimatum bargaining experiment two subjects are paired anonymously together. The roles of the proposer and the responder are assigned randomly to the subjects, and the subjects play the ultimatum game only once. First, the proposer makes a proposal to the responder in which she determines how to divide a fixed amount of money provided by the experimenter. Then, the responder can either accept or reject the proposal. In case of acceptance, the proposal is implemented. However, if she rejects, both players get nothing.

In the subgame perfect equilibrium (Selten, 1965, 1973) of the ultimatum game, a money-maximizing responder accepts any positive amount of money, which is anticipated by a money-maximizing proposer, who therefore offers the smallest positive amount of money. However, contrary to the predictions of subgame perfect equilibrium, most participants split equally and reject low offers in ultimatum bargaining experiments (Güth et al., 1982; Camerer, 2003).

One explanation for the deviations between subgame perfect equilibrium predictions and observations made in ultimatum bargaining experiments is that people have other-regarding preferences that cannot be controlled by the experimenter. (Fehr and Schmidt, 1999; Bolton and Ockenfels, 2000; De Bruyn and Bolton, 2008). An alternative explanation for the deviations is that the framing of the ultimatum game triggers the use of simple decision rules that have evolved in real-life bargaining situations in which people know each other, bargain repeatedly and have the opportunity to change bargaining partners. In such an environment a decision rule to reject low offers or to split equally can be evolutionary stable. Thus, the initial play of inexperienced subjects in ultimatum bargaining experiments does not need to be close to the subgame perfect equilibrium (Gale et al., 1995).

While the proponents of the first explanation model the behavior of participants in one-shot games as if they (stochastically) optimize an other-regarding utility function, the proponents of the second explanation question that inexperienced subjects optimize anything at all (Gigerenzer, 2004; Binmore, 2007). Against this background, we construct a quantitative heuristic mix model in which the individuals base their decisions on simple rules of thumb. We then compare how well the heuristics model fits the outcomes of a two-person ultimatum game (Güth et al., 2003). In the next step, we compare the heuristic mix model to three inequality aversion models with respect to their predictive power for a three-person ultimatum game (Güth et al., 2007).[1]

The main results of our study are that (1) the heuristic mix fits the outcomes observed in the ultimatum bargaining experiment no worse than the inequality aversion models and that (2) the heuristic mix makes better out-of-sample predictions for the three-person ultimatum game.

Section 5 is based on joint work with Robert Böhm, Pantelis Pipergias Analytis and Konstantinos Katsikopoulos. The article is on equilibrium models and strategy mix models for predicting behavior in a broad set of one hundred and twenty extensive form games. Each game in the prediction set is a one-shot extensive form game of complete information with

[1] In the three-person ultimatum game the proposer allocates the monetary cake among three participants. The responder can either accept or reject the proposed allocation, and the so-called dummy (the third participant) has to accept the final allocation.

two individuals: a first mover and a second mover. The first mover can either choose a payoff distribution or let the second mover choose between two alternative payoff distributions.

The prediction set involves ten classes of games with different properties that are labeled safe shot, near dictator, common interest, costly help, trust, rational punish, costly punish, strategic dummy, free help and free punish; it was used for testing the predictive power of thirty-eight first mover models and thirty-nine second mover models that were submitted to the prediction competition for simple extensive form games. The competition was organized by (Ert et al., 2011) and was divided into two sub-competitions: one was on predicting the behavior of the first mover and the other one on predicting the behavior of the second mover.

The organizers of the competition provided an estimation set of another one hundred and twenty different extensive form games. They used the estimation set for fitting five baseline models for each sub-competition: the classic subgame perfect equilibrium (Selten, 1965, 1973), three stochastic variants of popular social preference models that they called inequality aversion (Fehr and Schmidt, 1999), equity reciprocity competition model (Bolton and Ockenfels, 2000; De Bruyn and Bolton, 2008), Charness and Rabin model (Charness and Rabin, 2002) and a new strategy mix model that consists of seven strategies (Ert et al., 2011). The surprising result was that the seven strategies model outperformed the popular social preference models.

Researchers that participated in the competition could use the baseline models together with the data of the estimation experiment for the development of their own models. After all models were submitted, the data of a prediction experiment were published. Based on this prediction set of games, all submitted models were ranked according to their mean predictive error and the ranking was published on the competition homepage.[2]

We submitted two models to the first mover competition and two models to the second mover competition. The first mover models were based on the discretized truncated subjective quantal response equilibrium model considered by Rogers et al. (2009). In both models we assume a slightly different heterogeneity in skill (i.e. preference responsiveness) and introduce heterogeneity in preferences (selfish versus other-regarding) and heterogeneity in preference beliefs (self-centered versus pessimistic). The heterogeneity in preferences is similar to the one assumed by Fehr and Schmidt (1999) and the heterogeneity in preference beliefs is inspired by the self-centered beliefs assumed in the seven strategies model and a very pessimistic strategy therein.

One second mover model is a stochastic social preference model. Compared to the social preference models that were used as baseline models, we assume a specification of social preferences that does not take the payoffs of a game directly into account. Instead the utility of a payoff distribution depends on whether it has certain characteristics relative to the other payoff distribution. The other second mover model is a lexicograpic strategy mix model that is based on a fast and frugal heuristic in the adaptive toolbox of bounded rationality (Gigerenzer and Selten, 2001): it is called take-the-best (Gigerenzer and Goldstein, 1996).

In our article we investigate four research questions: (1) Is it possible to achieve better fitting

[2] See `https://sites.google.com/site/extformpredcomp/competition-results-and`.

and prediction results by specifying and estimating different equilibrium models that were used by Ert et al. (2011)? Our results show that the gap between the seven strategies model and the equilibrium models in fitting the data of the estimation experiment is smaller than expected; however the seven strategies model still predicts the data of the prediction experiment better. (2) How good are the predictions of our submitted models in comparison to the baseline models? Our results show that our second mover models predict the choice behavior in the prediction set of games better than each baseline model and that our first mover models are only outperformed by the seven strategies model. (3) How reliable are the predictions results of the competition? We check how reliable the prediction results of the competition are by comparing them to predictions results of two different cross validations. Our results show that the ranking of the models may change in the cross validations if the prediction results in the competition are close and that only groups of models with similar results that differ considerably between groups do not change ranks. (4) How can we achieve better predictions by combining predictions of different models? Our results show that simple averaging of predictions of good models yields better predictions than each individual model and that optimal predictions are only obtained if predictions of semi-good models that are not highly correlated to the predictions of the good models are included.

2 Decisions from Experience in Market Entry Games

2.1 Introduction

We submitted three models to the market entry competition 2010 (see Section 2.2). All three models are based on the inertia, sampling and weighting model which we explain in Section 2.3. The models introduced four new assumptions. In the first model an adjustment process was introduced through which the tendency for exploration was higher at the beginning and decreased over time in the exploration stage. Another new assumption was that surprise as a factor influencing the weight of a trial in the sampling procedure was added. In the second model we added the possibility of an exclusion of unreliable experiences gained in the early trials of a game and the possibility of a revision of a reasonable alternative which was responsible for a very bad outcome in the previous trial. Three of the four added assumptions were combined in the third model. In Section 2.4 we describe the four additional assumptions we examined throughout the three models that we present in Section 2.5. Because each of our models contains at least two new assumptions, we estimated the relative effect of each assumption on the estimation and prediction scores and carried out a test of robustness in Section 2.6. In this way, we were able to clarify the usefulness of each added assumption. In Section 2.7 we summarize the analysis results and the theoretical conclusions.

2.2 Market Entry Prediction Competition

Erev et al. (2010) organized the market entry prediction competition with the motivation to improve our knowledge about the effect of experience on choice behavior in strategically and environmentally uncertain situations. The focus of the competition was on predicting decisions from experience in a broad set of repeated market entry games with a wide class of quantitative learning models. The participants that played the market entry games did not know the payoff rule (incentive structure) but were provided with feedback after each trial. This experimental design minimized the effect of other factors beside experience on the behavior of the participants (e.g. framing, fairness, reciprocation and reputation).

Validation Procedure The organizers Erev et al. (2010) provided experimental data on an estimation set of 40 market entry games (that are listed in Table 1) and some baseline models that were implemented in computer programs and fitted to the data of the estimation experiment.[1] The baseline models included reinforcement learning (Roth and Erev, 1995; Erev and Roth, 1998), normalized reinforcement learning (Erev et al., 1999), stochastic fictitious play (Goeree and Holt, 1999; Cheung and Friedman, 1997; Cooper et al., 1997; Fudenberg and Levine, 1998), normalized fictitious play (Ert and Erev, 2007), experience weighted attraction (Camerer and Ho, 1999) and the inertia, sampling and weighting model (Nevo and Erev, 2012). The baseline models served as a benchmark and could be used together with the data of the estimation experiment for the development of own models. After all models were submitted, the data of a prediction experiment were published. The prediction experiment contained another 40 different games that are listed in Table 2. Based on this prediction set of games, all submitted models were ranked according to validation criterion of the competition and the ranking was published on the competition homepage.[2]

Structure of the Basic Market Entry Game Each market entry game Γ is played by $n = 4$ individuals repeatedly for $R = 50$ trials. All individuals have the same action space $A = \{enter, not\ enter\}$. The risky action *enter* corresponds to "entering a risky market" and the safe action *not enter* corresponds to "staying out". The payoffs in trial t depend on the outcome of a binary gamble G_t, the number of entrants E_t and two game parameters k and S. G_t is equal to the high outcome H with probability $p \in [0, 1]$ and equal to the low outcome L otherwise. The payoff each individual i at trial t is equal to

$$V_{i,t} = \begin{cases} 10 - k \cdot E_t + G_t & \text{if enter} \\ \|G_t/S\| \text{ with } p = .5; - \|G_t/S\| \text{ otherwise} & \text{if not enter;} \end{cases}$$

and depends on her own choice, the choices of the other individuals $-i$ (strategic uncertainty; the more other individuals enter the lower is her payoff) and on the rounded value $\|\cdot\|$ of a ratio between a binary gamble G_t and a game parameter S (environmental uncertainty).[3]

Game Selection Algorithm Each market entry game Γ is characterized by different randomly determined parameters values (k, p, H, L, S). Game parameter k is drawn uniformly from $\{2, 3, ..., 7\}$ and game parameter S is drawn uniformly from $\{2, 3, ..., 6\}$. The high outcome H of the binary gamble G_t depends on a randomly generated number h drawn uniformly from $\{1, 2, ..., 10\}$ and a randomly generated number r_h drawn uniformly from the interval $[0, 1]$. If $r_h < 0.5$ then $H = h$; otherwise $H = 10 \cdot h$. The low outcome L of the binary gamble G_t

[1] The computer programs can be downloaded from the prediction competition homepage: `https://sites.google.com/site/gpredcomp/7-baseline-models`.
[2] See `https://sites.google.com/site/gpredcomp/8-competition-results-and-winners`.
[3] The participants of the estimation experiment and the participants of the prediction experiment did not know the exact payoff rule but received feedback after each trial. The feedback included their obtained payoff and their forgone payoff. The forgone payoff is the payoff that they would have received if they had chosen differently. Notice that the obtained payoff of entrants $10 - k \cdot E_t + G_t$ is larger than the forgone payoff of non-entrants $10 - k \cdot (E_t + 1) + G_t$. For a detailed description of the experimental method and instructions see Erev et al. (2010). The raw data of both experiments can be downloaded from the competition homepage under the following link: `https://sites.google.com/site/gpredcomp/raw-data`.

depends on a randomly generated number l drawn uniformly form the set $\{-10, -9, ..., -1\}$ and a randomly generated number r_l drawn uniformly from the interval $[0, 1]$. If $r_l < 0.5$ then $L = l$; otherwise $H = 10 \cdot l$. Probability p is equal to $-L/(H - L)$ rounded to the second decimal place.

Validation Criterion The competition focuses on the prediction of six statistics that are presented on the right-hand side of Table 2. The six statistics are the entry rate, the mean payoff (efficiency) and the alternation rate of each block of 25 trials (B1 denotes the first block and B2 the second one). The predictive power of each model is validated by means of a normalized mean squared error criterion. For each statistic the deviation score of a model is computed by (i) taking the squared deviation between the prediction of a model and the observation in each of the 40 games, (ii) by taking the mean squared deviation over the 40 games and (iii) by normalizing the mean squared deviation by the estimated error variance of the statistic (see Table 2). The validation criterion is the final score of a model that is calculated by taking the mean of the six normalized mean squared deviations (MSD; the lower the better). The estimation score MSD_{est} measures how well a model fits the data of the estimation set games and the prediction score MSD_{pre} measures how well a model predicts the data of the prediction set of games.

2.3 Inertia, Sampling and Weighting Model

Both, the estimation experiment and the prediction experiment are modeled as a series of $m = 40$ market entry games that are played by artificial individuals. A market entry game $\Gamma \in \{1, ..., m\}$ is characterized by different random values for its five parameters (k, p, H, L, S). The I-SAW model (Nevo and Erev, 2012) generates for each market entry game Γ a group of $N = 4$ individuals that play repeatedly for $R = 50$ trials. Each individual i is characterized by five traits whose values differ between individuals and are distributed uniformly with $\epsilon_i \sim U[0, .24]$, $\pi_i \sim U[0, .6]$, $\omega_i \sim U[0, .8]$, $\rho_i \sim U[0, .2]$, and $\mu_i \sim U\{1, 2, 3\}$. All individuals have the same action space $A = \{enter, not\ enter\}$ and each individual i has to choose in each round $t \in T = \{1, ..., R\}$ an action $a_{i,t} \in A$ without knowing how the other individuals will decide.

The decision process of each individual i is divided into three stages: exploration, inertia, and exploitation. Exploration implies to enter the market with probability $p^{enter} = 0.66$ or otherwise not to enter. The probability for an individual to explore is given by

$$p_i^{explore} = \begin{cases} 1 & \text{if } t = 1 \\ \epsilon_i & \text{if } t > 1. \end{cases}$$

If an individual does not explore, then she enters the second stage. Inertia implies to repeat the last action $a_{i,t} = a_{i,t-1}$ with probability

$$p_i^{inertia} = \pi_i^{Surprise^{t-1}} \in [\pi_i, 1], with\ Surprise^{t-1} \in [0, 1].$$

All individuals that have neither entered the exploration stage nor have decided in the inertia stage to repeat their last action, make their decision in the exploitation stage. In this stage each individual chooses the action $a_{i,t} \in A$ with the highest estimated subjective value (ESV).

Given the set of payoffs for all past cases $X(a_{i,past\ case}) = \{x(a_{i,1}), ..., x(a_{i,t-1})\}$ and the number of sample experiences or sample cases $\mu_i \sim U\{1, 2, 3\}$, the ESV of action $a_{i,t}$ for an individual i is given by the sum of two terms: the average payoff from all past cases weighted by $\omega_i \sim U[0, .8]$ and the average payoff from her set of sample cases $\{sample\ case^1, ..., sample\ case^{\mu_i}\}$ weighted by $(1 - \omega_i)$, i.e.,

$$ESV(a_{i,t}) = \omega_i \frac{\sum_{k=1}^{t-1} x(a_{i,k})}{t-1} + (1 - \omega_i) \frac{\sum_{l=1}^{\mu_i} x(a_{i,sample\ case^l})}{\mu_i}$$

where the sampling procedure for any sample case l is given by $sample\ case^l = t - 1$ with probability ρ_i and otherwise $sample\ case^l \sim U\{1, ..., t-1\}$.

2.4 Additional Assumptions

2.4.1 The Adjustment of Exploration over Time

In the I-SAW model, the probability to explore $p_i^{explore}$ equals ϵ_i if $t > 1$. The trait ϵ_i differs between individuals, but is constant within an individual throughout all trials of a game. However, it seems reasonable to assume that when faced with an unfamiliar environment, subjects will display higher explorative behavior at the beginning than after gaining some experience. As indicated by machine learning models, the change of exploration can be linear (Crook and Hayes, 2003; De Croon et al., 2005; Loch and Singh, 1998) or discontinuous by involving a switching point (Lee et al., 2011) . Moreover, research on repeated choice, shows that people repeat their choices, i.e. develop routines, when they repeat similar decisions (Betsch et al., 2001). A routine is described as a preference for a specific solution to a known problem. Thus, we introduced a higher exploration level at the beginning and a decrease of exploration with increasing numbers of trials. The decrease is modeled in four steps:

$$p_i^{explore} = \begin{cases} 1 & \text{if } t = 1 \\ 6 \cdot \epsilon_i / t & \text{if } 2 \leq t \leq 5 \\ \epsilon_i & \text{if } 6 \leq t \leq 30 \\ .9 \cdot \epsilon_i & \text{if } t \geq 31. \end{cases}$$

In the first trial the probability of individual i to explore $p_i^{explore}$ is 1. From trial two to trial six

$p_i^{explore}$ decreases with each trial and is equal to ϵ_i in trial six. In the trials six to 30, $p_i^{explore}$ is depending solely on the individual tendency for exploration ϵ_i. In trial 31 to trial 50, $p_i^{explore}$ is even lower than the individual tendency to explore, because we assumed that subjects would have adjusted to the context and choices should be even more less prone to randomness than in the trials before. Thus, the individual tendency to explore $p_i^{explore}$ is not only a function of the trait ϵ_i of individual i but also a function of the level of experience. It therefore captures additionally the adjustment process to a new environment.

2.4.2 The Recalling of Surprising Experiences

In the I-SAW model, when sampling (past) experiences, the most recent trial has a higher probability to be included in the sample due to the recency effect. All other past trials have the same probability to be sampled. However, studies concerning the von-Restorff-Effect (Von Restorff, 1933a) suggest that not all past experiences are equally likely to be included in the sample of experiences. It was found that stimulus items that are distinct from the general item pool are more apt to be recalled (Green, 1956; Hunt and Lamb, 2001; Von Restorff, 1933b). Furthermore early research on animal learning and the disruptive effect of surprising events on memory recall, found that surprising events lead to a lower rate of recall of events subsequent to the surprising one (Tulving, 1969). Therefore, we propose the influence of surprise on the sampling process in the exploitation stage. If the surprise term of a given trial $Surprise^{t-1}$ exceeds a threshold of .85 (according to fitted data), the probability to sample this trial for the calculation of the ESV is increased. To take the underweighting of rare events in decisions from experience (Barron and Erev, 2003; Hertwig et al., 2004) into consideration we limited this property to the last very surprising trial. Since the recency effect is assumed to vary across individuals, as indicated by trait ρ_i in the I-SAW model, we chose to use this parameter in order to depict surprise about a trial for the sampling process. Therefore, the last very surprising trial, has a higher probability to be sampled, and its probability to be sampled depends on the individual tendency to recall the most recent trial is equal to ρ_i.

2.4.3 The Exclusion of Very Early Trials from the Sample of Experiences

As previously noted, besides the most recent trial the sampling procedure of the I-SAW model assigns the same probability to be recalled to all other past trials. However, in the first trials of a new game, strategic uncertainty and uncertainty about the payoff rule is likely to be higher. Thus, early choices are more prone to randomness. This led us to the assumption, that later in the game, the participants should be more likely to question the reliability of the information gained through the very early trials of the game. In order to include this "doubt about experiences in very early trials" we introduced the following modification: Early experiences or cases are revised and can be excluded from the sample even if they are drawn at first during the sampling process. Revision implies that the individual repeats the sampling procedure for a given sample experience or sample case l if $sample\ case^l < 9$ once, repeats it a second time if $sample\ case^l < 7$, and again if $sample\ case^l < 5$, and again if $sample\ case^l < 3$.

This stepwise revision of her sampling decisions implies that an earlier *sample case*^{l} is excluded more likely from her set of sample cases.

2.4.4 The Influence of a Very Bad Experience in the Previous Trial

Imagine action $a_{i,t} = \textit{not enter}$ has the higher ESV in trial t, but in the previous trial this choice led to a very bad experience. In the I-SAW model the individual would have chosen simply the action with the higher ESV which is "not enter" which does not capture the affective reaction caused by negative experiences. However decisions are not only influenced by probability, but also by affective information (Loewenstein et al., 2001; Rottenstreich and Hsee, 2001; Glöckner and Hochman, 2011). Thus, we introduced the assumption that the individual revises her choice, although it has a higher ESV, if she made a very bad experience with it in the previous trial. This means that individual i revises her action if one of the two following sets of conditions is true:

First, if it is jointly true that $x(a_{i,t-1} = \textit{not enter} < 0$, $x(a_{i,t-1} = \textit{enter} > 0$ and $3 \cdot a_{i,t-1} = \textit{not enter} > -x(a_{i,t-1} = \textit{enter})$. Second, if it is jointly true that $x(a_{i,t-1} = \textit{not enter} > 0$, $x(a_{i,t-1} = \textit{enter} > 0$ and $3 \cdot a_{i,t-1} = \textit{not enter} < x(a_{i,t-1} = \textit{enter})$.

Revision implies to choose $a_{i,t} = \textit{enter}$ with probability $\lambda_i \sim U[0, 0.5]$ (a trait) and otherwise the action with the higher ESV which is $a_{i,t} = \textit{not enter}$. Note that the revision process is analogous if action $a_{i,t} = \textit{enter}$ has the higher ESV in trial t.

2.5 Description of Our Models and Their Performance in the Competition

2.5.1 Teodorescu et al. (2010)

The model of Teodorescu, Hariskos and Leder (2010) introduces two changes in the I-SAW model: First, the tendency for exploration is higher at the beginning and decreases over time in the exploration stage (2.4.1). Second, the last surprising trial is included with higher probability in the sampling of past cases in the exploitation stage (2.4.2). One of the main advantages of these suggested changes to the I-SAW model is that although it takes into account the changes of exploration over time and the effect of surprise on memory processes, it does not add any other traits than the ones estimated by the original I-SAW model.[4]

2.5.2 Hariskos et al. (2010)

The model of Hariskos, Leder and Teodorescu (2010) introduces two changes to the exploitation stage of the I-SAW model: First, very early trials are excluded with higher probability from

[4] The SAS code that was submitted to the competition is attached in Appendix 3.

the sample of experiences (2.4.3). Second, the affective reaction caused by negative experiences was addressed (2.4.4).[5]

2.5.3 Leder et al. (2010)

After simulating the first two models, we created a third model that includes a decreasing tendency to explore with increasing numbers of trials (2.4.1), the doubt about the reliability of experiences in very early trials (2.4.2), and the revision of a reasonable alternative given an associated very bad experience in the previous trial (2.4.4). We kept all parameters other than a slight change in the function determining the tendency to explore as depicted below[6]:

$$p_i^{explore} = \begin{cases} 1 & \text{if } t = 1 \\ 9 \cdot \epsilon_i / t & \text{if } 2 \leq t \leq 9 \\ .95 \cdot \epsilon_i & \text{if } 10 \leq t \leq 30 \\ .90 \cdot \epsilon_i & \text{if } t \geq 31. \end{cases}$$

2.5.4 The Models's Performance

Table 2.1 summarizes the performance of our models relative to the I-SAW model, once for the data of the estimation set (MSD_{est}) and once for the data of the prediction set (MSD_{pre}).

Table 2.1
The Performance of Our Models Relative to the Baseline Model

Section	Model	MSD_{est}	Δ_{est}	MSD_{pre}	Δ_{pre}
2.3	I-SAW	1.38		1.17	
2.5.1	Teodorescu et al.	1.35	-2.12%	1.16	-1.27%
2.5.2	Hariskos et al.	1.15	-16.33%	1.22	3.81%
2.5.3	Leder et al.	1.15	-16.07%	1.19	1.56%

All three models yield a better fit for the data from the estimation set than the I-SAW model. The fit of the first model (2.5.1) was slightly better than the I-SAW model and he fit of the other two models (2.5.2 and 2.5.3) were by far better. However, only the first model predicted the prediction data set better than the I-SAW model. In the following Section we will focus on this issue.

[5] The MATLAB code that was submitted to the competition is attached in Appendix 1.
[6] The MATLAB code that was submitted to the competition is attached in Appendix 2.

2.6 The Predictive Power of the Additional Assumptions

Since we added more than one assumption to the I-SAW model in each of our models, we cannot state the relative effect of each assumption individually. For this reason, we calculated the MSD scores after the competition by adding only one assumption to the I-SAW model (10,000 simulations) and summarized the relative effect of each assumption. The relative effect for the estimation score and prediction score is depicted in Table 2.2.

Table 2.2
The Relative Effect of Each Assumption on the Estimation and Prediction Score

Section	Assumption	MSD_{est}	Δ_{est}	MSD_{pre}	Δ_{pre}
2.3	I-SAW	1.38		1.17	
2.4.1	Exploration Over Time	1.35	-2.28%	1.17	-0.09%
2.4.2	Surprising Experiences	1.35	-2.20%	1.16	-1.12%
2.4.3	Very Early Trials	1.28	-7.31%	1.14	-3.18%
2.4.4	Bad Experience	1.23	-10.78%	1.25	6.27%

As depicted, each of our additional assumptions improved the estimation score. The first three assumptions (2.4.1, 2.4.2, and 2.4.3) also improved the prediction score. Whereas the fourth assumption (2.4.4), while leading to the largest improvement for the estimation set, impaired the prediction score, this clearly indicates over-fitting. Thus, we can conclude that the additional fourth assumption is responsible for the poor predictive performance of our second and third models. In order to examine whether the very small improvement that resulted from adding the first assumption (3.1) was not obtained by chance, we conducted an additional analysis.

One simple prediction of the decreasing exploration assumption is that in problems in which the best reply is relatively stable across trials, best reply behaviors are expected to become more common as time advances. On the other hand, constant exploration rate, as assumed by the original I-SAW model, predicts that in these cases, the frequency of best reply behaviors will remain constant over all trials. Problems 3 and 8 satisfy the relatively stable best reply requirement, since in these problems about 95% of the experiences yielded better payoffs for entering than staying out (obtained greater than forgone payoffs for entering and vice versa for staying out). Table 2.3 shows the percentages of best reply behaviors to previous trials for the first 12 trials:

Table 2.3
Best Reply Behavior to Previous Trials for Trial 1-12

Trial	2	4	6	8	10	12
Best Replies	75.0%	73.3%	86.7%	86.7%	91.7%	90%

The frequency of best reply behaviors increases with increasing numbers of trials, a result that cannot be explained by the original stable exploration assumption of the I-SAW model. Rather, these results can be captured by the assumption that the tendency to explore is higher in the first trials and decreases throughout the trials. Further support to the robustness of the decreasing exploration assumption can be found in the results of the following problem presented by Erev and Haruvy (forthcoming). In an experiment using what the authors call the clicking paradigm, subjects were asked to choose repeatedly between unlabeled keys on the computer screen. Pressing on one of the keys always resulted in a payoff of eight points and the other always resulted in a payoff of nine. As in the market entry game, after each trial subjects received information about the forgone payoff, in addition to their obtained payoff. The surprising result was that the proportion of choosing the clearly better option increased gradually during the first 10 trials before reaching 90%-100% in later trials (see Figure 4 in Nevo and Erev (2012)). Therefore, it seems that decreasing exploration over time is a robust phenomenon, even when collecting information actively is not needed and is counterproductive.

2.7 Summary

We examined four additional assumptions to the I-SAW model. The first assumption implies that the tendency for exploration is higher at the beginning and decreases over time in the exploration stage. Although it improved the predictions only slightly, we showed that this assumption appears to be robust, even beyond market entry games. The second assumption suggests that the last surprising trial needs to be included with higher probability in the sampling of past cases in the exploitation stage. This minor change consistently improved the predictions slightly, and is in line with the von-Restorff-Effect as well as with animal research on the disruptive effect of surprising events on memory recall. In the third additional assumption, we proposed that very early trials are excluded with higher probability from the sample of experiences. We suggested that this can be a result of “doubt about experiences in very early trials”, though one can argue that it might result also from memory limitation. It is important to note, that this additional assumption yields a high relative effect in the prediction and the estimation set, thus, we believe that future research should address its importance and its underling processes. The fourth assumption implies the revision of a reasonable alternative given an associated very bad experience in the previous trial. However, we did not find evidence to support this assumption; therefore, we concluded that the large improvement of the predictions for the estimated data set was the result of over fitting. We believe that the first three assumptions presented here address robust learning processes and are not only specific for market entry games. Future research is needed to determine the robustness and limitations of the above additional assumptions.

3 Inequality Aversion and Asymmetric Cooperation in Public Goods Games

3.1 Introduction

Everyone enjoys the benefits of public goods like national defense or knowledge. However, if a good is public, then by definition its consumption by one individual does not preclude its consumption by another individual. As a result, if one individual contributes to a public good, then all individuals will benefit irrespective of whether they contributed to the public good or not. Those incentives favor free-riding, a behavior in which one saves money by not contributing to the public good while hoping to enjoy the benefits created by those who contributed. However, if everyone thinks the same way, the public good will not be provided in the first place and therefore nobody is going to enjoy its benefits (Collel et al., 1995).

The public goods game is an abstract model of this real-life dilemma. It is a normal form game where two or more individuals interact in the following way: Each individual owns the same endowment and decides how much of it to contribute to the public good. The sum of contributions to the public good are multiplied by a factor greater one representing the positive externalities of the public good and are then divided evenly among all individuals. Although it is collectively best for everyone to contribute her complete endowment to the public good it is a dominant strategy for money-maximizing individuals not to contribute to the public good, i.e., free-riding is the unique Nash equilibrium (Nash, 1951). And thus, the result of free-riding is that nobody benefits from the positive externalities of the public good. However, in one-shot public goods experiments participants behave differently. They contribute, on average, about half of their endowment (Ledyard, 1995).

Behind this background, Fehr and Schmidt (1999) show that cooperation in a public goods game can be rationalized as a Nash equilibrium strategy if enough individuals are sufficiently inequality averse. In particular they prove the existence of cooperation equilibria in which one set of sufficiently inequality averse individuals (conditional cooperators) choose the *same positive* contribution level (symmetric cooperation) while another set of not sufficiently inequality averse individuals contribute nothing to the public good (free-riding). In this article we will extend their analysis by providing a simple numerical example that draws the attention to unknown equilibria in which the set of conditional cooperators contributes different positive amounts to the public good (asymmetric cooperation). For a more general analysis of asymmetric cooperation equilibria in N-player public good games see Hariskos and Königstein (2015).

The numerical example will be introduced in Section 3.2 after a formal treatment of the public goods game and the inequality aversion model of Fehr and Schmidt (1999). To keep things as simple as possible we will consider a public goods game with three different contribution levels – nothing, one euro, and two euros – and three sets of individuals: individuals with low inequality aversion that are always free riding and conditional cooperators with medium inequality aversion or high inequality aversion that contribute under certain conditions positive amounts of money to the public good.

After that we will use proposition 4 of Fehr and Schmidt (1999, p. 839) to derive the known equilibria for our numerical example (Section 3.3). Those are the defection equilibrium in which everyone is free riding and two symmetric cooperation equilibria in which each conditional cooperator with either medium or high inequality aversion contributes the same positive amount (either one euro or two euros) to the public good while each low inequality averse individual is free riding.

In Section 3.4 we will extend our analysis of the numerical example by showing that there is an additional asymmetric cooperation equilibrium. In this equilibrium only the low inequality averse individual is free riding while each conditional cooperator with medium inequality aversion contributes a positive amount that is lower than the amount contributed by each high inequality averse conditional cooperator. We conclude in Section 3.5.

3.2 Numerical Example

Public Goods Game In the public goods game $n \geq 2$ individuals decide simultaneously and privately how much to contribute to a public good G. Each individual $i \in \{1, ..., n\}$ owns the same endowment e, chooses a contribution level $c_i \in [0, e]$, and receives a monetary income $m_i = e - c_i + r \cdot G$ where the marginal return from the public good $G \equiv \sum_{j=1}^{n} c_j$ equals $r \in (\frac{1}{n}, 1)$.

Inequity Aversion The utility function of an inequality averse individual i for the public goods game is given by

$$u_i = e - c_i + r \cdot G - \frac{\alpha_i}{n-1} \sum_{i \neq j} max\{c_i - c_j, 0\} - \frac{\beta_i}{n-1} \sum_{i \neq j} max\{c_j - c_i, 0\}.$$

Her own income m_i is represented by the first three terms of her utility function. The next term captures her utility loss from disadvantageous income inequality (envy loss) and the last one her utility loss from advantageous income inequality (guilt loss).

The degree of aversion against an income difference to her advantage is captured by the guilt parameter $\beta_i \in [0, 1)$. Due to the non-negativity restriction she cannot gain utility from advantageous income inequality, and because of the restriction to values below one she does not burn money in order to reduce advantageous income inequality.

The degree of aversion against an income difference to her disadvantage is given by the envy parameter $\alpha_i \geq \beta_i$. Due to this parameter restriction she cannot suffer more from an advantageous income inequality in comparison to an equivalent disadvantageous income inequality.

Numerical Example Consider a public goods game with $n = 8$ individuals and a marginal return per capita of $r = \frac{13}{14}$. Each individual i has the same endowment of $e = 2$ and can pick one out of three different contribution levels, i.e., $c_i \in \{0, 1, 2\}$. The individuals have the same envy parameter $\alpha_i = 1$ but different guilt parameters. There is $n_l = 1$ individual with a guilt parameter of $\beta_i = 0$ (low inequality aversion), there are $n_m = 2$ individuals with a guilt parameter of $\beta_i = 0.45$ (middle inequality aversion), and there are $n_h = 5$ individuals with a guilt parameter of $\beta_i = 0.9$ (high inequality aversion) with $n = n_l + n_m + n_h$.

3.3 Symmetric Cooperation

The proposition of Fehr and Schmidt (1999, 839) for the public goods game consists of three parts. Part (a) states that it is a dominant strategy for an individual i to choose the lowest contribution level $c_i = 0$ if his guilt parameter is not sufficiently high, i.e., if $\beta_i < 1-r$. Applied to our numerical example, it means that only the individual with low inequality aversion has a dominant strategy. Part (b) of the proposition states that there is a unique defection equilibrium with $c_i = 0$ for all $i \in \{1, 2, ..., n\}$ if $k/(n-1) > r/2$ with $k \in \{0, 1, ..., n\}$ denoting the number of individuals with a dominant strategy. Since in our numerical example $k = 1$, $n = 8$ and $r = \frac{13}{14}$ we cannot conclude that there is a unique defection equilibrium. Thus we have to check part (c) of the proposition stating that other equilibria with positive contribution levels exist if $k/(n-1) < (r + \beta_i - 1)/(\alpha_i + \beta_i)$ for each individual i that does not have a dominant strategy. In these (cooperation) equilibria the individuals with a dominant strategy contribute nothing to the public good while all other individuals choose the same contribution level $c_i = \bar{c} \in \{0, 1, 2\}$. In our numerical the last part of the proposition holds for both medium and high inequality averse individuals. Therefore if $n_l = 1, n_m = 2, n_h = 5$, then three equilibria exist: a defection equilibrium $c^* = (c_l^* = 0, c_m^* = 0, c_h^* = 0)$ as well as two symmetric cooperation equilibria $c^* = (c_l^* = 0, c_m^* = 1, c_h^* = 1)$ and $c^* = (c_l^* = 0, c_m^* = 2, c_h^* = 2)$.

The simplicity of the numerical example makes it possible to confirm the above predictions by checking for each individual if the equilibrium payoff is higher than the two possible deviation payoffs. We will apply this deviation analysis to the three equilibria we derived so far as well as to the three equilibria that we are going to derive in the next section.

The results of the deviation analysis for the defection equilibrium are summarized in Table 3.1. In the defection equilibrium it is a best response for each individual to free ride if all other individuals are free-riding. A deviation to above either by one euro or by two euro does not increase the utility for each preference type l, m, h. Thus defection, i.e., $c^* = (c_l^* = 0, c_m^* = 0, c_h^* = 0)$ with $n_l = 1, n_m = 2, n_h = 5$, is a Nash equilibrium.

Table 3.1
Deviation Analysis for the Defection Equilibrium

(α_i, β_i)	$c_i = 0$	$c_i = 1$	$c_i = 2$
$(1, 0)$	$u_i(c_l^*) = 2$	$u_i(c_l^* + 1) = 0.93$	$u_i(c_l^* + 2) = -0.14$
$(1, 0.45)$	$u_i(c_m^*) = 2$	$u_i(c_m^* + 1) = 0.93$	$u_i(c_m^* + 2) = -0.14$
$(1, 0.9)$	$u_i(c_h^*) = 2$	$u_i(c_h^* + 1) = 0.93$	$u_i(c_h^* + 2) = -0.14$

The table displays for the public goods game with $e = 2$, $n = 8$ and $r = \frac{13}{14}$ the following information: the utility payoff for each preference type l, m, h for the equilibrium profile $c^* = (c_l^* = 0, c_m^* = 0, c_h^* = 0)$ with $n_l = 1, n_m = 2, n_h = 5$ and the utility payoff for each possible unilateral deviation from the equilibrium payoff $\Delta \in \{1, 2\}$ for l, m, h.

Given Table 3.2, it is a best response for the free rider l to contribute nothing given the conditional cooperators m and h contribute one euro to the public good. Furthermore, it is a best response for each conditional cooperator to contribute one euro given all other conditional cooperators contribute one euro while the free rider contributes nothing. Thus symmetric cooperation, i.e., $c^* = (c_l^* = 0, c_m^* = 1, c_h^* = 1)$ with $n_l = 1, n_m = 2, n_h = 5$, is a Nash equilibrium.

Table 3.2
Deviation Analysis for the Symmetric Cooperation Equilibrium I

(α_i, β_i)	$c_i = 0$	$c_i = 1$	$c_i = 2$
$(1, 0)$	$u_i(c_l^*) = 8.5$	$u_i(c_l^* + 1) = 8.43$	$u_i(c_l^* + 2) = 7.36$
$(1, 0.45)$	$u_i(c_m^* - 1) = 7.19$	$u_i(c_m^*) = 7.36$	$u_i(c_m^* + 1) = 6.29$
$(1, 0.9)$	$u_i(c_h^* - 1) = 6.8$	$u_i(c_h^*) = 7.36$	$u_i(c_h^* + 1) = 6.29$

The table displays for the public goods game with $e = 2$, $n = 8$ and $r = \frac{13}{14}$ the following information: the utility payoff for each preference type l, m, h for the equilibrium profile $c^* = (c_l^* = 0, c_m^* = 1, c_h^* = 1)$ with $n_l = 1, n_m = 2, n_h = 5$ and the utility payoff for each possible unilateral deviation from the equilibrium payoff $\Delta \in \{1, 2\}$ for l and $\Delta \in \{-1, 1\}$ for m, h.

Finally, we can infer from Table 3.3 that there is a second symmetric cooperation equilibrium in which the conditional cooperators contribute two euros to the public good, i.e., $c^* = (c_l^* = 0, c_m^* = 2, c_h^* = 2)$ with $n_l = 1, n_m = 2, n_h = 5$.

Table 3.3
Deviation Analysis for the Symmetric Cooperation Equilibrium II

(α_i, β_i)	$c_i = 0$	$c_i = 1$	$c_i = 2$
$(1, 0)$	$u_i(c_l^*) = 15$	$u_i(c_l^* + 1) = 14.93$	$u_i(c_l^* + 2) = 14.86$
$(1, 0.45)$	$u_i(c_m^* - 2) = 12.37$	$u_i(c_m^* - 1) = 12.54$	$u_i(c_m^*) = 12.71$
$(1, 0.9)$	$u_i(c_h^* - 2) = 11.60$	$u_i(c_h^* - 1) = 12.16$	$u_i(c_h^*) = 12.71$

The table displays for the public goods game with $e = 2$, $n = 8$ and $r = \frac{13}{14}$ the following information: the utility payoff for each preference type l, m, h for the equilibrium profile $c^* = (c_l^* = 0, c_m^* = 2, c_h^* = 2)$ with $n_l = 1, n_m = 2, n_h = 5$ and the utility payoff for each possible unilateral deviation from the equilibrium payoff $\Delta \in \{1, 2\}$ for l and $\Delta \in \{-2, -1\}$ for m, h.

So far we identified three equilibria of the public goods game: one defection equilibrium and two symmetric cooperation equilibria. Next we will show by means of the deviation analysis used so far that more equilibria exist.

3.4 Asymmetric Cooperation

In this section we will show that there is an asymmetric cooperation equilibrium in which conditional cooperators contribute positive amounts with middle inequality aversion (preference type m) contributing less than conditional cooperators with high inequality aversion (preference type h), i.e., given $n_l = 1, n_m = 2, n_h = 5$, there is an equilibrium candidate $c^* = (c_l^* = 0, c_m^* = 1, c_h^* = 2)$.

Table 3.4 summarizes the result of the deviation analysis for the asymmetric cooperation equilibrium in which the free rider l contributes nothing, each conditional cooperator m contributes one euro and each conditional cooperator h contributes two euros. Since no individual has an incentive to deviate unilaterally, asymmetric cooperation, i.e., $c^* = (c_l^* = 0, c_m^* = 1, c_h^* = 2)$ with $n_l = 1, n_m = 2, n_h = 5$, is a Nash equilibrium.

Table 3.4
Deviation Analysis for the Asymmetric Cooperation Equilibrium I

(α_i, β_i)	$c_i = 0$	$c_i = 1$	$c_i = 2$
$(1, 0)$	$u_i(c_l^*) = 13.14$	$u_i(c_l^* + 1) = 13.07$	$u_i(c_l^* + 2) = 12.71$
$(1, 0.45)$	$u_i(c_m^* - 1) = 11.51$	$u_i(c_m^*) = 11.68$	$u_i(c_m^* + 1) = 11.64$
$(1, 0.9)$	$u_i(c_h^* - 2) = 10$	$u_i(c_h^* - 1) = 10.56$	$u_i(c_h^*) = 10.57$

The table displays for the public goods game with $e = 2$, $n = 8$ and $r = \frac{13}{14}$ the following information: the utility payoff for each preference type l, m, h for the equilibrium profile $c^* = (c_l^* = 0, c_m^* = 1, c_h^* = 2)$ with $n_l = 1, n_m = 2, n_h = 5$ and the utility payoff for each possible unilateral deviation from the equilibrium payoff $\Delta \in \{1, 2\}$ for l, $\Delta \in \{-1, 1\}$ for m and $\Delta \in \{-2, -1\}$ for h.

3.5 Summary

An important result of standard economic theory is that cooperation in a public goods game cannot be rationalized as a Nash equilibrium if individuals care only about their own income. However, what we typically observe in one-shot public goods experiments is that participants behave differently. They contribute, on average, about half of their endowment (Ledyard, 1995). By introducing inequality aversion as an additional motive of decision making Fehr and Schmidt (1999) show that cooperation among sufficiently inequality averse individuals (conditional cooperators) is rational if there are not too many free riders. In those cooperation equilibria all conditional cooperators choose the same contribution level.

Behind this background, we provided a simple example which shows that the set of possible cooperation equilibria that follows from their inequality aversion theory may also include asymmetric cooperation equilibria in which conditional cooperators with lower inequality aversion contribute less than conditional cooperators with higher inequality aversion. Thus, the equilibrium analysis of the public goods game is much more complicated than expected if one allows for different degrees of inequality aversion. For future research it remains therefore to be shown under which general conditions asymmetric cooperation emerges in the N-player public good game (Hariskos and Königstein, 2015) and if asymmetric cooperation can be observed in the laboratory.

4 Heuristics and Ultimatum Bargaining Games

4.1 Introduction

In a one-shot ultimatum bargaining experiment two subjects are paired anonymously together, the roles of the proposer and the responder are assigned randomly to the subjects, and the subjects play the ultimatum game only once. The proposer makes first a proposal to the responder. The proposal determines how to divide a fixed amount of money provided by the experimenter. The responder can then either accept or reject the proposal. In case of acceptance, the proposal is implemented. However, if she rejects, both players get nothing. In the subgame perfect equilibrium (Selten, 1965, 1973) of the ultimatum game, a selfish responder accepts any positive amount of money. This is anticipated by a selfish proposer who offers therefore the smallest positive amount of money. However, in ultimatum bargaining experiments most participants split equally and reject low offers (Güth et al., 1982; Camerer, 2003).

One explanation for the deviations from subgame perfect equilibrium is that people have other-regarding preferences that cannot be controlled by the experimenter. The other-regarding preference models that we consider assume that individuals care about their own income and dislike inequality (Fehr and Schmidt, 1999; Bolton and Ockenfels, 2000; De Bruyn and Bolton, 2008). An alternative explanation for the deviations is that the framing of the ultimatum game triggers the use of simple decision rules that have evolved in real-life bargaining situations in which people know each other, bargain repeatedly and have the opportunity to change bargaining partners. In such an environment a decision rule to reject low offers or to split equally can be evolutionary stable. Thus, the initial play of inexperienced subjects in ultimatum bargaining experiments does not need to be close to the subgame perfect equilibrium (Gale et al., 1995).

While the proponents of the first explanation model the behavior of participants in one-shot games as if they would (stochastically) optimize an other-regarding utility function, the proponents of the second explanation question if inexperienced subjects optimize anything at all. They agree that social preference models are indeed more useful for fitting data than the subgame perfect equilibrium with selfish preferences, but disagree about their usefulness for predicting behavior and outcomes in other one-shot games (Gigerenzer, 2004; Binmore, 2007).

So far, nobody tested the predictive power of a quantitative rule-based model compared to a set of well established utility-based models for ultimatum bargaining games. Therefore we construct a heuristic mix model in which individuals base their decision on simple rules of thumb. Then, we compare how well the heuristic mix fits the outcomes of a two-person ultimatum experiment (Güth et al., 2003) in comparison to three inequality aversion models. Next, we assess the performance of the estimated models in predicting the outcomes of a three-person ultimatum experiment (Güth et al., 2007).[1] The main results of our study are that (1) the heuristic mix fits the outcomes of the two-person game no worse than the inequality aversion models and (2) and that it predicts the outcomes of the three-person game better.

In Section 4.2 we explain the heuristic mix that contains seven lexicographic strategies. We estimate the model with the individual data of the two-person ultimatum game and derive out-of-sample predictions for the three-person ultimatum game. In Section 4.3 we consider four utility-based models, the classic subgame perfect equilibrium with selfish preferences and three inequality aversion models. Finally, we discuss the main results and and conclude in Section 4.4.

4.2 Mix of Heuristics

Psychology and Economics have been coming steadily closer since the emergence of experimental and behavioral economics in the 1960 and the 1970s (Camerer and Ho, 1999). Conducting controlled experiments has become a handy tool that can be used to test the predictions of economic theories. The quick growth of experimental and behavioral traditions in economics has made individual data largely available within the discipline and has casted doubts on the rational maximizer model which until then has served as the economic stereotype of individual behavior. On the individual level the theories that were called to replace the profit maximizing agent maintained the basic idea that behavior could be represented by a utility function which was sensitive not only to profits but also to other variables such as equality (Berg and Gigerenzer, 2010).

Economics is well-known as being a science of aggregates. Thus, refining the models of individual behavior will hopefully lead to better models on the aggregate level. The power of utility function inspired models that dominate behavioral economics at the moment lies on their parametric flexibility which allows them to succinctly summarize data from individuals that follow very different behavioral rules. While these methods achieve relatively high behavioral fits they have the disadvantage of dissimulating what happens at the individual level and do not shed light at the decision process of the participants. Modelers disregard individuals differences that may exist in the decision processes of subjects in their experiments and attempt to fit the observed behavior to an overarching all-embracing model.

An alternative modeling approach that can be employed to bridge the conceptual void between the individual and the aggregate level is to employ models which consist several simple strategies. There are plenty of examples in which individual behavior has been shown to follow simple heuristic rules (Gigerenzer and Goldstein, 1996; Goldstein and Gigerenzer, 2002; Brandstätter et al., 2006). The heuristic mix models incorporate a number of simple rule of

[1] In the three-person ultimatum game the proposer allocates the monetary cake among three subjects including herself, the responder can either accept or reject the proposed allocation, and the third subject (the so called dummy) has nothing to say.

thumb strategies, which are derived directly from observations at the individual level, as the independent variables of the aggregate statistical model.

The modeler can then proceed by using a combination of optimization or extensive search methods and comparative model testing at the individual level to find a mix of strategies for which on the one hand the model performs well in predicting new data while on the other hand the correspondence between the observed individual behavior and the aggregate behavior is maintained. Thus the solution of the aggregate level should closely map the observed behavior on the individual level and vice versa. The heuristic mix presented in this Section is a first attempt to apply this alternative modeling approach to the context of ultimatum bargaining games.

Ultimatum Game In the ultimatum game that we consider, $N = 2$ individuals bargain about how to divide a monetary cake $c = 1000$ (Güth et al., 2003). The game is played in the strategy vector mode, i.e., each individual i decides first as a proposer P and then as a responder R. As a proposer she selects her demand $d_P \in \{100, 200, ..., 900\}$ that determines her offer to the other responder $o_{-R} = c - d_P$. As a responder she has to select a response alternative $a \in \{accept, reject\}$ for each possible offer $o_R \in \{100, 200, ..., 900\}$ that is determined by the demand of the other proposer $o_R = c - d_{-P}$. Next, the roles of proposer P and responder R are assigned randomly and the game is played according to the following rules: If R accepted proposal (d_{-P}, o_R), it is implemented; if R rejected, both get nothing, i.e., $(d_{-P}, o_R) = (0, 0)$.

Lexicographic Strategy There is an infinite population of individuals $i \in \{P, R\}$ that use the same lexicographic strategy:

1. Each individual i identifies the characteristic that is most important for her and selects the alternative that is best on this characteristic.
2. In the case of ties, she compares the tied alternatives on the next most important characteristic, and so on.
3. If there is no characteristic left in her set of characteristics, she chooses one of the tied alternatives randomly.

Like other formal models of fast and frugal heuristics in the adaptive toolbox of bounded rationality (Gigerenzer and Selten, 2001) the lexicographic strategy has three buildings blocks: (a) a simple search rule that specifies to consider characteristics step-by-step in the order of their importance; (b) a simple stopping rule that specifies to stop search if there are no ties or no characteristic left to consider; and (c) a simple decision rule that specifies – after search is stopped – to choose the best alternative on the last characteristic if there are no ties or to choose otherwise one of the tied alternatives randomly.

Proposer Sets of Characteristics There are three possible proposer sets of characteristics. Each proposer set is defined by

$$c_k^P = \begin{cases} \{d_P = c/N = o_{-R}\} & \text{if } k = 1 \\ \{d_P > c/N, min(d_P - o_{-R})\} & \text{if } k = 2 \\ \{o_{-R} > 0, max(d_P)\} & \text{if } k = 3. \end{cases}$$

A proposer with c_1^P considers only one characteristic. She selects the proposal that allocates c equally among N individuals. If there are tied proposals, she chooses one of the tied proposals randomly.

A proposer with c_2^P may consider two characteristics. She selects the proposal where she gets more than c/N and ignores the other characteristic. If there are tied proposals, she selects the tied proposal that minimizes the differences between her demand and what she offers to the responder. If there are still tied proposals, she chooses one of the tied proposals randomly.

A proposer with c_3^P may consider two characteristics. She selects the proposal where the responder gets something positive and ignores the other characteristic. If there are ties, she selects the tied proposal that maximizes her demand. If there are still ties, she chooses one of the tied proposals randomly.

Responder Sets of Characteristics There are three possible responder sets of characteristics. Each responder set is defined by

$$c_l^R = \begin{cases} \{o_R \geq o_{-R}, min(d_{-P})\} & \text{if } l = 1 \\ \{o_R \geq o_{-R} - c/10, min(d_{-P})\} & \text{if } l = 2 \\ \{max(o_R)\} & \text{if } l = 3. \end{cases}$$

For each proposal, a responder with c_1^R may consider two characteristics. If the first characteristic implies no ties, she selects the alternative where she gets at least what she offered and ignores the other characteristic. If there are ties on the first characteristic and no ties on the second one, she selects the alternative that minimizes the demand of the proposer. If there are still ties after considering the second characteristic, she chooses one of tied alternatives randomly.

A responder with c_2^R may consider two characteristics. If the first characteristic implies no ties, she selects the alternative where she gets at least what she offered minus $c/10$ and ignores the other characteristic. If there are ties on the first characteristic and no ties on the second one, she selects the alternative that minimizes the demand of the proposer. If there are still ties after considering the second characteristic, she chooses one of tied alternatives randomly.

A responder with c_3^R considers only one characteristic. If the first alternative implies no ties, she selects the alternative that maximizes her payoff; otherwise she chooses one of the tied alternatives randomly.

Distribution over Types We assume that the population consists of seven types. Each type $t \in \{A, B, ..., G\}$ is defined by her proposer set of characteristics c_k^P and her responder set of characteristics c_k^R. The behavior of each type is given by her demand d_t and her minimum acceptance threshold m_t. The distribution over types f depends on the distribution over proposer sets f^P and the distribution over responder sets f^R. Table 4.1 presents the seven types, their behavior and the distribution over the seven types.

Table 4.1
Types, Behavior of Types, and Distribution over Types

t	c_1^P	c_2^P	c_3^P
c_1^R	A	D	-
c_2^R	B	E	-
c_3^R	C	F	G

(d_t, m_t)	c_1^P	c_2^P	c_3^P
c_1^R	(500,500)	(600,400)	-
c_2^R	(500,400)	(600,300)	-
c_3^R	(500,100)	(600,100)	(900,100)

f_t	f_1^P	f_2^P	f_3^P
f_1^R	f_A	f_D	-
f_2^R	f_B	f_E	-
f_3^R	f_C	f_F	f_G

The heuristic used by G is equivalent to the subgame perfect equilibrium strategy used by a selfish individual (Selten, 1965, 1973). The other six strategies abstract behavioral tendencies that were proposed as explanations for observed deviations from the subgame perfect equilibrium strategy in ultimatum bargaining experiments (see below).

Proposer Decision Table 4.2 presents the lexicographic decision process of each type for the proposer decision .

Table 4.2
Ultimatum Game (UG): Type-Dependent Decision
Process for Proposer Decision

(d_P, o_{-R})	$A \vee B \vee C$	$D \vee E \vee F$	G
$(900, 100)$	0	1,0	1,1
$(800, 200)$	0	1,0	1,0
$(700, 300)$	0	1,0	1,0
$(600, 400)$	0	1,1	1,0
$(500, 500)$	1	0	1,0
$(400, 600)$	0	0	1,0
$(300, 700)$	0	0	1,0
$(200, 800)$	0	0	1,0
$(100, 900)$	0	0	1,0

The absence of a characteristic is indicated by a value of 0 and its presence by a value of 1. If a type considers more than one characteristic, then their values are separated by a comma.

The most important characteristic for A, B and C is that the responder gets the same share of the monetary cake. Therefore they select proposal $(500, 500)$ that is best on this characteristic. Their choice behavior depends on proposer set c_1^P that abstracts the behavioral tendency of people to use a simple rule called equality heuristic if there is an equal split among several alternatives. There is empirical support for the use of the equality heuristic in mini ultimatum games (Güth et al., 2001) and real-life allocation problems, e.g., the allocation of parental resources among children (Hertwig et al., 2002) or the allocation of monetary resources among investment options (Benartzi and Thaler, 2001).

The most important characteristic for D, E and F is that they get more than the c/N. Since there is more than one proposal that is best on this characteristic, they check the second most important characteristic that specifies to select the proposal that minimizes the difference between their demand and their offer. Therefore they select proposal $(600, 400)$ that is the only one of the tied proposals that is best on this characteristic. Their choice behavior depends on proposer set c_2^P that abstracts the behavioral tendency to offer more in the ultimatum game than in the dictator game (Forsythe et al., 1994) or the impunity game (Bolton and Zwick, 1995). In the latter games the proposer has not to fear rejection of low offers since the responder has to accept any proposal in the dictator game while in the impunity game her rejection is only symbolic and has no effect on the outcomes.

As mentioned above, the strategy applied by G is equivalent to the subgame perfect equilibrium strategy. The most important characteristic for her is that the responder gets something positive. This characteristic is common to all proposals, therefore she consults the second most important characteristic that specifies to select the proposal that maximizes her demand. Thus, she selects proposal $(900, 100)$ that is the only one of the tied proposals that is best on this characteristic.

Responder Decision Table 4.3 presents the decision process of each type for the responder decisions (see next page).

A and D accept the proposals where they get at least what they offered (based on the first characteristic) and reject the other ones (based on the second characteristic). Their choice behavior depends on their proposer set and responder set c_1^R. The latter is akin to the mirror heuristic that was used by Hertwig et al. (2012) for explaining the behavior of responders in mini-ultimatum games and abstracts the behavioral tendency to adopt one's own behavior as a benchmark in a situation with heterogeneous norms (López-Pérez, 2008).

B and E accept the proposals where they get at least what they offered minus fraction $c/10$ (based on the first characteristic) and reject the other ones (based on the second characteristic). Their choice behavior depends on their proposer set and responder set c_2^R. The latter abstracts the behavioral tendency to reject low offers due to angry feelings (Pillutla and Murnighan, 1996).

G, C and F accept any proposal based on the first characteristic in their responder set which abstracts the basic interest of income-maximization.

Table 4.3
Ultimatum Game: Decision Process for Responder Decisions

$a \to (d_{-P}, o_R)$	A	$B \vee D$	E	$C \vee F \vee G$
$accept \to (900, 100)$	0,0	0,0	0,0	1
$reject \to (0, 0)$	0,1	0,1	0,1	0
$accept \to (800, 200)$	0,0	0,0	0,0	1
$reject \to (0, 0)$	0,1	0,1	0,1	0
$accept \to (700, 300)$	0,0	0,0	1	1
$reject \to (0, 0)$	0,1	0,1	0	0
$accept \to (600, 400)$	0,0	1	1	1
$reject \to (0, 0)$	0,1	0	0	0
$accept \to (500, 500)$	1	1	1	1
$reject \to (0, 0)$	0	0	0	0
$accept \to (400, 600)$	1	1	1	1
$reject \to (0, 0)$	0	0	0	0
$accept \to (300, 700)$	1	1	1	1
$reject \to (0, 0)$	0	0	0	0
$accept \to (200, 800)$	1	1	1	1
$reject \to (0, 0)$	0	0	0	0
$accept \to (100, 900)$	1	1	1	1
$reject \to (0, 0)$	0	0	0	0

The absence of a characteristic is indicated by a value of 0 and its presence by a value of 1. If a type considers more than one characteristic, then their values are separated by a comma.

Predictions for the Ultimatum Game For each proposal, the predicted probability to be selected is given by

$$p_P(d_P) = \begin{cases} f_1^P & \text{if } d_P = 500 \\ f_2^P & \text{if } d_P = 600 \\ f_3^P & \text{if } d_P = 900 \\ 0 & \text{otherwise.} \end{cases}$$

For each proposal, the predicted probability to be accepted is given by

$$p_R(o_R) = \begin{cases} f_C + f_F + f_G & \text{if } o_R \leq 200 \\ 1 - f_A - f_B - f_D & \text{if } o_R = 300 \\ 1 - f_A & \text{if } o_R = 400 \\ 1 & \text{otherwise.} \end{cases}$$

Estimated Distribution over Types The distribution over types depends on the joint distribution over proposer and responder sets of characteristics. The estimated distribution over proposer sets of characteristics is given by $f^P = (0.56 \;\; 0.30 \;\; 0.14)$. The estimated distribution over responder sets of characteristics is given by

$$f^R(t) = \begin{cases} (0 \;\; 0 \;\; 1) & \text{if } t = G \\ (1/3 \;\; 1/3 \;\; 1/3) & \text{otherwise.} \end{cases}$$

We estimated the free parameters of the model based on observed numbers of individual choices that are consistent with the predictions of the model. The data were provided by Güth et al. (2003) who conducted the two-person ultimatum game that we consider in the German newspaper *Berliner Zeitung*. The researchers received 1035 complete decision forms. Table 4.4 presents the individual observations of 931 subjects that have a minimum acceptance offer (MAO).

Table 4.4
Observed Joint Behavior

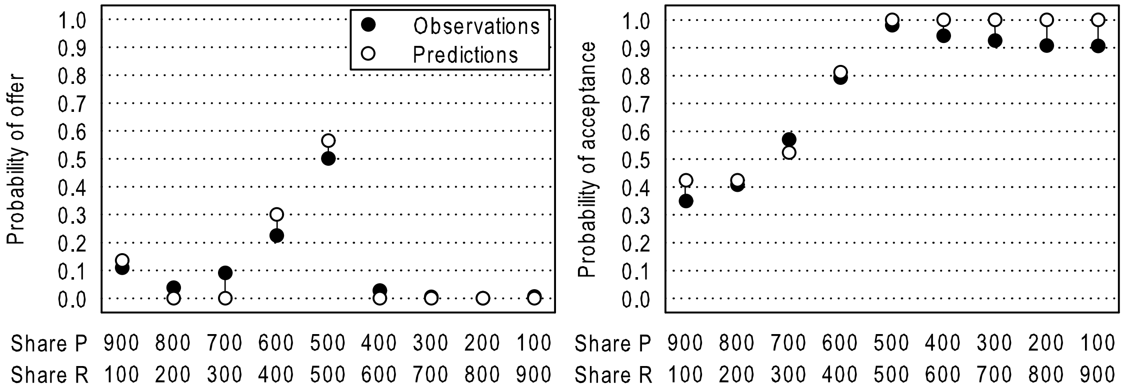
MAO	Demand								
	900	800	700	600	500	400	300	200	100
900	1	0	0	0	0	0	0	0	1
800	0	0	0	0	2	0	0	0	0
700	1	0	0	0	0	0	0	0	0
600	3	0	0	1	1	1	0	0	0
500	7	3	3	9	<u>114</u>	4	0	0	1
400	5	0	8	<u>84</u>	<u>111</u>	6	0	0	0
300	3	4	22	<u>52</u>	66	6	1	0	0
200	4	11	14	16	19	0	1	0	0
100	<u>86</u>	21	41	<u>55</u>	<u>134</u>	5	1	0	3

Source. Güth et al. (2003)
The underlined numbers of observations are consistent to the predictions of the heuristics model.

The estimated distribution over proposer sets f^P is equal to the distribution $(\frac{359}{n} \;\; \frac{191}{n} \;\; \frac{86}{n})$ over the observed behavior of $n = 636$ subjects that behaved consistently to the predictions of the model. $n_3 = 86$ subjects demand 900 for themselves and have a minimum acceptance threshold of 100 which implies $f^R(t = G) = (0\ \ 0\ \ 1)$. Since the distribution of $(\frac{198}{n-n_3}\ \frac{163}{n-n_3}\ \frac{189}{n-n_3})$ over the observed behavior of subjects that demand 500 or 600 for themselves and behave consistently to the predictions of the model is not significantly different from equal distribution ($\chi^2(2) = 3.6$, $p = 0.16$), we assume a distribution over responder sets that is equal to $f^R(t \neq G) = (\frac{1}{3}\ \frac{1}{3}\ \frac{1}{3})$.

Predictions vs Observations The observed and predicted offer or acceptance probabilities for each proposal of the ultimatum game are depicted in Figure 4.1.

Figure 4.1
Probability for Each Proposal to be Offered or Accepted in the Ultimatum Game, Observations vs Predictions of the Heuristic Mix



Source. Güth et al. (2003).

The heuristic mix predicts an average offer of 42% of the monetary cake and an average acceptance rate of 87%. The corresponding observations are 41% and 81%. The sum of absolute differences between actual and predicted probabilities are (i) for proposals to be offered $SAD_P = 0.33$, (ii) for proposals to be accepted $SAD_R = 0.49$ and (iii) for outcomes to be observed $SAD_{P\times R} = 0.35$. The correlations between actual and predicted probabilities are (i) for proposals to be offered $R_P = 0.98$, (ii) for proposals to be accepted $R_R = 0.99$ and (iii) for outcomes to be observed $R_{P\times R} = 0.99$.

Three-Person Ultimatum Game The three-person ultimatum that we consider contains $N = 3$ individuals who bargain about how to divide a monetary cake $c = 1200$ (Güth et al., 2007). The game is played in the strategy vector mode, i.e., each individual i decides first as a proposer P and then as a responder R. As a proposer she selects her demand $d_P \in \{0, 200, 400, 600, 800, 1000\}$ and her offer to the other responder $o_{-R} \in O = \{100, 200, 300, 400, 500, 600\}$. Both determine the remainder $r = c - d_P - o_{-R}$ that is allocated to the dummy D, with $r \in O$. As a responder she has to select a response alternative $a \in \{accept, reject\}$ for each possible proposal (d_{-P}, o_R, r) that is determined by the demand of the other proposer d_{-P} and offer o_R that is allocated to her. Next, the roles of proposer P, responder R and dummy D are assigned randomly and the game is played according to the following rules: If R accepted proposal (d_{-P}, o_R, r), it is implemented; if R rejected, all three go away empty handed, i.e., $(d_{-P}, o_R, r) = (0, 0, 0)$.

Demand and Minimum Acceptance Offer Table 4.5 presents for each type $t \in \{A, B, ..., G\}$ her demand d_t and her minimum acceptance threshold m_t.

Table 4.5
Three-Person Ultimatum Game: Type-Dependent
Demand and Minimum Acceptance Offer

(d_t, m_t)	c_1^P	c_2^P	c_3^P
c_1^R	(400,400)	(600,500)	-
c_2^R	(400,300)	(600,400)	-
c_3^R	(400,100)	(600,100)	(1000,100)

The lexicographic decision process of each type is presented for the proposer decision in Table 3 and for the responder decision in Table 4 (see Appendix refC).

Predictions for the Three-Person Ultimatum Game For each proposal, the predicted probability to be selected is given by

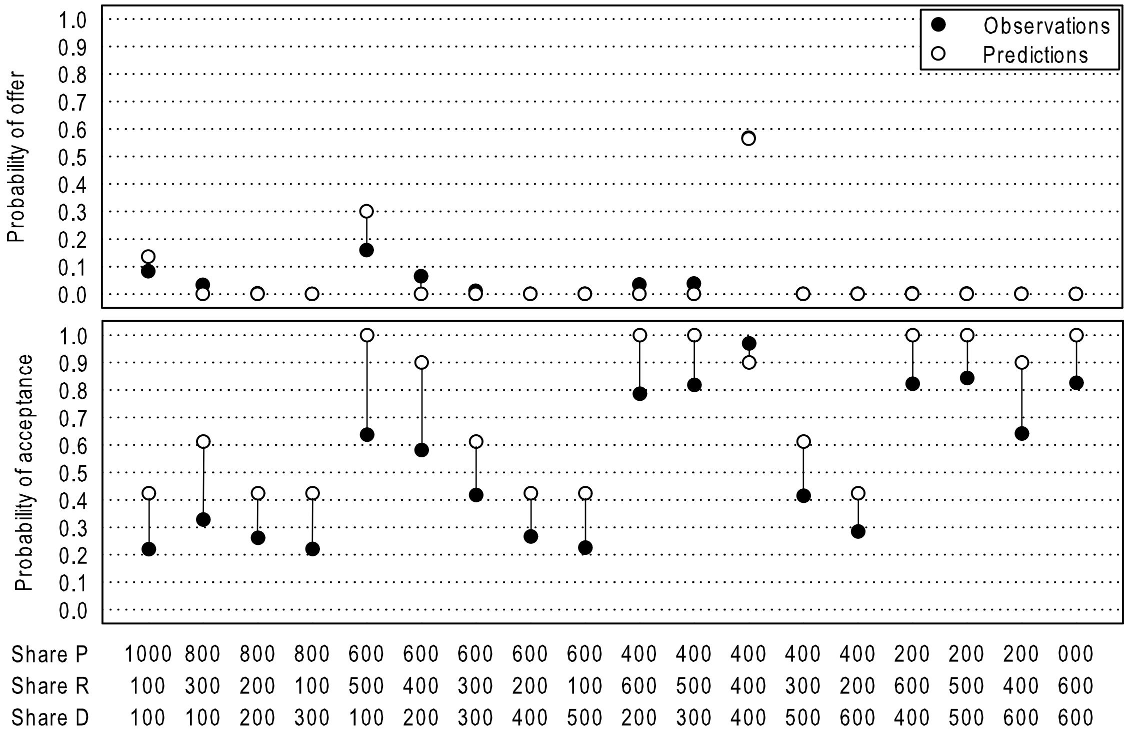
$$p_P(d_P, o_{-R}) = \begin{cases} f_1^P & \text{if } (d_P, o_{-R}) = (400, 400) \\ f_2^P & \text{if } (d_P, o_{-R}) = (600, 500) \\ f_3^P & \text{if } (d_P, o_{-R}) = (1000, 100) \\ 0 & \text{otherwise.} \end{cases}$$

For each proposal, the predicted probability to be accepted is given by

$$p_R(o_R) = \begin{cases} f_C + f_F + f_G & \text{if } o_R \leq 200 \\ 1 - f_A - f_D - f_E & \text{if } o_R = 300 \\ 1 - f_D & \text{if } o_R = 400 \\ 1 & \text{otherwise.} \end{cases}$$

Predictions vs Observations The observed and predicted offer or acceptance probabilities for each proposal of the three-person ultimatum game are depicted in Figure 4.2. The heuristic mix predicts an average offer of 32% of the monetary cake and an average acceptance rate of 87%. The corresponding observations are 33% and 79%. The sum of absolute differences between actual and predicted probabilities are (i) for proposals to be offered $SAD_P = 0.39$, (ii) for

Figure 4.2
Probability for Each Proposal to be Offered or Accepted in the Three-Person Ultimatum Game, Observations vs Predictions of the Heuristic Mix



Source. Güth et al. (2007).

proposals to be accepted $SAD_R = 3.65$ and (iii) for outcomes to be observed $SAD_{P\times R} = 0.58$. The correlations between actual and predicted probabilities are (i) for proposals to be offered $R_P = 0.96$, (ii) for proposals to be accepted $R_R = 0.94$ and (iii) for outcomes to be observed $R_{P\times R} = 0.92$.

4.3 Equilibrium Models

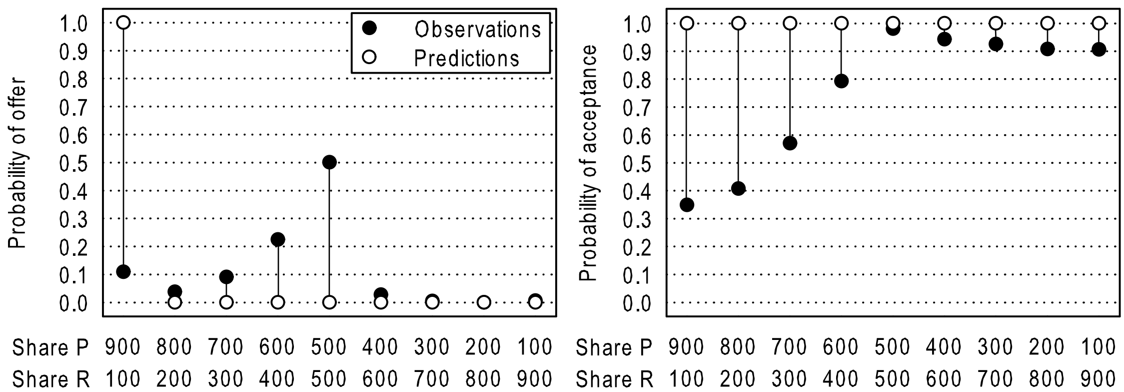
4.3.1 Subgame Perfect Equilibrium

Subgame perfect equilibrium (SPE) is a refinement of Nash equilibrium (Nash, 1951) that was introduced by Selten (1965, 1973). In the SPE model each individual i maximizes her expected utility (*A1*), her preferences are represented by utility function u_i (*A2*) that depends only on her income x_i (*A3*), assumptions *A1*–*A3* are common knowledge (*A4*) and she uses backward induction to solve the game (*A5*).

Predictions Given *A1-A4*, the proposer faces an equilibrium selection problem because for each minimum acceptance offer of the responder, there is a Nash equilibrium $(o_P, m_{-R} = o_P)$ in which the proposer offers the minimum acceptance offer of the responder. However, if the proposer uses backward induction to solve the ultimatum game, she eliminates Nash equilibria that depend on non-credible threats. Given *A1-A5*, a threat to reject positive offers of a responder who is only interested in her own income is not credible because it implies that she would leave money on the table. Thus, eight of the nine Nash equilibria of the ultimatum game with $m_{-R} > 100$ depend on non-credible threats, i.e., the only SPE is $(o_P, m_{-R}) = (100, 100)$. The same reasoning applies to the three-person ultimatum game: A selfish responder would never leave money on the table and accepts therefore any positive offer. A selfish proposer anticipates this behavior and makes therefore the lowest positive offer. Consequently, the unique SPE of the three-person ultimatum game is $((d_P, o_P), m_{-R}) = ((1000, 100), 100)$.

Observations vs Predictions The observed and predicted offer or acceptance probabilities for each proposal of the ultimatum game are depicted in Figure 4.3. The SPE model predicts

Figure 4.3
Probability for Each Proposal to be Offered or Accepted in the Ultimatum Game, Observations vs Predictions of the SPE Model

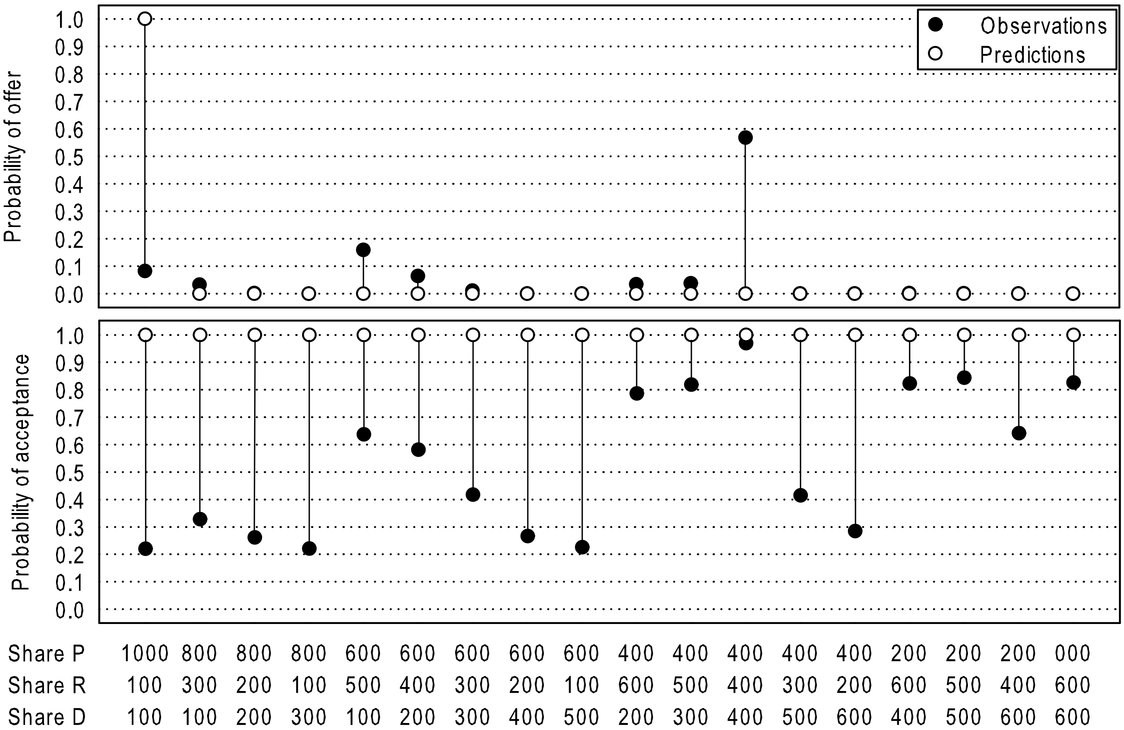


Source. Güth et al. (2003).

an average offer of 10% of the monetary cake and an average acceptance rate of 100%. The corresponding observations are 41% and 81%. The sum of absolute differences between actual and predicted probabilities are (i) for proposals to be offered $SAD_P = 1.78$, (ii) for proposals to be accepted $SAD_R = 2.22$ and (iii) for outcomes to be observed $SAD_{P \times R} = 1.92$.

The observed and predicted offer or acceptance probabilities for each proposal of the three-person ultimatum game are depicted in Figure 4.4. The SPE model predicts an average offer of 8% of the monetary cake and an average acceptance rate of 100%. The corresponding observations are 33% and 79%. The sum of absolute differences between actual and predicted probabilities are (i) for proposals to be offered $SAD_P = 1.83$, (ii) for proposals to be accepted $SAD_R = 8.44$ and (iii) for outcomes to be observed $SAD_{P \times R} = 1.96$.

Figure 4.4
Probability for Each Proposal to be Offered or Accepted in the Three-Person Ultimatum Game, Observations vs Predictions of the SPE Model



Source. Güth et al. (2007).

4.3.2 Subgame Perfect Equilibrium with Fehr-Schmidt Preferences

Fehr and Schmidt (1999) propose an alternative model for bargaining behavior in ultimatum games that relaxes two assumptions of the traditional SPE model. In their behavioral model the individuals do not care only about their own income (*A3*) but also dislike unequal income distributions. Moreover, they do not have identical preferences (*A2*) but differ in their degree of inequality aversion.

Inequality Aversion An inequality averse individual i still cares about her own income x_i but is also altruistic towards each other individual j with a lower income ($x_j < x_i$) or feels envy if she has a lower income ($x_i < x_j$). The utility function of an inequality averse individual i is given by

$$u_i(x) = x_i - \frac{\alpha_i}{N-1}\sum_{i \neq j} max\left\{x_j - x_i, 0\right\} - \frac{\beta_i}{N-1}\sum_{i \neq j} max\left\{x_i - x_j, 0\right\}. \tag{4.1}$$

The first term captures her utility from own income, the second term her utility loss from disadvantageous income inequality, and third term her utility loss from advantageous income inequality. The degree of aversion against advantageous inequality is given by $\beta_i \in [0, 1)$. The non-negativity restriction implies that i cannot gain utility from advantageous inequality and the restriction to values below one implies that she does not burn money to reduce advantageous inequality. The degree of aversion to disadvantageous inequality is measured by $\alpha_i \geq \beta_i$. The parameter restriction means that the individual cannot suffer more from an advantageous inequality in comparison to an equivalent disadvantageous inequality.

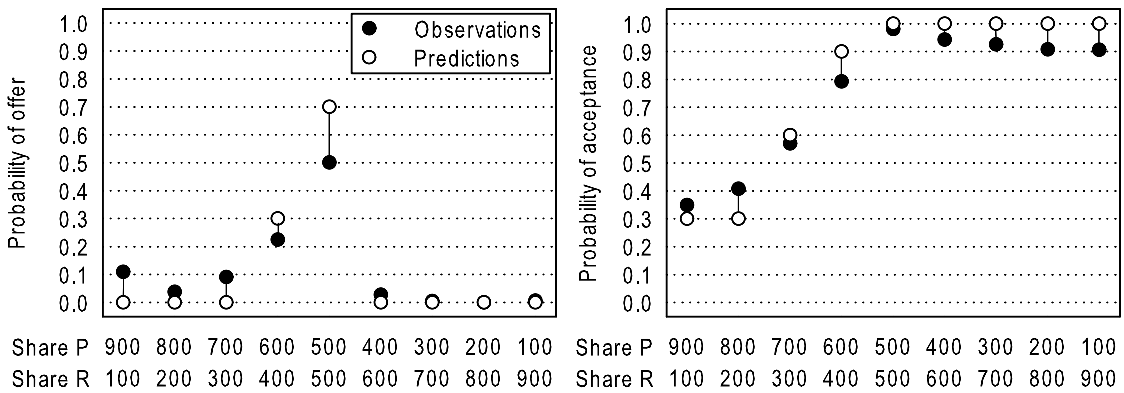
Heterogeneity in Preferences In their model, Fehr and Schmidt (1999) assume a distribution over four types of individuals. Each type t is characterized by a combination of (α_t, β_t) values. The parametrization of their model is given in Table 4.6.

Table 4.6
Distribution of Preferences in the Fehr-Schmidt SPE Model

Type t	Share s	α_t	β_t
A	0.3	0	0
B	0.3	0.5	0.25
C	0.3	1.0	0.60
D	0.1	4.0	0.60

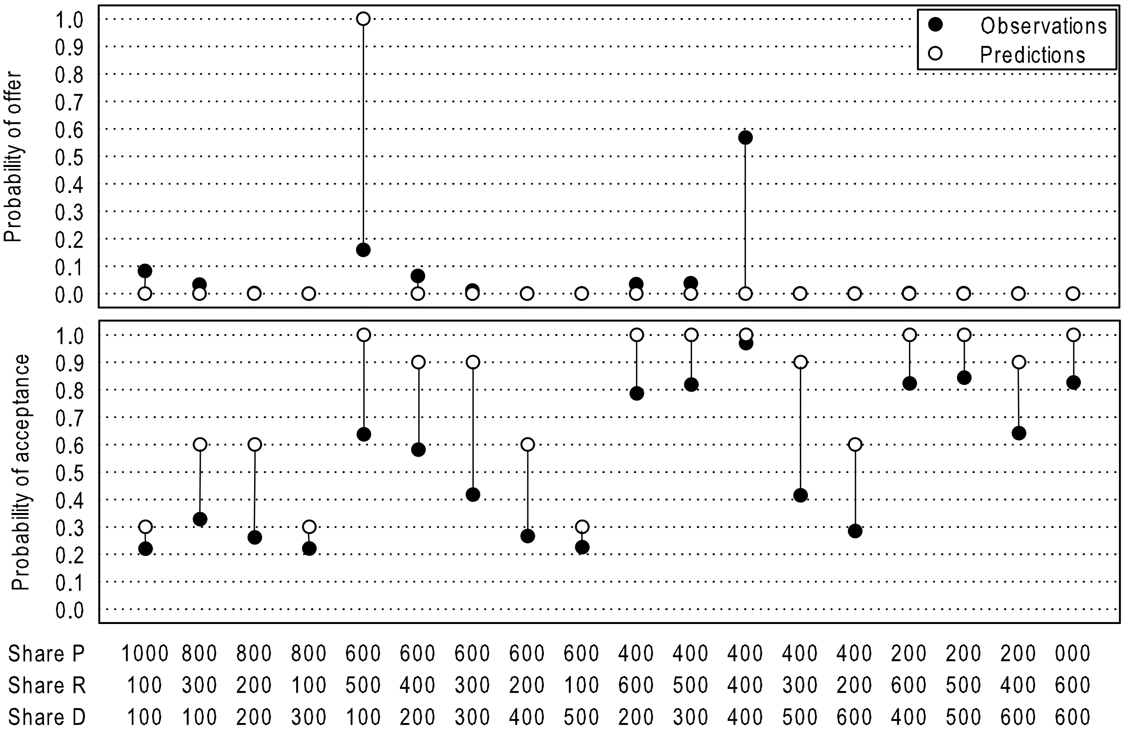
An individual of type A is selfish and cares only about her own income. B, C and D care about their own income but also about the equality of the income distribution with D being more inequality averse than C; and C being more inequality averse than B.

Figure 4.5
Probability for Each Proposal to be Offered or Accepted in the Ultimatum Game, Observations vs Predictions of the Fehr-Schmidt SPE Model



Source. Güth et al. (2003).

Figure 4.6
Probability for Each Proposal to be Offered or Accepted in the Three-Person Ultimatum Game, Observations vs Predictions of the Fehr-Schmidt SPE Model



Source. Güth et al. (2007).

Predictions The Fehr-Schmidt model predicts that there is a unique equilibrium of the ultimatum game in which (i) A accepts at least 100 and offers 400, (ii) B accepts at least 300 and offers 500, (iii) C accepts at least 400 and offers 500 and (iv) D accepts at least and offers 500. In the unique equilibrium of the three-person ultimatum game each type offers the $(600, 500)$ proposal while A rejects no proposal, B rejects proposals with $o_P = 100$, C rejects proposals with $o_P \leq 200$ and proposal $(800, 300)$, and D rejects proposals $o_P \leq 400$ except the equal split of $(400, 400)$.

Observations vs Predictions The observed and predicted offer or acceptance probabilities for each proposal of the ultimatum game are depicted in Figure 4.5. The model predicts an average offer of 47% of the monetary cake and an average acceptance rate of 97%. The corresponding observations are 41% and 81%. The sum of absolute differences between actual and predicted probabilities are (i) for proposals to be offered $SAD_P = 0.55$, (ii) for proposals to be accepted $SAD_R = 0.63$ and (iii) for outcomes to be observed $SAD_{P\times R} = 0.60$. The correlations between actual and predicted probabilities are (i) for proposals to be offered

$R_P = 0.97$, (ii) for proposals to be accepted $R_R = 0.99$ and (iii) for outcomes to be observed $R_{P\times R} = 0.98$.

The observed and predicted offer or acceptance probabilities for each proposal of the three-person ultimatum game are depicted in Figure 4.6.The model predicts an average offer of 42% of the monetary cake and an average acceptance rate of 100%. The corresponding observations are 33% and 79%. The sum of absolute differences between actual and predicted probabilities are (i) for proposals to be offered $SAD_P = 1.68$, (ii) for proposals to be accepted $SAD_R = 4.34$ and (iii) for outcomes to be observed $SAD_{P\times R} = 1.80$. The correlations between actual and predicted probabilities are (i) for proposals to be offered $R_P = 0.19$, (ii) for proposals to be accepted $R_R = 0.87$ and (iii) for outcomes to be observed $R_{P\times R} = 0.14$.

4.3.3 Quantal Response Equilibrium with Fehr-Schmidt Preferences

De Bruyn and Bolton (2008) propose a model for repeated bargaining games that combines a variant of equity, reciprocity, and competition (ERC) preferences (Eq. 4.4) with a quantal response equilibrium (QRE, McKelvey and Palfrey, 1998). They fit their model to ultimatum game data and assess how well it performs in predicting behavior in other bargaining games in comparison to an alternative specification in which they use Fehr-Schmidt preferences (Eq. 4.1). The Fehr-Schmidt QRE model that we present in this Section is an adaptation of the latter model to one-shot games in which learning is not possible. It relaxes three assumptions of the traditional SPE model: First, the individuals do not care only about their own income (*A3*) but also dislike unequal income distributions. Second, the individuals maximize their expected utility (*A1*) but they do it stochastically which leads to deviations from optimal play that are interpreted as mistakes. Third, the proposer does not use backward induction (*A5*) to solve the game.

Predictions For each proposal $x^k = (x_1, ..., x_N)$ with $k \in \{1, ..., K\}$, the probability that responder R accepts proposal x^k is given by

$$p_R(x^k) = \frac{e^{\lambda_R u_R(x^k)}}{e^{\lambda_R u_R(\oslash)} + e^{u_R(x^k)}} = \frac{e^{\lambda_R u_R(x^k)}}{1 + e^{\lambda_R u_R(x^k)}}. \tag{4.2}$$

By definition, $p_R(x^k) \in [0, 1]$ for all x^k. R's utility from accepting proposal x^k equals $u_R(x^k)$ while her utility from rejecting is $u_R(\oslash) = 0$ – since the monetary cake shrinks to $c = 0$. The coefficient of certitude $\lambda_R \in [0, \infty]$ indicates the choice consistency of R. The larger λ_R, the higher is the probability that R chooses the action that maximizes her utility. This implies that if $\lambda_R = 0$, R acts randomly and that if $\lambda_R = \infty$, R maximizes always her utility.

The probability that proposer P offers proposal x^k is given by

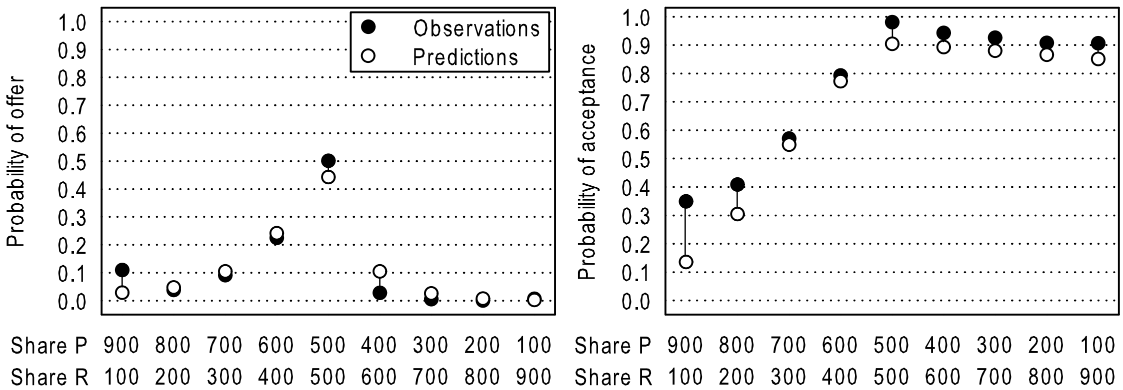
$$p_P(x^k) = \frac{e^{\lambda_P E(u_P(x^k))}}{\sum_{j=1}^{K} e^{\lambda_P E(u_P(x^j))}} = \frac{e^{\lambda_P p_R(x^k) u_P(x^k)}}{\sum_{j=1}^{K} e^{\lambda_P p_R(x^j) u_P(x^j)}}. \tag{4.3}$$

By definition, $p_P(x^k) \in [0,1]$ for all x^k and $\sum p_P(x^k) = 1$. Since P knows each acceptance probability $p_R(x^k)$, her expected utility $E(u_P(x^k))$ of proposing x^k to R is equal to $p_R(x^k)u(x^k)$.

We fit the Fehr-Schmidt QRE model to the ultimatum game data provided by Güth et al. (2003) with maximum likelihood estimation. The log-likelihood of the model is -1993 and its parameter estimates (and standard deviations) are $\alpha_P = \alpha_R = 0.6414$ (0.0528), $\beta_P = \beta_R = \alpha_P$, $\lambda_P = 0.0069$ (0.0005) and $\lambda_R = 0.0045$ (0.0003).[2] The predictions for the ultimatum game with $N = 2$ or $N = 3$ can be computed by inserting the parameter estimates into Eq. 4.1, Eq. 4.2 and Eq. 4.3.

Observations vs Predictions The observed and predicted offer or acceptance probabilities for each proposal of the ultimatum game are depicted in Figure 4.7.

Figure 4.7
Probability for Each Proposal to be Offered or Accepted in the Ultimatum Game, Observations vs Predictions of the Fehr-Schmidt QRE Model



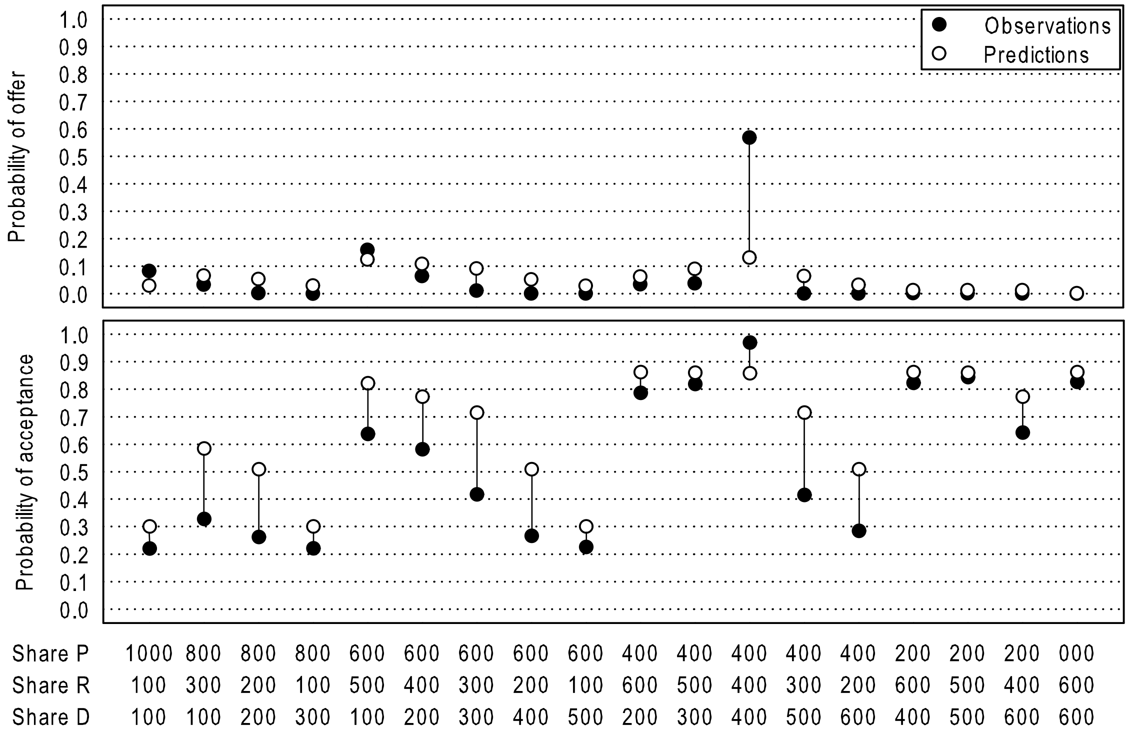
Source. Güth et al. (2003).

The model predicts an average offer of 45% of the monetary cake and an average acceptance rate of 78%. The corresponding observations are 41% and 81%. The sum of absolute differences between actual and predicted probabilities are (i) for proposals to be offered $SAD_P = 0.29$, (ii) for proposals to be accepted $SAD_R = 0.63$ and (iii) for outcomes to be observed $SAD_{P\times R} = 0.35$. The correlations between actual and predicted probabilities are (i) for proposals to be offered $R_P = 0.96$, (ii) for proposals to be accepted $R_R = 0.99$ and (iii) for outcomes to be observed $R_{P\times R} = 0.97$.

The observed and predicted offer or acceptance probabilities for each proposal of the three-person ultimatum game are depicted in Figure 4.8. The model predicts an average offer of 32%

[2] We restricted the values of the inequality aversion parameters to $\alpha_i \geq \beta_i$. Notice that the results of the unconstrained maximum likelihood estimation with a log-likelihood of -1993, $\alpha_P = \alpha_R = 0.6335$ (0.0649), $\beta_P = 0.6497$ (0.0667), $\lambda_P = 0.0069$ (0.0005) and $\lambda_R = 0.0045$ (0.0003) are very close.

Figure 4.8
Probability for Each Proposal to be Offered or Accepted in the Three-Person Ultimatum Game, Observations vs Predictions of the Fehr-Schmidt QRE Model



Source. Güth et al. (2007).

of the monetary cake and an average acceptance rate of 73%. The corresponding observations are 33% and 79%. The sum of absolute differences between actual and predicted probabilities are (i) for proposals to be offered $SAD_P = 1.05$, (ii) for proposals to be accepted $SAD_R = 2.63$ and (iii) for outcomes to be observed $SAD_{P\times R} = 1.06$. The correlations between actual and predicted probabilities are (i) for proposals to be offered $R_P = 0.63$, (ii) for proposals to be accepted $R_R = 0.91$ and (iii) for outcomes to be observed $R_{P\times R} = 0.63$.

4.3.4 Quantal Response Equilibrium with ERC Preferences

The equity, reciprocity and competition (ERC) QRE model is an adaption of the De Bruyn and Bolton (2008) for repeated bargaining games to the context of one-shot games. Compared to the FS QRE model, the ERC QRE model differs in the specification of inequality aversion, i.e., the individuals care about their own income and dislike to get less than the equal share of $1/N$.

Inequality Aversion The ERC preferences of each individual $i \in \{P, R\}$ are given by

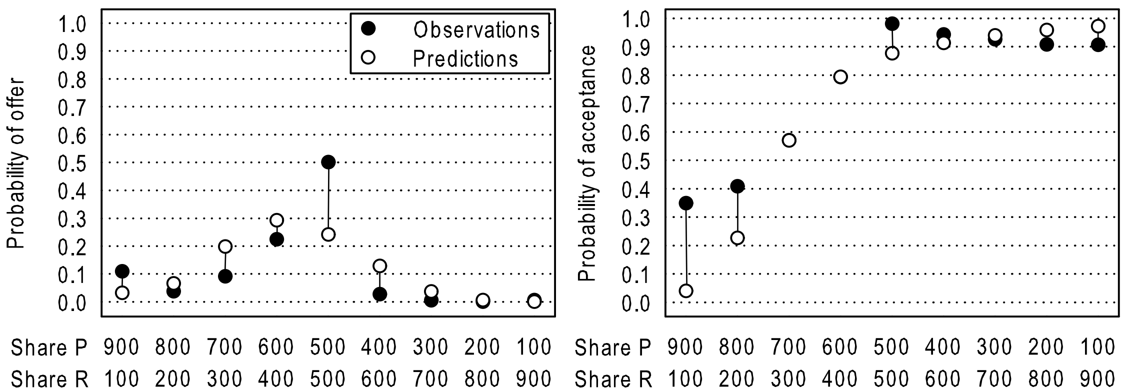
$$u_i(\sigma_i) = \begin{cases} c\left(\sigma_i - \frac{\alpha_i}{2}\left(\sigma_i - \frac{1}{N}\right)^2\right) & \text{if } \sigma_i < \frac{1}{N} \\ c\sigma_i & \text{otherwise.} \end{cases} \tag{4.4}$$

The utility function is based on the functional forms proposed by Bolton (1991) and Bolton and Ockenfels (2000) and it depends on the size of the cake c, the received relative share $\sigma_i \in [0, 1]$, and α_i which measures the utility loss from any disadvantageous deviation from the equal split.

Predictions We fit the ERC QRE model to the ultimatum game data provided by Güth et al. (2003) with maximum likelihood estimation. The log-likelihood of the model is -2231 and its parameter estimates (and standard deviations) are $\alpha_P = \alpha_R = 11.4214$ (0.9355), $\lambda_P = 0.0051$ (0.0003) and $\lambda_R = 0.0039$ (0.0002). The predictions for the ultimatum game with $N = 2$ or $N = 3$ can be computed by inserting the parameter estimates into Eq. 4.4, Eq. 4.2 and Eq. 4.3.

Observations vs Predictions The observed and predicted offer or acceptance probabilities for each proposal of the ultimatum game are depicted in Figure 4.9.

Figure 4.9
Probability for Each Proposal to be Offered or Accepted in the Ultimatum Game, Observations vs Predictions of the ERC QRE Model



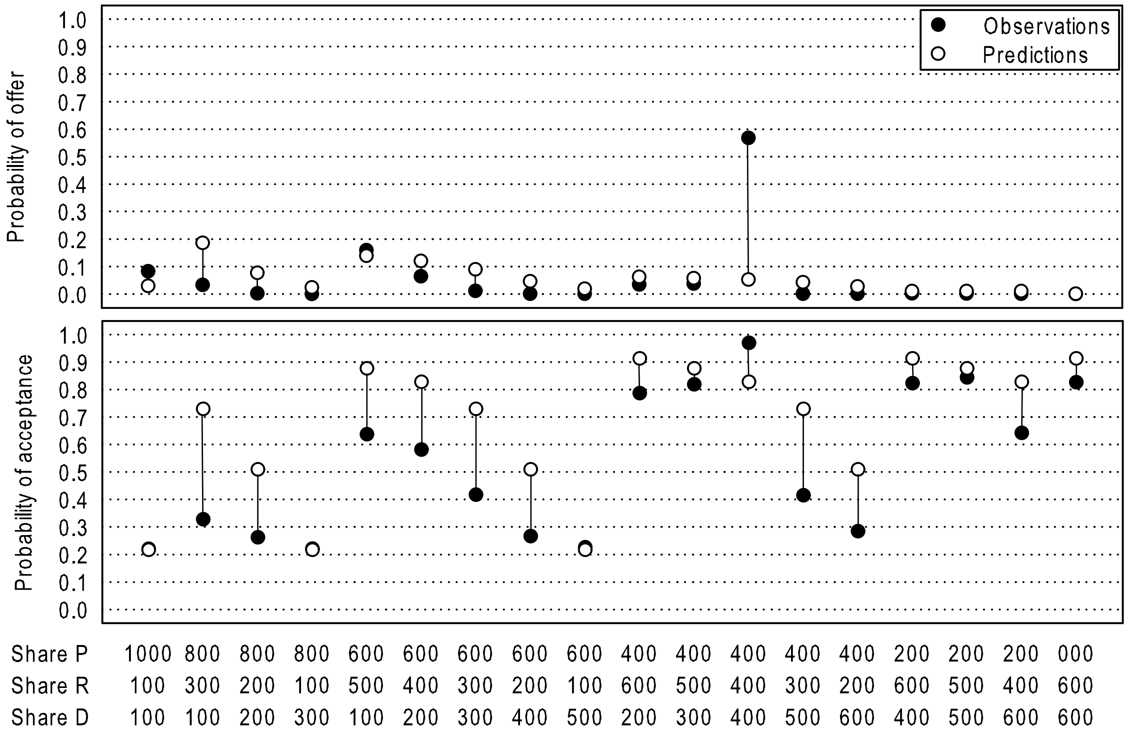
Source. Güth et al. (2003).

The model predicts an average offer of 42% of the monetary cake and an average acceptance rate of 73%. The corresponding observations are 41% and 81%. The sum of absolute differences between actual and predicted probabilities are (i) for proposals to be offered $SAD_P = 0.68$, (ii) for proposals to be accepted $SAD_R = 0.76$ and (iii) for outcomes to be observed $SAD_{P \times R} =$

0.73. The correlations between actual and predicted probabilities are (i) for proposals to be offered $R_P = 0.72$, (ii) for proposals to be accepted $R_R = 0.97$ and (iii) for outcomes to be observed $R_{P\times R} = 0.77$.

The observed and predicted offer or acceptance probabilities for each proposal of the three-person ultimatum game are depicted in Figure 4.10.

Figure 4.10
Probability for Each Proposal to be Offered or Accepted in the Three-Person Ultimatum Game, Observations vs Predictions of the ERC QRE Model



Source. Güth et al. (2007).

The model predicts an average offer of 29% of the monetary cake and an average acceptance rate of 72%. The corresponding observations are 33% and 79%. The sum of absolute differences between actual and predicted probabilities are (i) for proposals to be offered $SAD_P = 1.18$, (ii) for proposals to be accepted $SAD_R = 2.97$ and (iii) for outcomes to be observed $SAD_{P\times R} = 1.24$. The correlations between actual and predicted probabilities are (i) for proposals to be offered $R_P = 0.16$, (ii) for proposals to be accepted $R_R = 0.85$ and (iii) for outcomes to be observed $R_{P\times R} = 0.20$.

4.4 Summary

The behavioral assumptions that are made in each of the target models (the heuristic mix or one of the inequality aversion models) introduce additional free parameters with the aim of increasing the fit to the observed behavior and outcomes in ultimatum bargaining experiments. The result is that each target model fits each dependent variable (the offer behavior of the proposers, the acceptance behavior of the responders and the outcomes of the ultimatum game) better than the traditional SPE model. In the Fehr-Schmidt SPE model 30% of the population care only about their own income. The other 70% of the population consist of three additional types of inequality averse individuals with two additional inequality aversion parameters that differ over types. In both QRE models the individuals do not care only about their own income but additionally also about inequality. Moreover, they make mistakes in the sense of not choosing always the option that maximizes their expected utility and the likelihood of making a mistake differs depending on the role that they play in the ultimatum game. Lastly, six strategies of the heuristic mix model are explicitly designed to capture behavioral tendencies that deviate from the subgame perfect equilibrium strategy.

Within-Sample Fit Table 4.7 summarizes the relative improvement in within-sample fits achieved by the target models in comparison to the SPE model.

Table 4.7
Relative Fit Index (RFI) of Each Target Model for Each Dependent Variable of the Ultimatum Game

Target Model	Behavior of Proposers	Behavior of Responders	Outcomes
Heuristics Mix	0.81	0.78	0.82
Fehr-Schmidt SPE	0.69	0.72	0.69
Fehr-Schmidt QRE	0.84	0.72	0.82
ERC QRE	0.62	0.66	0.62

The RFI of a target model and a dependent variable is given by (SAD of SPE Model − SAD of Target Model)/SAD of SPE Model with the sum of absolute differences between predictions and observations (SAD) as criterion of fit. The RFI can take values between 1 and $-\infty$. A RFI of 1 means that the SAD of the target models equals 0 or that the SAD of the target model is 100% lower than the SAD of the SPE model. A RFI of 0 means that the SAD of the target model equals the SAD of the SPE model.

While each target model has a higher within-sample fit than the SPE, the relative improvements differ across models and dependent variable. Overall, the heuristic mix and the Fehr-Schmidt QRE fit the offer behavior, acceptance behavior and the outcomes of the ultimatum game best. While the heuristic mix performs best and the Fehr-Schmidt QRE model second-best in fitting the acceptance behavior of the responders, they change ranks in fitting the offer behavior of the proposers. The outcomes of the ultimatum game that depend on both offers and acceptances are fitted by both models equally well.[3]

Result 1 *The heuristic mix model is no worse in fitting the outcomes of the two-person ultimatum experiment than any of the inequality aversion models.*

Out-of-Sample Fit The relative out-of-sample fits depicted in Table 4.8 are lower than the within-sample fits but the predictions of all target models are still better than the ones of the SPE model.

Table 4.8
Relative Fit Index (RFI) of Each Target Model for Each Dependent Variable of the Three-Person Ultimatum Game

Target Model	Behavior of Proposers	Behavior of Responders	Outcomes
Heuristic Mix	0.79	0.57	0.70
Fehr-Schmidt SPE	0.08	0.49	0.08
Fehr-Schmidt QRE	0.43	0.69	0.46
ERC QRE	0.36	0.65	0.37

The heuristic mix predicts the behavior of the proposers best, followed by Fehr-Schmidt QRE, ERC QRE and Fehr-Schmidt SPE. The behavior is predicted best by the QRE models, followed by the heuristics mix and the Fehr-Schmidt SPE. The outcomes of the three-person ultimatum that depend on both proposer and responder behavior are predicted best by heuristic mix followed by Fehr-Schmidt QRE, ERC QRE and Fehr-Schmidt SPE.[4]

Result 2 *The heuristic mix model better in predicting the outcomes of the three-person ultimatum experiment than any of the inequality aversion models.*

The good performance of the heuristic mix in outcome prediction is due to its more accurate proposer predictions. It predicts that the three most frequently chosen proposals are chosen with positive probability and ignores the remaining 15 proposals. While the proposer prediction of the Fehr-Schmidt SPE model is too narrow (everyone chooses the $(600, 500)$ split) the proposer predictions of the QRE models are too vague.

Our study focuses on prediction rather than fitting. Furthermore, we study beside the equilibrium models widely used in behavioral economics a heuristic mix type of model. The main result of the study is that a heuristics mix model can predict better than equilibrium models. Thus, we take a first step to bridge the conceptual void between the individual and the aggregate level. For future research it would be useful to investigate the applicability of our heuristics mix model to other one-shot games and the incorporation of trial-and-error learning for games that are played repeatedly.

[3] The ultimatum game has 18 outcomes since each of the nine proposal can either be accepted or rejected.
[4] The three-person ultimatum game 36 outcomes since each of the eighteen proposals can either be accepted or rejected.

5 Choice Behavior in Extensive Form Games

5.1 Introduction

In the last decades experimental studies of one-shot extensive form games of complete information with two players (e.g., on the dictator game (Forsythe et al., 1994), the ultimatum game (Güth et al., 1982), the trust game (Berg et al., 1995) and gift exchange game (Fehr et al., 1998)) revealed that people apparently behave less selfishly and less sophisticated than one would expect based on the predictions of subgame perfect equilibrium (Selten, 1965, 1973) in which expected utility maximizers care only about their own payoff and may eliminate unreasonable Nash equilibria (Nash, 1951) by applying backward induction.

Several distinct explanations for the observed deviations from subgame perfect equilibrium were offered. The most important ones are altruism (Andreoni, 1990), envy (Bolton, 1991), inequity or inequality aversion (Loewenstein et al., 1989; Fehr and Schmidt, 1999; Bolton and Ockenfels, 2000), reciprocity (Rabin, 1993; Falk and Fischbacher, 2006), joint payoff maximization (Messick and McClintock, 1968), competitiveness (Messick and McClintock, 1968), level-k reasoning (Stahl and Wilson, 1995; Nagel, 1995) and quantal responses (McKelvey and Palfrey, 1998).

Recent research focused on the inclusion of one or more of these behavioral tendencies into quantitative behavioral models. It was demonstrated for a narrow set of games that these models fit and predict the behavior of participants better than subgame perfect equilibrium (e.g., Fehr and Schmidt, 1999; Bolton and Ockenfels, 2000; Charness and Rabin, 2002; De Bruyn and Bolton, 2008). However, there is still little research on two questions: (1) Which quantitative model is the best abstraction for predicting behavior in a broad set of extensive form games? (2) What is the relative importance of each behavioral tendency for predicting behavior in a broad set of extensive form games?

Ert et al. (2011) addressed both questions by organizing a competition on predicting behavior in a broad set of simple extensive form games of complete information. The validation procedure of the competition splits the data into an estimation set of games and a prediction set of games. Each set contains 120 extensive form games with two individuals: a first mover and a second mover. The game space involves ten classes of games with different properties. Each model is formulated and trained on the estimation set and the predictive power of each model is validated on the prediction set.

The competition was divided into two sub-competitions: one is on predicting the behavior of the first mover and the other one is on predicting the behavior of the second mover. Ert et al.

(2011) used the estimation set for fitting five baseline models for each sub-competition: the classic subgame perfect equilibrium (Selten, 1965, 1973), three stochastic variants of popular social preference models that they called inequality aversion (Fehr and Schmidt, 1999), equity reciprocity competition model (Bolton and Ockenfels, 2000; De Bruyn and Bolton, 2008), Charness and Rabin model (Charness and Rabin, 2002) and a new strategy mix model that consists of seven strategies (Ert et al., 2011). The surprising result was that the seven strategies model outperformed the popular social preference models.

Researchers could participated in one or both sub-competitions. They were allowed to use the baseline models together with the data of the estimation experiment for the development of their own models. After all models were submitted, the data of a prediction experiment were published. Based on this prediction set of games, all submitted models were ranked according to their mean predictive error and the ranking was published on the competition homepage.[1]

We submitted two models to the first mover competition and two models to the second mover competition. The first mover models were based on the discretized truncated subjective quantal response equilibrium model considered by Rogers et al. (2009). In both models we assume a slightly different heterogeneity in skill (i.e. preference responsiveness) and introduce heterogeneity in preferences (selfish versus other-regarding) and heterogeneity in preference beliefs (self-centered versus pessimistic). The heterogeneity in preferences is similar to the one assumed by Fehr and Schmidt (1999) and the heterogeneity in preference beliefs is inspired by the self-centered beliefs assumed in the seven strategies model and a very pessimistic strategy therein.

One second mover model is a stochastic social preference model. Compared to the social preference models that were used as baseline models, we assume a specification of social preferences that does not take the payoffs of a game directly into account. Instead the utility of a payoff distribution depends on whether it has certain characteristics relative to the other payoff distribution. The other second mover model is a strategy mix model that is based on a fast and frugal heuristic in the adaptive toolbox of bounded rationality (Gigerenzer and Selten, 2001): it is called take-the-best (Gigerenzer and Goldstein, 1996).

In our article we investigate four research questions: (1) Is it possible to achieve better fitting and prediction results by specifying and estimating different equilibrium models that were used by Ert et al. (2011)? Our results show that the gap between the seven strategies model and the equilibrium models in fitting the data of the estimation experiment is smaller than expected; however the seven strategies model still predicts the data of the prediction experiment better. (2) How good are the predictions of our submitted models in comparison to the baseline models? Our results show that our second mover models predict the choice behavior in the prediction set of games better than each baseline model and that our first mover models are only outperformed by the seven strategies model. (3) How reliable are the predictions results of the competition? We check how reliable the prediction results of the competition are by comparing them to predictions results of two different cross validations. Our results show that the ranking of the models may change in the cross validations if the prediction results

[1] See `https://sites.google.com/site/extformpredcomp/competition-results-and`.

in the competition are close and that only groups of models with similar results that differ considerably between groups do not change ranks. (4) How can we achieve better predictions by combining predictions of different models? Our results show that simple averaging of predictions of good models yields better predictions than each individual model and that optimal predictions are only obtained if predictions of semi-good models that are not highly correlated to the predictions of the good models are included.

Section 5.2 explains in greater detail structure and content of the prediction competition. In Section 5.3 we consider six baseline models that are based on the recent literature on predicting behavior in games.

Table 5.1
Baseline Models

Section	Mnemonic	Name of Model
5.3.1	SPE	Subgame Perfect Equilibrium
5.3.2	QRE	Quantal Response Equilibrium
5.3.3	FS-QRE	Fehr-Schmidt Quantal Response Equilibrium
5.3.4	BO-QRE	Bolton-Ockenfels Quantal Response Equilibrium
5.3.5	CR-QRE	Charness-Rabin Quantal Response Equilibrium
5.3.6	7S	Seven Strategies

Each baseline model in Table 5.1 contains a second mover model and a first mover model. We fit the free parameters of the 2×6 models to the data of the estimation experiment, predict the data of the prediction experiment and report within sample and out-of-sample fits. In Section 5.4 we describe the four models that we submitted to the prediction competition.

Table 5.2
Own Models

Section	Mnemonic	Name of Model
5.4.1	SUM	Stochastic Utility Maximizer
5.4.2	TTB	Take-The-Best
5.4.3	SQRE	Subjective Quantal Response Equilibrium
5.4.4	SVO-SQRE	SQRE with Social Value Orientation

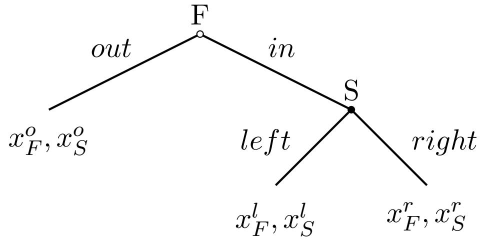
The first two models of Table 5.2 were submitted to the second mover competition and the other two models to the first mover competition. We explain how we fitted the free parameters of each model and report within sample and out-of-sample fits. In Section 5.5 we rank and compare the predictive power of our models to the predictive power of the baseline models, check if the ranking is reliable and show that averaging good predictions yields better but not optimal predictions. In Section 5.6 we summarize the main results of our study and answer the two research questions that were raised by the competition organizers.

5.2 Prediction Competition

Validation Procedure The organizers of the prediction competition Ert et al. (2011) provided experimental data on an estimation set of 120 games (that are listed in Table 6) and five baseline models that were implemented in computer programs and fitted to the data of the estimation experiment.[2] The baseline models served as a benchmark and could be used together with the data of the estimation experiment for the development of own models. After all models were submitted, the data of a prediction experiment were published. The prediction experiment contained another 120 different games that are listed in Table 7. Based on this prediction set of games, all submitted models were ranked according to their mean predictive error (validation criterion) and the ranking was published on the competition homepage.[3]

Structure of the Games Each of the 2×120 games involves two individuals that act sequentially (see Figure 5.1). The first mover F can choose either alternative *out* or alternative *in*. Alternative *out* implies the implementation of payoff distribution (x_F^o, x_S^o). Alternative *in* implies that the second mover S can choose between alternative $left$ that yields payoff distribution (x_F^l, x_S^l) or alternative $right$ that yields payoff distribution (x_F^r, x_S^r).

Figure 5.1
Structure of the Basic Extensive Form Game



Source. Ert et al. (2011)

Depending on the values of the six payoffs $x_I^a \in \{-8, -7, ..., 8\}$ with $I \in \{F, S\}$ and $a \in \{o, l, r\}$, a particular game involves selfish, altruistic, or neutral actions by the first mover and allows for reciprocation, punishment, or reward of the first mover's choice by the second mover (see Table 5.3).

Game Space Classification The 2×120 games were sampled by a quasi random sampling algorithm (see Appendix in Ert et al., 2011). Table 5.3 classifies the game space into ten

[2] The computer programs can be downloaded from the prediction competition homepage: `https://sites.google.com/site/extformpredcomp/baseline-models`.
[3] See `https://sites.google.com/site/extformpredcomp/competition-results-and`.

classes of games and denotes the proportions for each class that are implied by the algorithm.

Table 5.3
Classification of Games

Class	Properties	Share[a]
Safe Shot	Choosing *in* yields highest payoff for F	38.0%
Near Dictator	Highest payoff of F does not depend on the choice of S	32.0%
Common Interest	One alternative yields the highest payoff for F and S	19.0%
Costly Help	Improving the other payoff is costly for the helper	7.4%
Trust Game	Choosing *in* improves payoff of S but not if S reciprocates	6.5%
Rational Punish	Punishing *in* choice of F yields highest payoff for S	5.3%
Costly Punish	Punishing *in* choice of F is costly for S	4.0%
Strategic Dummy	S cannot affect payoffs	3.5%
Free Help	Improving the other payoff is not costly	2.8%
Free Punish	S can punish *in* choice of F with no cost	1.2%

Source. Ert et al. (2011)
[a] The sum of shares is greater than 100% since some games fall into more than one class.

Sub-Competitions The competition was divided into two independent but related sub-competitions. One sub-competition was on predicting the behavior of the first mover and the other sub-competition was on predicting the behavior of the second mover. The researches could submit different models to each sub-competition or one model that predicts the behavior in both sub-competitions. Both sub-competitions were based on the same estimation set and the same prediction set of games. Table 6 contains the experimental data of the estimation set of games on which the baseline models and the submitted models were fitted and Table 7 contains the experimental data of the prediction set of games on which the submitted models were ranked based on their mean predictive error. For each game $\Gamma \in \{1, 2, ..., 120\}$ of the prediction set, a first mover model had to predict the observed *in* choice probability $p^i_\Gamma \in [0, 1]$ while a second mover model had to predict the observed *right* choice probability $p^r_\Gamma \in [0, 1]$.[4]

Validation Criterion The validation criterion of the competition was the mean of the squared deviations (MSD) of the observed choice probabilities of the games in the predictions set and the corresponding choice probabilities that are predicted by a model. The mean predictive error of a first mover model is given by

$$MSD^F_{pre} = \sum_{\Gamma=1}^{120} \frac{1}{120} \cdot (p^i_\Gamma - \hat{p}^i_\Gamma)^2,$$

[4] Each choice probability is based on 16 individual observations. The raw data and the instructions can be downloaded from the prediction competition homepage (see section raw data and section method): `https://sites.google.com/site/extformpredcomp/`.

with $\hat{p}^i_\Gamma \in [0,1]$ denoting the *in* choice probability that is predicted by a first mover model for game Γ.

The mean predictive error of a second mover model is given by

$$MSD^S_{pre} = \sum_{\Gamma=1}^{120} \frac{1}{120} \cdot (p^r_\Gamma - \hat{p}^r_\Gamma)^2,$$

with $\hat{p}^r_\Gamma \in [0,1]$ denoting the *right* choice probability that is predicted by a second mover model for game Γ.

5.3 Baseline Models

Table 5.4 summarizes for each baseline model that we consider in this Section the fitting result (within-sample fit) and the prediction result (out-of-sample fit).

Table 5.4
Fitting versus Predicting: Performance of Baseline Models

Section	Model	MSD^F_{est}	MSD^F_{pre}	MSD^S_{est}	MSD^S_{pre}
5.3.1	SPE	0.0545	0.0532	0.0105	0.0071
5.3.2	QRE	0.0170	0.0141	0.0092	0.0057
5.3.3	FS-QRE	0.0118	0.0140	0.0082	0.0056
5.3.4	BO-QRE	0.0141	0.0172	0.0073	0.0056
5.3.5	CR-QRE	0.0112	0.0143	0.0042	0.0067
5.3.6	7S	0.0119	0.0083	0.0029	0.0043

The QRE models[5] and the seven strategies model relax one or more assumptions of the traditional SPE model (Selten, 1965, 1973). The first QRE model (McKelvey and Palfrey, 1998) introduces choice errors and permits therefore deviations from optimal play but keeps the assumption that individuals have monetary preferences.[6] The other three QRE models introduce additionally different specifications of social preferences (Fehr and Schmidt, 1999; Bolton and Ockenfels, 2000; Charness and Rabin, 2002) that permit deviations from selfish

[5] Strictly speaking, each QRE model that we consider is a so called logit agent quantal response equilibrium (AQRE). Logit AQRE is a parametric specification of AQRE in which the choice probabilities of the individuals are defined by logit response functions. AQRE applies QRE to the agent normal form of extensive form games and QRE is a statistical generalization of Nash equilibrium (Nash, 1951) in which individuals stochastically best respond (McKelvey and Palfrey, 1995). Since logit AQRE is a subclass of QRE, we use the shorter terminus QRE.

[6] SPE is a refinement of Nash equilibrium (Nash, 1951) that assumes additionally that individuals use backward induction. Notice that QRE models do not assume backward induction since assuming that individuals stochastically best respond implies that a unique equilibrium is selected (McKelvey and Palfrey, 1995, 1998).

behavior. Lastly, one of the strategies in the seven strategies model (Ert et al., 2011) is a prescription of the SPE model.

Some of the baseline models that are listed in Table 5.4 differ from the ones that the organizers of the competition used as baseline models and implemented in computer programs. In particular, we reformulated the social preference QRE models and achieved thereby better within-sample fits than the ones reported by Ert et al. (2011) (see Appendix 13 and Table 5 for more information). In our study, the seven strategies model that performed best in fitting the behavior of both individuals in the study of Ert et al. (2011) has still the highest within-sample fit of all second mover models but no more the highest within-sample fit of all first mover models. However, the seven strategies model still outperforms all other models in predicting the behavior of the second mover and the behavior of the first mover.[7]

For seperating the effect of choice errors from the effect of social preferences in fitting and predicting the choice behavior of the first mover and the second mover, we estimated additionally a QRE model with monetary preferences. A comparison of the QRE models listed in Table 5.4 reveals that in any case, an introduction of social preferences increases the within-sample fit but *not* the out-of-sample fit. In the case of the Bolton-Ockenfels QRE (first mover) and in the case of Charness-Rabin QRE (second mover) social preferences even decrease the out-of-sample fit.

The remainder of this Section introduces and describes each baseline model that contains a second mover model and a first mover model. Since the latter may include the former, we start first with the description of the second mover model and present the parameter estimates that minimisze the second mover MSD of the estimation set of games

$$MSD^S_{est} = \sum_{\Gamma=1}^{120} \frac{1}{120} \cdot (p^r_\Gamma - \hat{p}^r_\Gamma)^2.$$

Then, we describe the first mover model that may include the second mover model (but not its parameter estimates[8]) and list then the parameter estimates that minimize the first mover MSD of the estimation set of games

$$MSD^F_{est} = \sum_{\Gamma=1}^{120} \frac{1}{120} \cdot (p^i_\Gamma - \hat{p}^i_\Gamma)^2.^9$$

[7] Ert et al. (2011) do not report prediction MSD scores for their baseline models. However, since none of the baseline models except the seven strategies was within the best 15 predictors in both competitions, we can infer that the seven strategies model had the best out-of-sample fit.

[8] Notice that the parameter estimates that determine the behavior of the second mover in a first mover model may be different from the parameter estimates that determine the behavior of the second mover in the second mover model.

[9] Notice that we do not use from now on the game index Γ to denote a choice probability in a game for saving some notation.

5.3.1 Subgame Perfect Equilibrium

Subgame perfect equilibrium (SPE) is a refinement of Nash equilibrium (Nash, 1951) that was introduced by Selten (1965, 1973) in order to eliminate unreasonable Nash equilibria in extensive form games of complete information.[10] In the subgame perfect equilibrium it is common knowledge that the first mover and the second mover maximize their own expected utility, that their preferences are defined by a utility function that depends only on their own payoff and that the first mover applies backward induction and may therefore eliminate unreasonable Nash equilibria.

SPE Second Mover Model Second mover S is a utility maximizer who cares only about her payoff. Her utility from choosing alternative $a \in \{l, r\}$ is equal to

$$u_S^a = x_S^a \tag{5.1}$$

and she chooses *right* with probability

$$\hat{p}^r = \begin{cases} 0 & \text{if } u_S^l > u_S^r \\ 1 & \text{if } u_S^l < u_S^r \\ 0.5 & \text{if } u_S^l = u_S^r. \end{cases} \tag{5.2}$$

The mean of squared deviations for the estimation set and for the prediction set are

$$MSD_{est}^S = 0.0105 \quad MSD_{pre}^S = 0.0071.$$

SPE First Mover Model First mover F is an expected utility maximizer who cares only about her payoff and who has consistent beliefs about the second mover. His utility from alternative $a \in \{o, l, r\}$ is equal to

$$u_F^a = x_F^a. \tag{5.3}$$

F knows the utility function of second mover S (Equation 5.1) and thus her actual choice probability $\hat{p}^r$ (Equation 5.2). This implies that his expected utility Eu_F^i from choosing *in* is equal to $(1 - \hat{p}^r) \cdot u_F^l + \hat{p}^r \cdot u_F^r$. F chooses *in* with probability

$$\hat{p}^i = \begin{cases} 0 & \text{if } u_F^o > Eu_F^i \\ 1 & \text{if } u_F^o < Eu_F^i \\ 0.5 & \text{if } u_F^o = Eu_F^i. \end{cases}$$

[10] Unreasonable Nash equilibria depend on non-credible threats (see Section 4.3.1).

The mean of squared deviations for the estimation set and for the prediction set are

$$MSD^F_{est} = 0.0545 \quad MSD^F_{pre} = 0.0532.$$[11]

5.3.2 Quantal Response Equilibrium

Quantal response equilibrium (QRE) is a solution concept that was introduced for normal form games by McKelvey and Palfrey (1995) and applied to extensive form games by McKelvey and Palfrey (1998). It is a combination of Nash equilibrium (Nash, 1951) and the stochastic utility model of Luce (1959). In comparison to subgame perfect equilibrium in which individuals always maximize their expected utility, in a quantal response equilibrium they do it stochastically which leads to deviations from optimal play that are interpreted as choice errors. How large the deviations from optimal play are depends on how responsive the individuals are to their preferences.

QRE Second Mover Model Second mover S is a stochastic utility maximizer who cares only about her payoff. Her utility function is defined by Equation 5.1. Given her preference responsiveness $\lambda_S \in [0, \infty]$, she chooses alternative $right$ with probability

$$\hat{p}^r = \frac{e^{\lambda_S \cdot u^r_S}}{e^{\lambda_S \cdot u^l_S} + e^{\lambda_S \cdot u^r_S}}. \tag{5.4}$$

Choice probability $\hat{p}^r \in [0, 1]$ depends on two factors: First, on the second mover's preference responsiveness λ_S; and second, on the differences between the utility scores u^l_S and u^r_S. For a given preference responsiveness, the likelihood that S acts in accordance with her preferences is higher, the greater the differences between the utility scores are; and for given utility scores, the likelihood that S acts in accordance with her preferences is higher, the greater the her preference responsiveness is. This implies two extreme cases in which S either decides randomly if $\lambda_S = 0$ or chooses always the alternative that maximizes her utility if $\lambda_S = \infty$. Notice that in the latter case the predictions of the QRE second mover model are equal to the predictions of the SPE second mover model (see Section 5.3.1).

The parameter estimate that minimizes the mean of squared deviations of the estimation set is

$$\lambda_S = 2.073$$

and the mean of squared deviations for the estimation set and for the prediction set of the estimated model are

$$MSD^S_{est} = 0.0092 \quad MSD^S_{pre} = 0.0057.$$

[11] The MATLAB code that was used to compute the mean squared deviations of both models is attached with detailed comments in Appendix 16.1.

QRE First Mover Model First mover F is a stochastic expected utility maximizer who cares only about her payoff and who has consistent beliefs about the second mover. His utility function is defined by Equation 5.3. F knows the utility function of second mover S (Equation 5.1), her preference responsiveness λ_S, and thus the actual choice probability $\hat{p}^r$ (Equation 5.4). This implies that $Eu_F^i = (1 - \hat{p}^r) \cdot u_F^l + \hat{p}^r \cdot u_F^r$. Given his preference responsiveness $\lambda_F \in [0, \infty]$, he chooses *in* with probability

$$\hat{p}^i = \frac{e^{\lambda_F \cdot Eu_F^i}}{e^{\lambda_F \cdot u_F^o} + e^{\lambda_F \cdot Eu_F^i}}. \tag{5.5}$$

The *in* choice probability $\hat{p}^i \in [0, 1]$ depends on λ_F, u_F^o, Eu_F^i and the relationships are analogous to the ones explained for Equation 5.4. Notice however that it is in particular true that the predictions of the SPE first mover model (see Section 5.3.1) are equal to the predictions of the QRE first mover model if $\lambda_F = \infty$ and $\lambda_S = \infty$.

The parameter estimates that minimize the mean of squared deviations of the estimation set are

$$\lambda_S = 0.6950 \quad \lambda_F = 0.6823$$

and the mean of squared deviations for the estimation set and for the prediction set of the estimated model are

$$MSD_{est}^F = 0.0170 \quad MSD_{pre}^F = 0.0141.$$[12]

5.3.3 Quantal Response Equilibrium with Fehr-Schmidt Preferences

In the Fehr-Schmidt QRE model we combine the quantal response equilibrium (that we consider in Section 5.3.2) with the social preference specification of Fehr and Schmidt (1999). The latter permits deviations from selfish behavior by assuming that individuals do not care only about their own payoff but also dislike unequal income distributions and the former permits deviations from optimal play.

FS-QRE Second Mover Model Second mover S is a stochastic utility maximizer who cares about her payoff and is inequality averse. Given her degree of aversion to disadvantageous inequality α_S and her degree of aversion to advantageous inequality β_S with $\beta_S \in [0, 1)$ and $\alpha_S \geq \beta_S$, her utility from choosing alternative $a \in \{l, r\}$ is equal to

$$u_S^a = x_S^a - \alpha_S \cdot max(0, x_F^a - x_S^a) - \beta_S \cdot max(0, x_S^a - x_F^a). \tag{5.6}$$

The first term of her utility function captures the utility from her payoff, the second term represents the utility loss from disadvantageous payoff inequality and the third term is the utility loss from advantageous payoff inequality. The degree of aversion against advantageous inequality is given by $\beta_S \in [0, 1)$. The non-negativity restriction implies that the second mover

[12] The MATLAB code that was used to estimate the free parameters of both models and to compute the mean squared deviations is attached with detailed comments in Appendix 16.4.

cannot gain utility from advantageous inequality and the restriction to values below one implies that she does not burn money to reduce advantageous inequality. The degree of aversion to disadvantageous inequality is measured by $\alpha_S \geq \beta_S$. The parameter restriction means that the second mover cannot suffer more from an advantageous inequality in comparison to an equivalent disadvantageous inequality. Given her preference responsiveness $\lambda_S \in [0, \infty]$, S chooses alternative *right* with probability $\hat{p}^r$ (Equation 5.4).

The parameter estimates that minimize the mean of squared deviations of the estimation set are

$$\lambda_S = 2.299 \quad \alpha_S = 0.0353 \quad \beta_S = \alpha_S$$

and the mean of squared deviations for the estimation set and for the prediction set of the estimated model are

$$MSD^S_{est} = 0.0082 \quad MSD^S_{pre} = 0.0056.$$ [13]

FS-QRE First Mover Model First mover F is a stochastic expected utility maximizer who has consistent beliefs about the second mover S and who is inequality averse and cares about his payoff. Given his degree of aversion to disadvantageous inequality α_F and his degree of aversion to advantageous inequality β_F with $\beta_F \in [0, 1)$ and $\alpha_F \geq \beta_F$, his utility from alternative $a \in \{o, l, r\}$ is equal to

$$u^a_F = x^a_F - \alpha_F \cdot max(0, x^a_S - x^a_F) - \beta_F \cdot max(0, x^a_F - x^a_S). \tag{5.7}$$

F knows the utility function of S (Equation 5.1) including α_S and β_S, her preference responsiveness λ_S, and thus the actual choice probability $\hat{p}^r$ (Equation 5.4). This implies that $Eu^i_F = (1 - \hat{p}^r) \cdot u^l_F + \hat{p}^r \cdot u^r_F$. Given his preference responsiveness $\lambda_F \in [0, \infty]$, F chooses *in* with probability $\hat{p}^i$ (Equation 5.5).

The parameter estimates that minimize the mean of squared deviations of the estimation set are

$$\lambda_S = 0.8295 \quad \alpha_S = 0.1494 \quad \beta_S = 0.1073 \quad \lambda_F = 0.8142 \quad \alpha_F = 0.0483 \quad \beta_F = \alpha_F$$

and the mean of squared deviations for the estimation set and for the prediction set of the estimated model are

$$MSD^F_{est} = 0.0118 \quad MSD^F_{pre} = 0.0140.$$ [14]

[13] The second mover model was estimated by restricting the values of β_S to the values of α_S. The reason for this restriction is that the estimates of an unconstrained estimation of the model ($\lambda_S = 3.317$, $\alpha_S = 0$, $\beta_S = 0.0787$) imply a higher value for β_S than for α_S which violates parameter restriction $\alpha_S \geq \beta_S$. The unconstrained estimation of the model yields a lower MSD of 0.0046 for the estimation set (in comparison to 0.0082) but a higher MSD of 0.0062 for the prediction set (in comparison to 0.0056). This indicates that the parameter restriction $\alpha_S \geq \beta_S$ made by Fehr and Schmidt (1999) prevented overfitting the data of the estimation set.

[14] The first mover model was estimated by restricting the values of β_F to the values of α_F. The reason for this restriction is that the estimates of an unconstrained estimation of the model ($\lambda_S = 0.8726$, $\alpha_S = 0.1344$, $\beta_S = 0.0824$, $\lambda_F = 0.8567$, $\alpha_F = 0.0145$, $\beta_F = 0.0808$) imply a higher value for β_F than for α_F which violates parameter restriction $\alpha_F \geq \beta_F$. Notice however that the performance of the unconstrained model ($MSD_{est} = 0.0112$, $MSD_{pre} = 0.0143$) in fitting and predicting is similar to the one of the constrained

5.3.4 Quantal Response Equilibrium with Bolton-Ockenfels Preferences

In the Bolton-Ockenfels QRE model we combine the quantal response equilibrium (that we consider in Section 5.3.2) with a specification of social preferences that was defined by Bolton and Ockenfels (2000). The latter permits deviations from selfish behavior by assuming that individuals do not care only about their own payoff but also dislike to get less or more than the equal split. The former permits deviations from optimal play.

BO-QRE Second Mover Model Second mover S is a stochastic utility maximizer who cares about her payoff and how far her payoff is away from the equal split. Given her degree of aversion to deviations from the equal split $b_S \geq 0$, her utility from choosing alternative $a \in \{l, r\}$ equals

$$u_S^a = c\sigma_S - \frac{b_S}{2} c \left(\sigma_S - \frac{1}{2} \right)^2. \tag{5.8}$$

The first term captures the utility from her payoff with σ_S denoting the proportion of the monetary cake c that she receives. The second term represents her utility loss from a deviation from the equal split. The greater the deviation from the equal split is, the higher is her utility loss. Her degree of aversion $b_S \geq 0$ measures the importance of a deviation from the equal split. Since the payoffs in the games that we consider can be negative, the form of the utility function requires a positive transformation of payoffs to exclude negative payoffs. The transformed payoffs are given by $\tilde{x}_S^a = x_S^a + m$ and $\tilde{x}_F^a = x_F^a + m$ with $m = |min(x_F^o, x_F^l, x_F^r, x_S^o, x_S^l, x_S^r)|$. The monetary cake is given by $c = \tilde{x}_S^a + \tilde{x}_F^a + \epsilon$ with ϵ being a very small positive number that is added to the monetary cake to avoid division by 0. The proportion of the monetary cake that S receives is defined by $\sigma_S = \frac{\tilde{x}_S^a}{c}$.[15] Given her preference responsiveness $\lambda_S \in [0, \infty]$, S chooses alternative *right* with probability $\hat{p}^r$ (Equation 5.4). The parameter estimates that minimize the mean of squared deviations of the estimation set are

$$\lambda_S = 2.361 \quad b_S = 0.3779$$

and the mean of squared deviations for the estimation set and for the prediction set of the estimated model are

$$MSD_{est}^S = 0.0073 \quad MSD_{pre}^S = 0.0056.$$

BO-QRE First Mover Model First mover F is a stochastic expected utility maximizer who has consistent beliefs about the second mover and who cares about his payoff and how far his payoff is away from the equal split. Given his degree of aversion to deviations from the equal split $b_F \geq 0$ and the proportion of the monetary cake that he receives $\sigma_F = \frac{\tilde{x}_F^a}{c}$, his utility

model. The MATLAB code that was used to estimate the free parameters of both models and to compute the mean squared deviations is attached with detailed comments in Appendix 16.5. The code of the unconstrained estimation is attached in Appendix 16.7.

[15] Ert et al. (2011) conducted a similar payoff transformation with the only difference that $\epsilon = 1$ which is less accurate than assuming a very small positive ϵ that approaches 0.

from alternative $a \in \{o, l, r\}$ is defined by

$$u_F^a = c\sigma_F - \frac{b_F}{2} c \left(\sigma_F - \frac{1}{2}\right)^2. \tag{5.9}$$

F knows the utility function of the second mover S (Equation 5.8) including b_S, her preference responsiveness λ_S and and thus the actual choice probability $\hat{p}^r$ (Equation 5.4). This implies that $Eu_F^i = (1-\hat{p}^r) \cdot u_F^l + \hat{p}^r \cdot u_F^r$. Given his preference responsiveness $\lambda_F \in [0, \infty]$, he chooses *in* with probability $\hat{p}^i$ (Equation 5.5). The parameter estimates that minimize the mean of squared deviations of the estimation set are

$$\lambda_S = 0.7783 \quad b_S = 0.9214 \quad \lambda_F = 0.7730 \quad b_F = 0.5413$$

and the mean of squared deviations for the estimation set and for the prediction set of the estimated model are

$$MSD_{est}^F = 0.0141 \quad MSD_{pre}^F = 0.0172.^{16}.$$

5.3.5 Quantal Response Equilibrium with Charness-Rabin Preferences

In the Charness-Rabin QRE model we combine the quantal response equilibrium (that we consider in Section 5.3.2) with a specification of social preferences that was formulated by Charness and Rabin (2002). Compared to the other two specifications of social preferences (that we considered in Sections 5.3.3 and 5.3.4), the individuals may not only be concerned with reducing inequality, but also with increasing social welfare. Moreover the second mover may also be motivated by reciprocity.

CR-QRE Second Mover Model Second mover S is a stochastic utility maximizer who cares about her payoff and may also care about the payoff of first mover F and whether he misbehaved or not. Her utility from choosing alternative $a \in \{l, r\}$ is defined by

$$u_S^a = (1 - \rho_S r - \sigma_S s - \theta_S q) \cdot x_S^a + (\rho_S r + \sigma_S s + \theta_S q) \cdot x_F^a, \tag{5.10}$$

where $r = 1$ if she is better off ($x_S^a > x_F^a$) and $r = 0$ otherwise; $s = 1$ if she is worse off ($x_S^a < x_F^a$) and $s = 0$ otherwise; $q = -1$ if the first mover F misbehaved and $q = 0$ otherwise. A misbehavior is present if F chooses *in* although *out* yields a higher joint payoff and a higher payoff for S. The utility of S from alternative a is a weighted sum of her payoff and the payoff of the first mover. The weight S places on the payoff of F may depend on whether F has a higher or lower payoff and on whether F misbehaved or not. The reciprocity parameter θ_S takes a misbehavior of the first mover into account and parameters ρ_S and σ_S allow for different types of social preferences. Inequality aversion (Fehr and Schmidt, 1999; Bolton and Ockenfels, 2000) corresponds to $1 > \rho_S > 0 > \sigma_S$, i.e., S cares about her own

[16] The MATLAB code that was used to estimate the free parameters of both models and to compute the mean squared deviations is attached with detailed comments in Appendix 16.9

payoff and is inequality averse which implies that she wishes to lower the payoff of F when she is worse off ($s = 1$). Social welfare preferences (Andreoni and Miller, 2002) correspond to $1 > \rho_S > \sigma_S > 0$, i.e., S prefers a higher payoff for herself and a higher payoff for F but favors herself when she is worse off ($s = 1$). Given her preference responsiveness $\lambda_S \in [0, \infty]$, S chooses alternative *right* with probability $\hat{p}^r$ (Equation 5.4). The parameter estimates that minimize the MSD of the estimation set are

$$\lambda_S = 3.437 \quad \rho_S = 0.0758 \quad \sigma_S = 0.0232 \quad \theta_S = 0.$$

The estimates imply that the second mover has social welfare preferences and does not care about a misbehavior of the first mover. The mean of squared deviations for the estimation set and for the prediction set of the estimated model are

$$MSD^S_{est} = 0.0042 \quad MSD^S_{pre} = 0.0067.$$

CR-QRE First Mover Model The first mover is a stochastic expected utility maximizer who cares about his payoff, the payoff of the second mover and who has consistent beliefs about the second mover. The utility of first mover F from alternative $a \in \{o, l, r\}$ is defined by

$$u^a_F = x^a_F \cdot (1 - \rho_F \cdot s - \sigma_F \cdot r) + x^a_S \cdot (\rho_F \cdot s + \sigma_F \cdot r)\,.$$

F knows the utility function of second mover S (Equation 5.10) including ρ_S, τ_S and θ_S, her preference responsiveness λ_S, and therefore her choice probability $\hat{p}^r$ (Equation 5.4). This implies that $Eu^i_F = (1 - \hat{p}^r) \cdot u^l_F + \hat{p}^r \cdot u^r_F$. Given his preference responsiveness $\lambda_F \in [0, \infty]$, F chooses *in* with probability $\hat{p}^i$ (Equation 5.5). The parameter estimates that minimize the mean of squared deviations of the estimation set are

$$\lambda_S = 0.8746 \;\; \rho_S = 0.0819 \;\; \sigma_S = -0.134 \;\; \theta_S = 0.0016 \;\; \lambda_F = 0.8585 \;\; \rho_F = 0.0815 \;\; \sigma_F = -0.0143.$$

The estimates imply that both individuals are inequality averse and that the second mover cares (a little) about the misbehavior of the first mover. The mean of squared deviations for the estimation set and for the prediction set of the estimated model are

$$MSD^F_{est} = 0.0112 \quad MSD^F_{pre} = 0.0143.$$[17]

5.3.6 Seven Strategies

Each strategy of the seven strategies (7S) model of Ert et al. (2011) is an effort to maximize a certain target value and may reflect a belief of the first mover about the rule the second mover applies. The rules and beliefs that are implied by each strategy are summarized in Table 5.5.

[17] The MATLAB code that was used to estimate the free parameters of both models and to compute the mean squared deviations is attached with detailed comments in Appendix 16.11.

Table 5.5
Rules and Beliefs Implied by Each of the Seven Strategies

Strategy	Rule	Belief
$Ratio$	Maximize own payoff	Other individual uses the same rule
$JointMx$	Maximize joint payoff	Other individual uses the same rule
$MxWeak$	Maximize minimum payoff	Other individual uses the same rule
$MnDiff$	Minimize payoff difference	Other individual uses the same rule
$MaxMin$	Maximize own minimum payoff	-
$Level-1$	Maximize own payoff	Other individual acts randomly
$NiceR$	Maximize own payoff; if you are indifferent, maximize other payoff	-

7S Second Mover Model There is an infinite population of second movers and each second mover S uses one of the seven strategies considered above. Since strategies $Maxmin$ and $Level-1$ are perfectly correlated with strategy $Ratio$ (see Ert et al., 2011), the population of second movers can be characterized by a probability distribution β over five strategies.

Fraction β_R of the population uses strategy $Ratio$ and chooses alternative $right$ with probability

$$\hat{p}_R^r = \begin{cases} 0 & \text{if } x_S^l > x_S^r \\ 1 & \text{if } x_S^l < x_S^r \\ 0.5 & \text{otherwise.} \end{cases}$$

Fraction β_N uses strategy $NiceR$ and chooses alternative $right$ with probability

$$\hat{p}_N^r = \begin{cases} 0 & \text{if } x_S^l > x_S^r \vee (x_S^l = x_S^r \wedge x_F^l > x_F^r) \\ 1 & \text{if } x_S^l < x_S^r \vee (x_S^l = x_S^r \wedge x_F^l < x_F^r) \\ 0.5 & \text{otherwise.} \end{cases}$$

Fraction β_J uses strategy $JointMx$ and chooses alternative $right$ with probability

$$\hat{p}_J^r = \begin{cases} 0 & \text{if } x_S^l + x_F^l > x_S^r + x_F^r \\ 1 & \text{if } x_S^l + x_F^l < x_S^r + x_F^r \\ 0.5 & \text{otherwise.} \end{cases}$$

Fraction β_W uses strategy $MxWeak$ and chooses alternative $right$ with probability

$$\hat{p}_W^r = \begin{cases} 0 & \text{if } min(x_S^l, x_F^l) > min(x_S^r, x_F^r) \\ 1 & \text{if } min(x_S^l, x_F^l) < min(x_S^r, x_F^r) \\ 0.5 & \text{otherwise.} \end{cases}$$

Fraction $\beta_D = 1 - \beta_R + \beta_N + \beta_J + \beta_W$ uses strategy $MnDiff$ and chooses alternative *right* with probability

$$\hat{p}_D^r = \begin{cases} 0 & \text{if } |x_S^l - x_F^l| < |x_S^r - x_F^r| \\ 1 & \text{if } |x_S^l - x_F^l| > |x_S^r - x_F^r| \\ 0.5 & \text{otherwise.} \end{cases}$$

Thus, the probability to observe a *right* choice that is predicted by the second mover model is given by

$$\hat{p}^r = \beta_R \cdot \hat{p}_R^r + \beta_N \cdot \hat{p}_N^r + \beta_J \cdot \hat{p}_J^r + \beta_W \cdot \hat{p}_W^r + \beta_D \cdot \hat{p}_D^r.$$

The probability distribution over the six strategies β is estimated by means of a regression analysis with no intercept and the restriction that the sum of the weights is equal to 1. The parameter estimates are

$$\beta_R = 0.5038 \quad \beta_N = 0.3565 \quad \beta_J = 0.0581 \quad \beta_W = 0.0445 \quad \beta_D = 0.0371$$

and the mean of squared deviations for the estimation set and for the prediction set of the estimated model are

$$MSD_{est}^S = 0.0029 \quad MSD_{pre}^S = 0.0043.$$

7S First Mover Model There is an infinite population of first movers and each first mover F uses one of six strategies. Since strategy $NiceR$ cannot be applied to the strategic choice problem of F (see Ert et al., 2011), the population of first movers can be characterized by a probability distribution α over six strategies.

Fraction α_R of the population uses strategy *Ratio* and chooses alternative *in* with probability

$$\hat{p}_R^i = \begin{cases} 0 & \text{if } x_F^o > (1 - \hat{p}_R^r) \cdot x_F^l + \hat{p}_R^r \cdot x_F^r \\ 1 & \text{if } x_F^o < (1 - \hat{p}_R^r) \cdot x_F^l + \hat{p}_R^r \cdot x_F^r \\ 0.5 & \text{otherwise.} \end{cases}$$

Fraction α_L uses strategy $Level - 1$ and chooses alternative *in* with probability

$$\hat{p}_L^i = \begin{cases} 0 & \text{if } x_F^o > 1/2 \cdot x_F^l + 1/2 \cdot x_F^r \\ 1 & \text{if } x_F^o < 1/2 \cdot x_F^l + 1/2 \cdot x_F^r \\ 0.5 & \text{otherwise.} \end{cases}$$

Fraction α_M uses strategy $MaxMin$ and chooses alternative *in* with probability

$$\hat{p}_M^i = \begin{cases} 0 & \text{if } x_F^o > min(x_F^l, x_F^r) \\ 1 & \text{if } x_F^o < min(x_F^l, x_F^r) \\ 0.5 & \text{otherwise.} \end{cases}$$

Fraction α_J uses strategy $JointMx$ and chooses alternative in with probability

$$\hat{p}^i_J = \begin{cases} 0 & \text{if } x^o_F + x^o_S > max(x^l_F + x^l_S, x^r_F + x^r_S) \\ 1 & \text{if } x^o_F + x^o_S < max(x^l_F + x^l_S, x^r_F + x^r_S) \\ 0.5 & \text{otherwise.} \end{cases}$$

Fraction α_W uses strategy $MxWeak$ and chooses alternative in with probability

$$\hat{p}^i_W = \begin{cases} 0 & \text{if } min(x^o_F, x^o_S) > max(min(x^l_F, x^l_S), min(x^r_F, x^r_S)) \\ 1 & \text{if } min(x^o_F, x^o_S) < max(min(x^l_F, x^l_S), min(x^r_F, x^r_S)) \\ 0.5 & \text{otherwise.} \end{cases}$$

Fraction $\alpha_D = 1 - \alpha_R + \alpha_L + \alpha_M + \alpha_J + \alpha_W$ uses strategy $MnDiff$ and chooses alternative in with probability

$$\hat{p}^i_D = \begin{cases} 0 & \text{if } |x^o_F - x^o_S| < min(|x^l_F - x^l_S|, |x^r_F - x^r_S|) \\ 1 & \text{if } |x^o_F - x^o_S| > min(|x^l_F - x^l_S|, |x^r_F - x^r_S|) \\ 0.5 & \text{otherwise.} \end{cases}$$

The probability to observe an in choice that is predicted by the first mover model is given by

$$\hat{p}^i = \alpha_R \cdot \hat{p}^i_R + \alpha_L \cdot \hat{p}^i_L + \alpha_M \cdot \hat{p}^i_M + \alpha_J \cdot \hat{p}^i_J + \alpha_W \cdot \hat{p}^i_W + \alpha_D \cdot \hat{p}^i_D.$$

The probability distribution over the six strategies α is estimated by means of a regression analysis with no intercept and the restriction that the sum of the weights are equal to 1. The parameter estimates are

$$\alpha_R = 0.4348 \quad \alpha_L = 0.1950 \quad \alpha_M = 0.2005 \quad \alpha_J = 0.0665 \quad \alpha_W = 0.0619 \quad \alpha_D = 0.0414$$

and the mean of squared deviations for the estimation set and for the prediction set of the estimated model are

$$MSD^F_{est} = 0.0119 \quad MSD^F_{pre} = 0.0083.$$[18]

5.4 Own Models

The first two models of this Section were submitted to the sub-competition for predicting the behavior of the second mover and the other two models of this Section were submitted to the

[18] The MATLAB code that was used to estimate the free parameters of both models and to compute the mean squared deviations is attached with detailed comments in Appendix 16.13. Notice that the code specifies the model as it is described by Ert et al. (2011). The code that is published on the competition homepage (`https://sites.google.com/site/extformpredcomp/baseline-models/seven-strategies-matlab`) deviates in some parts which are outlined in Appendix 16.14 and Appendix 16.15.

sub-competition for predicting the behavior of the first mover.

5.4.1 Stochastic Utility Maximizer

Second mover S is a stochastic utility maximizer who cares about own welfare, social welfare and equality. The utility of second mover S from choosing $right$ is given by

$$u_S^r = o \cdot w_o + s \cdot w_s + e \cdot w_e. \tag{5.11}$$

Weight $w_o \in [0, 1]$ measures how much S cares about own welfare and indicator

$$o = \begin{cases} 0 & \text{if } x_S^l > x_S^r \\ 1 & \text{if } x_S^l < x_S^r \\ 0.5 & \text{otherwise} \end{cases}$$

indicates which alternative $a \in \{l, r\}$ maximizes her own payoff. Weight $w_s \in [0, 1]$ measures how much S cares about social welfare and indicator

$$s = \begin{cases} 0 & \text{if } x_S^l + x_F^l > x_S^r + x_F^r \\ 1 & \text{if } x_S^l + x_F^l < x_S^r + x_F^r \\ 0.5 & \text{otherwise} \end{cases}$$

indicates which alternative $a \in \{l, r\}$ maximizes the joint payoff. Weight $w_e \in [0, 1]$ measures how much S cares about own welfare and equality (self-biased equality). Indicator

$$e = \begin{cases} 0 & \text{if } x_S^l - |x_S^l - x_F^l| > x_S^r - |x_S^r - x_F^r| \\ 1 & \text{if } x_S^l - |x_S^l - x_F^l| < x_S^r - |x_S^r - x_F^r| \\ 0.5 & \text{otherwise.} \end{cases}$$

indicates which alternative $a \in \{l, r\}$ maximizes the difference of her own payoff and the absolute difference between her own payoff and the other payoff. The sum of weights $w_o + w_s + w_e$ is equal to 1 and the utility of S from alternative $left$ is given by $u_S^l = 1 - u_S^r$ with $u_S^r \in [0, 1]$. This implies that

- $u_S^r = 1$ and $u_S^l = 0$ when only $right$ maximizes own welfare, social welfare and self-biased equality, i.e., $o = s = e = 1$;
- $u_S^r = u_S^l = 0.5$ when both alternatives maximize own welfare, social welfare and equality, i.e., $o = s = e = 0.5$;
- $u_S^r = 0$ and $u_S^l = 1$ when only $left$ maximizes own welfare, social welfare and self-biased equality, i.e., $o = s = e = 0$.

Our specification of social preference does not take directly the payoffs of a game into account; compared to ones of the QRE models that we consider in Sections 5.3.3, 5.3.4 and 5.3.5. Instead the utility of a payoff distribution depends on whether it has certain characteristics relative to the other payoff distribution. However the characteristics themselves depend directly on the payoff distribution. We estimated the model under the assumption that S weights differently depending on whether one alternative $a \in \{l, r\}$ yields a higher own payoff ($o \neq 0.5$) or not ($o = 0.5$). If $o \neq 0.5$, then $w_s = w_e = \alpha$, i.e., she cares about social welfare as much as she cares about self-biased equality when one alternative yields a higher own payoff. If $o = 0.5$, then $w_s = \alpha + \beta$ and $w_e = \alpha$, i.e., she cares more about social welfare than about equality when both alternatives yield the same own payoff. Given her preference responsiveness $\lambda_S \in [0, \infty]$, S chooses alternative *right* with probability $\hat{p}^r$ (Equation 5.4).[19]

The parameter estimates that minimize the mean of squared deviations of the estimation set are

$$\lambda_S = 3.8880 \quad \alpha = 0.1551 \quad \beta = 0.1068$$

and the implied distribution over weights ($w_o \quad w_s \quad w_e$) is equal to ($0.6898 \quad 0.1551 \quad 0.1551$) if $o \neq 0.5$ and equal to ($0.5830 \quad 0.2619 \quad 0.1551$) if $o = 0.5$. The mean of squared deviations for the estimation set and for the prediction set of the estimated model are

$$MSD^S_{est} = 0.0016 \quad MSD^S_{pre} = 0.0038.$$[20]

5.4.2 Take-The-Best

Take-the-best (TTB) is one of several other formal models of fast and frugal heuristics in the adaptive toolbox of bounded rationality (Gigerenzer and Selten, 2001). Fast and frugal heuristics are based on satisficing rather than optimizing and on bounded rationality rather then unbounded rationality (Simon, 1956). Take-the-best was proposed by Gigerenzer and Goldstein (1996) as a process model of how people infer which of two alternatives has a higher value on an unknown criterion. An example of such an inference problem is to answer the question which of two cities has more inhabitants without knowing the correct answer. Although take-the-best is mainly used for inference problems, it is akin to models of heuristics for preferences (Tversky, 1972; Payne et al., 1993) and can be therefore applied to preference problems, e.g., in consumer preference research (Dieckmann et al., 2009).

In our preference model, we use one important property of take-the-best, the underlying lexicographic process that searches sequentially through cues in a given order until the first discriminating cue is found and the individual makes the decision based on this cue. In the take-the-best preference model, the order of the cues reflects the importance of each cue for

[19] The stochastic utility maximizer model uses the same choice rule as the QRE second mover model. This is the reason why we estimate a preference responsiveness parameter λ_S. It was not labelled as a QRE model because it only applies to the second mover and does not involve belief formation.

[20] The MATLAB code that was used to estimate the free parameters of the model and to compute the mean squared deviations is attached with detailed comments in Appendix 16.16. The model can be estimated with less constrains on the distributions over weights. Notice that the implied distributions over weights of the unconstrained estimation (($0.6898 \quad 0.1628 \quad 0.1474$) if $o \neq 0.5$ and ($0.5802 \quad 0.2631 \quad 0.1567$) if $o = 0.5$) are very similar and do not change the MSD scores (see Appendix 16.18).

the individual. A cue indicates the presence of an aspect of a payoff distribution (higher own payoff, higher joint payoff, etc.). If a cue value is 1 the aspect specified by a cue is present; if a cue value is 0 the specified aspect is absent. A cue is discriminating if the cue value of one alternative is 1 while the cue value of the other alternative is 0.

Like other fast and frugal heuristics the take-the-best preference model has three building blocks (Gigerenzer et al., 1999):

1. a search rule that specifies to examine cues sequentially and to compare at each step the cue values between alternatives;
2. a stopping rule that specifies to stop search if a cue is discriminating; and
3. a decision rule that specifies to choose the alternative with the positive cue value if a cue is discriminating.

In contrast, the (stochastically) optimizing individuals in the social preference models that we consider in Sections 5.3.3–5.3.5 compute a utility score for each alternative by weighing and adding all aspects specified in their utility function (e.g. own payoff, inequality, social welfare, reciprocity, and so on), then they compare the alternatives based on their utility score and choose the alternative with the highest utility score. The satisfying individuals in the take-the-best preference model however consider only one aspect at a time and do not weigh and add all aspects at the same time. They do not base their decision on utilities but on the first aspect that discriminates, thereby ignoring all other aspects.

TTB Second Mover Model There is an infinite population of second movers that make their choice between $left$ and $right$ based on n aspects. For each alternative $a \in \{l, r\}$, a cue $c_k^a \in \{0, 1\}$ indicates the presence of an aspect $k \in \{1, 2, ..., n\}$ if $c_i^a = 1$ or the absence of an aspect k if $c_i^a = 0$. This implies that each alternative a can be characterized by a cue profile $c = \{c_1^a, c_2^a, ..., c_n^a\}$.

Each second mover S uses the heuristic take-the-best for choosing between $left$ and $right$, she has a subset of cues $c_S \subseteq c$ that lists the order of the cues S takes into account. The order of cues reflects how important a cue for S is. Take-the-best specifies that S makes a choice if the first cue discriminates between $left$ and $right$. If the first cue does not discriminate, S makes a choice if the second cue discriminates, and so on. If none of the cues in her cue set c_S discriminates, S chooses randomly.

We assume three possible cues $c = \{c_1^a, c_2^a, c_3^a\}$:

- Cue c_1^a indicates the presence or absence of aspect *highest own payoff*, i.e., $c_1^a = 1$ if $x_S^a > x_S^{-a}$ and otherwise $c_1^a = 0$ ($-a$ denotes the other alternative).
- Cue c_2^a indicates the presence or absence of aspect *highest joint payoff*, i.e., $c_2^a = 1$ if $x_S^a + x_F^a > x_S^{-a} + x_F^{-a}$ and otherwise $c_2^a = 0$.
- Cue c_3^a indicates the presence or absence of aspect *highest difference of own payoff and absolute difference of own and other payoff*, i.e., $c_3^a = 1$ if $x_S^a - |x_S^a - x_F^a| > x_S^{-a} - |x_S^{-a} - x_F^{-a}|$ and otherwise $c_3^a = 0$.

There are five types of second movers $S \in \{A, B, C, D, E\}$ that have different sets of cues c_S. Moreover there is a small probability that each second mover S makes a choice error $\epsilon \in [0, 1]$ in the sense that she chooses an unintended alternative if a cue is discriminating. The choice errors that we assume are akin to the trembles assumed by Selten (1975) in his trembling hand perfect equilibrium.

Share s_A of the population consists of type A second movers that make their choice based on one single cue c_1^a. They choose alternative *right* with probability

$$\hat{p}_A^r = \begin{cases} 0 + \epsilon & \text{if } c_1^l = 1 \wedge c_1^r = 0 \\ 1 - \epsilon & \text{if } c_1^l = 0 \wedge c_1^r = 1 \\ 0.5 & \text{otherwise.} \end{cases}$$

Share s_B of the population consists of type B second movers that make their choice based on one single cue c_2^a. They choose alternative *right* with probability

$$\hat{p}_B^r = \begin{cases} 0 + \epsilon & \text{if } c_2^l = 1 \wedge c_2^r = 0 \\ 1 - \epsilon & \text{if } c_2^l = 0 \wedge c_2^r = 1 \\ 0.5 & \text{otherwise.} \end{cases}$$

Share s_C of the population consists of type C second movers that make their choice based on one single cue c_3^a. They choose alternative *right* with probability

$$\hat{p}_C^r = \begin{cases} 0 + \epsilon & \text{if } c_3^l = 1 \wedge c_3^r = 0 \\ 1 - \epsilon & \text{if } c_3^l = 0 \wedge c_3^r = 1 \\ 0.5 & \text{otherwise.} \end{cases}$$

Share s_D of the population consists of type D second movers that make their choice based on cue set $c_D = \{c_1^a, c_2^a\}$. They choose alternative *right* with probability

$$\hat{p}_D^r = \begin{cases} 0 + \epsilon & \text{if } c_1^l = 1 \wedge c_1^r = 0 \\ 1 - \epsilon & \text{if } c_1^l = 0 \wedge c_1^r = 1 \\ \hat{p}_B^r & \text{otherwise.} \end{cases}$$

Share $s_E = 1 - s_A - s_B - s_C - s_D$ of the population consists of type E second movers that make their choice based on cue set $c_E = \{c_1^a, c_3^a\}$. They choose alternative *right* with probability

$$\hat{p}_E^r = \begin{cases} 0 + \epsilon & \text{if } c_1^l = 1 \wedge c_1^r = 0 \\ 1 - \epsilon & \text{if } c_1^l = 0 \wedge c_1^r = 1 \\ \hat{p}_C^r & \text{otherwise.} \end{cases}$$

Thus, the probability to observe a *right* choice that is predicted by the second mover TTB model is given by

$$\hat{p}^r = s_A \cdot \hat{p}^r_A + s_B \cdot \hat{p}^r_B + s_C \cdot \hat{p}^r_C + s_D \cdot \hat{p}^r_D + s_E \cdot \hat{p}^r_E.$$

The parameter estimates that minimize the mean of squared deviations of the estimation set are

$$s_A = 0.2986 \quad s_B = 0.0707 \quad s_C = 0.0637 \quad s_D = 0.3845 \quad s_E = 0.1825 \quad \epsilon = 0.0167$$

and the mean of squared deviations for the estimation set and for the prediction set of the estimated model are

$$MSD^S_{est} = 0.0018 \quad MSD^S_{pre} = 0.0038.$$[21]

5.4.3 Subjective Quantal Response Equilibrium I

In a subjective quantal response equilibrium (Rogers et al., 2009), compared to the QRE considered in Sections 5.3.2 to 5.3.5, the beliefs about the choice probabilities of others may not be consistent with the actual choice probabilities. Although a subjective quantal response equilibrium (SQRE) permits subjective beliefs, it is still an equilibrium model in which choice probabilities are conditional on types common knowledge. We think that this property of SQRE is especially useful for predicting behavior of inexperienced subjects since they do not have the possibility for trial and error learning which would enable their beliefs to adjust to the actual choice probabilities (Binmore, 1992; Rogers et al., 2009).[22]

We consider three types of heterogeneity in our SQRE first mover model. Heterogeneity in preferences, heterogeneity in skill (i.e. preference responsiveness), and heterogeneity in preference beliefs.

- Each first mover has either a high skill or a low skill and a one-step-below belief about the skill of the second movers, i.e., a high skill first mover believes that all second movers have a low skill while a low skill first mover believes that all second movers have no skill. This type of heterogeneity in skills and skill beliefs is based on the one assumed in the discretized truncated subjective quantal response equilibrium considered by Rogers et al. (2009). We differ only in the assumption that high skill individuals believe that there are only others with low skill and *no* others with no skill.
- Each first mover is either selfish or inequality averse. This type of heterogeneity in preferences is similar to the one assumed in the inequality aversion model of Fehr and Schmidt (1999). However, since we want to predict the behavior of inexperienced subjects, we use a SQRE framework that permits choice errors and subjective beliefs while Fehr and Schmidt (1999) use a subgame perfect equilibrium framework in which individuals do not make choice errors and have consistent beliefs which may be more appropriate

[21] The MATLAB code that was used to estimate the free parameters of the model and to compute the mean squared deviations is attached with detailed comments in Appendix 16.20.

[22] With actual choice probabilities we refer to model predictions and not to the empirically observed choice probabilities.

for predicting the behavior of experienced subjects.

- Lastly, each first mover has either a self-centered belief or a pessimistic belief about the preferences of others. Each self-centered first mover believes that he plays against a second mover who has the same preference structure[23] and each pessimistic first mover believes that he plays against a "bad guy" who wants to minimize his expected utility.[24]

SQRE First Mover Model There is an infinite population of first movers with different characteristics. Each first mover F is a stochastic expected utility maximizer who has either a high skill or a low skill (and in both cases a one-step-below belief about the skill of the second movers) who has either selfish preferences or inequality averse preferences and who has either a self-centered belief or a pessimist belief about the preferences of the second movers.

- **High Skill and Low Skill** Given his level of sophistication $k \in \{1, 2\}$, the skill of each first mover is defined by $\lambda_F = k \cdot \gamma$ with $\gamma \in [0, \infty]$ and his one-step-below belief about the skill of the second movers is defined by $\lambda_S^B = (k-1) \cdot \gamma$. High skill first movers have a level of sophistication of $k = 2$ while low skill first movers have a lower level of sophistication of $k = 1$. The strategic hierarchy that we consider is similar to the one studied by Nagel (1995) in the context of beauty contest games where k-step individuals think that the other individuals do $k - 1$ steps of reasoning as well as to the one of the limited-step types of Stahl and Wilson (1995).
- **Selfishness and Inequality Aversion** The preferences of a selfish first mover are represented by a standard utility function that is defined in Equation 5.3 and the preferences of an inequality averse first mover are represented by the utility function of Fehr and Schmidt (1999) that is defined in Equation 5.7, with the only difference that in our model β_F can be greater than α_F.
- **Self-Centered Beliefs and Pessimistic Beliefs** A self-centered first mover believes that the second mover S has the same preference structure. This implies that a self-centered and selfish first mover believes that the utility function of S is defined by Equation 5.1 and that a self-centered and inequality averse first mover believes that the utility function of S is defined by Equation 5.6, with the only difference that in our model β_S can be greater than α_S. A pessimistic first mover F stochastically maximizes his expected utility under the belief that the second mover is a "bad guy" who stochastically minimizes the utility of F (or, which is equivalent, who maximizes the negative of the utility of F).

[23] McKelvey et al. (2000) consider a SQRE where players differ in their skill and believe self-centered that all other individuals have the same skill as they have. We transfer this logic to the context of preferences. Our motivation for this assumption is based on the false consensus effect (Ross et al., 1977; Marks and Miller, 1987) according to which people have the tendency to believe that their own preferences among other things are shared by others. Notice moreover that the 7S first mover model (see Section 5.3.6) also assumes self-centered beliefs.

[24] In other words: the first movers play a stochastic version of strategy $MaxMin$ (see Section 5.3.6) with utilities. $MaxMin$ is a strategy which maximizes one's own minimum payoff. In our case the first mover does something similar: he stochastically maximizes his own expected utility (not payoff) under the belief that the second mover minimizes stochastically his utility, i.e., the utility of the first mover.

This implies 2 x 2 x 2 = 8 different types of first movers. The probability distribution f over types F_t with $t \in \{1, 2, ..., 8\}$ depends on the share of selfish first movers $s_1 \in [0, 1]$ and on the share of inequality averse first movers $1 - s_1$, on the share of high skill first movers $s_2 \in [0, 1]$ and on the share of low skill first movers $1 - s_2$ and on the share of self-centered first movers $s_3 \in [0, 1]$ and on the share of pessimistic first movers $1 - s_3$.

The characteristics of each type F_t are summarized in Table 5.6.

Table 5.6
Characteristics of Each Type F_t

Possible Characteristics	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
s_1 are selfish	$\surd$	$\surd$	$\surd$	$\surd$				
$1 - s_1$ are inequality-averse					$\surd$	$\surd$	$\surd$	$\surd$
s_2 have high skill	$\surd$	$\surd$			$\surd$	$\surd$		
$1 - s_2$ have low skill			$\surd$	$\surd$			$\surd$	$\surd$
s_3 are self-centered	$\surd$		$\surd$		$\surd$		$\surd$	
$1 - s_3$ are pessimistic		$\surd$		$\surd$		$\surd$		$\surd$

The probability distribution f over first mover types F_t is given in Table 5.7.

Table 5.7
Distribution over Types of First Movers

t	f_t	λ_{F_t}	$\lambda_S^{B_t}$	u_{F_t}	$u_S^{B_t}$
1	$s_1 s_2 s_3$	2γ	γ	Eq. 5.3	Eq. 5.1
2	$s_1 s_2 (1 - s_3)$	2γ	γ		$-u_{F_t}$
3	$s_1 (1 - s_2) s_3$	γ	0		Eq. 5.1
4	$s_1 (1 - s_2)(1 - s_3)$	γ	0		$-u_{F_t}$
5	$(1 - s_1) s_2 s_3$	2γ	γ	Eq. 5.7	Eq. 5.6
6	$(1 - s_1) s_2 (1 - s_3)$	2γ	γ		$-u_{F_t}$
7	$(1 - s_1)(1 - s_2) s_3$	γ	0		Eq. 5.6
8	$(1 - s_1)(1 - s_2)(1 - s_3)$	γ	0		$-u_{F_t}$

The preferences of each inequality averse first mover F_t with $t \in \{5, 6, 7, 8\}$ are determined by α_F and β_F. The preference belief of each self-centered and inequality-averse first mover F_t with $t \in \{5, 7\}$ is determined by α_S^B and β_S^B.

Each type F_t chooses *in* with probability $\hat{p}_t^i$ that is defined by Equation 5.5. The SQRE first mover model predicts a probability to observe an *in* choice of

$$\hat{p}^i = \sum_{t=1}^{8} f_t \cdot \hat{p}_t^i.$$

The model has eight free parameters: Parameters s_1, s_2 and s_3 determine the distribution f over types F_t. Parameter γ determines the skill of each type F_t with $t = \{1, 2, ..., 8\}$, parameters α_F and β_F determine the preferences of each inequality averse first mover F_t with $t = \{5, 6, 7, 8\}$ and parameters α_S^B and β_S^B determine the preference belief of each self-centered and inequality averse first mover F_t with $t = \{5, 7\}$.

Based on an unconstrained estimation of the model[25] and subsequent cross validations, we decided to fix the preference parameters in the following way: $\alpha_F = 0$ and $\beta_F = \alpha_S^B = \beta_S^B = \delta$. This implies that each inequality averse first mover cares only about advantageous inequality and not about disadvantageous inequality. If he is moreover self-centered, he believes that all second movers care about both types of inequality; namely to the same degree as he cares about advantageous inequality.

For the constrained SQRE model that we submitted to the competition, the parameter estimates that minimize the mean of squared deviations estimation set are

$$\gamma = 1.3170 \quad \delta = 0.2174 \quad s_1 = 0.5131 \quad s_2 = 0.7029 \quad s_3 = 0.8793$$

and the mean of squared deviations for the estimation set and for the prediction set of the estimated model are

$$MSD_{est}^F = 0.0050 \quad MSD_{pre}^F = 0.0094.$$[26]

5.4.4 Subjective Quantal Response Equilibrium II

In this Section we extend the subjective quantal response equilibrium (SQRE) model that we consider in Section 5.4.3 so that it captures a concept of social psychology that is called social value orientation (SVO). Social value orientation is understood as a person's preference about how to allocate resources between the self and the other person. It describes individual differences in how much the others' welfare in relation to the own welfare is weighted (Messick and McClintock, 1968; Kelley and Thibaut, 1978). Typically, it is distinguished between two types of people: proselfs and prosocials (Van Lange, 1999; Van Lange et al., 2007; Murphy et al., 2011). Proselfs are motivated to merely maximize their own outcome. Thus, they have no or even a negative other-regarding preference. In contrast, prosocials want to

[25] The parameter estimates of the unconstrained estimation are $\gamma = 1.283, \alpha_F = 0.019, \beta_F = 0.1969, \alpha_S^B = 0.2268, \beta_S^B = 0.2497, s_1 = 0.4852, s_2 = 0.7093, s_3 = 0.8761$ with $MSD_{est}^F = 0.0049, MSD_{pre}^F = 0.0093$.

[26] The MATLAB code that was used to estimate the free parameters of the submitted model and to compute the mean squared deviations is attached with detailed comments in Appendix 16.22. The code for the unconstrained estimation is attached in Appendix 16.24.

maximize both persons' outcomes as well as to minimize the difference between persons' outcomes. Thus, they value both efficiency and equality in outcomes. Notice that both prosocials and inequality-averse people according to Fehr and Schmidt (1999) value equality in outcomes, whereas only prosocials value additionally efficiency in outcomes. We capture this additional element in the utility function of prosocials by adding a concern for efficiency to the utility function of the inequality-averse types that we consider in Section 5.4.3.

SVO-SQRE First Mover Model The preferences of a proself first mover are represented by a standard utility function that is defined in Equation 5.3 and a self-centered proself first mover believes that the utility function of the second mover is defined by Equation 5.1. The preferences of a prosocial first mover are represented by a utility function that is given by

$$u_F^a = x_F^a - \alpha_F \cdot max(0, x_S^a - x_F^a) - \beta_F \cdot max(0, x_F^a - x_S^a) + \omega_F \cdot (x_F^a + x_S^a), \tag{5.12}$$

with $\beta_F \in [0,1)$, $\alpha_F \geq \beta_F$ and efficiency parameter $\omega_F \geq 0$. A self-centered prosocial first mover believes that the utility function of the second mover is defined by

$$u_S^a = x_S^a - \alpha_S \cdot max(0, x_F^a - x_S^a) - \beta_S \cdot max(0, x_S^a - x_F^a) + \omega_S \cdot (x_F^a + x_S^a), \tag{5.13}$$

with $\beta_S \in [0,1)$, $\alpha_S \geq \beta_F$ and efficiency parameter $\omega_S \geq 0$. The assumed specification of social preferences extends the utility function of Fehr and Schmidt (1999) by a fourth term that captures the utility of a prosocial first mover from social welfare (i.e. efficiency).

There are 2 x 2 x 2 = 8 different types of first movers. The probability distribution over types $t \in \{1, 2, ..., 8\}$ of first movers F_t depends on the share of proselfs $s_1 \in [0,1]$ and the share of prosocials $1 - s_1$, on the share of high skill first movers $s_2 \in [0,1]$ and the share of low skill first movers $1 - s_2$ and on the share of self-centered first movers $s_3 \in [0,1]$ and the share of pessimistic first movers $1 - s_3$. Table 5.7 presents the probability distribution f over first mover types F_t. Each type F_t chooses *in* with probability $\hat{p}_t^i$ that is defined by Equation 5.5. The SVO-SQRE first mover model predicts a probability to observe an *in* choice of

$$\hat{p}^i = \sum_{t=1}^{8} f_t \cdot \hat{p}_t^i.$$

The model has ten free parameters: Parameters s_1, s_2 and s_3 determine the distribution f over types F_t. Parameter γ determines the skill of each type F_t with $t = \{1, 2, ..., 8\}$, parameters α_F, β_F and ω_F determine the preferences of each prosocial first mover F_t with $t = \{5, 6, 7, 8\}$ and parameters α_S^B, β_S^B and ω_S^B determine the preference belief of each self-centered and prosocial first mover F_t with $t = \{5, 7\}$.

Table 5.8
Distribution over Types of First Movers

t	f_t	λ_{F_t}	$\lambda_S^{B_t}$	u_{F_t}	$u_S^{B_t}$
1	$s_1 s_2 s_3$	2γ	γ	Eq. 5.3	Eq. 5.1
2	$s_1 s_2(1-s_3)$	2γ	γ		$-u_{F_t}$
3	$s_1(1-s_2)s_3$	γ	0		Eq. 5.1
4	$s_1(1-s_2)(1-s_3)$	γ	0		$-u_{F_t}$
5	$(1-s_1)s_2 s_3$	2γ	γ	Eq. 5.12	Eq. 5.13
6	$(1-s_1)s_2(1-s_3)$	2γ	γ		$-u_{F_t}$
7	$(1-s_1)(1-s_2)s_3$	γ	0		Eq. 5.13
8	$(1-s_1)(1-s_2)(1-s_3)$	γ	0		$-u_{F_t}$

The preferences of each prosocial first mover F_t with $t \in \{5,6,7,8\}$ are determined by his degrees of inequality aversion α_F, β_F and his efficiency parameter ω_F. The preference belief of each self-centered prosocial first mover F_t with $t \in \{5,7\}$ is determined by α_S^B, β_S^B and ω_S^B.

Based on an unconstrained estimation of the model[27] and subsequent cross validations, we decided to fix the preference parameters in the following way: $\alpha_F = \beta_F = \omega_F = \beta_S^B = \omega_S^B = \delta$ and $\alpha_S^{B_t} = \delta + \epsilon$. This means that each prosocial first mover weighs all other regarding components with δ. If he is moreover self-centered, he believes that all second movers weigh also advantageous inequality or social welfare with δ and disadvantageous inequality with $\delta+\epsilon$, i.e., he believes that they care more about disadvantageous inequality than he does.

For the constrained SVO-SQRE first mover model that we submitted to the competition, the parameter estimates that minimize the mean of squared deviations estimation set are

$$\gamma = 1.1940 \quad \delta = 0.1859 \quad \epsilon = 0.3442 \quad s_1 = 0.5703 \quad s_2 = 0.7036 \quad s_3 = 0.8704$$

and the mean of squared deviations for the estimation set and for the prediction set of the estimated model are

$$MSD_{est}^F = 0.0051 \quad MSD_{pre}^F = 0.0090.$$ [28]

[27] The parameter estimates of the unconstrained estimation are $\gamma = 1.239, \alpha_F = 0.2726, \beta_F = 0.2726, \omega_F = 0.1869, \alpha_S^{B_t} = 0.5998, \beta_S^{B_t} = 0.1746, \omega_S^{B_t} = 0.2296, s_1 = 0.5835, s_2 = 0.6991, s_3 = 0.8628$ with $MSD_{est}^F = 0.0047, MSD_{pre}^F = 0.0091$.

[28] The MATLAB code that was used to estimate the free parameters of the submitted model and to compute the mean squared deviations is attached with detailed comments in Appendix 16.26. The code for the unconstrained estimation is attached in Appendix 16.28.

5.5 Discussion and Results

5.5.1 Comparison of Models

In Table 5.9 we compare the fitting and prediction performance of all models.

Table 5.9
Fitting versus Predicting: Performance of All Models

Section	Model	MSD^F_{est}	MSD^F_{pre}	MSD^S_{est}	MSD^S_{pre}
5.3.1	SPE	0.0545	0.0532	0.0105	0.0071
5.3.2	QRE	0.0170	0.0141	0.0092	0.0057
5.3.3	FS-QRE	0.0118	0.0140	0.0082	0.0056
5.3.4	BO-QRE	0.0141	0.0172	0.0073	0.0056
5.3.5	CR-QRE	0.0112	0.0143	0.0042	0.0067
5.3.6	7S	0.0119	0.0083	0.0029	0.0043
5.4.1	SUM	-	-	0.0016	0.0038
5.4.2	TTB	-	-	0.0018	0.0038
5.4.3	SQRE	0.0050	0.0094	-	-
5.4.4	SVO-SQRE	0.0051	0.0090	-	-

The second mover results show that choice errors increase the out-of-sample fit (S.1) but social preferences do not (S.2); and that the three models with the highest out-of-sample fit are 7S, SUM and TTB (S.3–S.5).

Result S.1 The observed deviations from (stochastic) own payoff maximization (cf. SPE and QRE) are larger in the games of the estimation set than in the games of the prediction set. The introduction of choice errors (cf. QRE) increases both the within-sample fit and the out-of-sample fit (compared to SPE).

Result S.2 Introducing social preferences additionally to choice errors (cf. FS-QRE, BO-QRE and CR-QRE) increases the within-sample fit *but not* the out-of-sample fit (compared to QRE). Charness-Rabin preferences even decrease the out-of-sample fit.

Result S.3 Conceptualizing SPE as a simple strategy and adding other strategies with different target values (nice rational play, joint payoff maximization, maximization of the payoff of the weaker player and minimization of payoff differences; cf. 7S) increases the within-sample fit and the out-of-sample fit (compared to all other baseline models).

Result S.4 A combination of choice errors and a social preference specification that takes into account own welfare maximization, social welfare maximization and self-biased equality maximization (cf. SUM) increases the within-sample fit and the out-of-sample fit (compared to 7S).

Result S.5 The introduction of a lexicographic decision process, individual heterogeneity in the order of decision-relevant aspects (own welfare, social welfare and equality) and a small choice error (cf. TTB) increases the within-sample fit and the out-of-sample fit (compared to 7S).

The first mover results show that choice errors increase the out-of-sample fit (F.1) but social preferences do not (F.2); and that the three models with the highest out-of-sample fit are 7S, SQRE and SVO-SQRE (F.3-F.5).

Result F.1 The observed deviations from (stochastic) own payoff maximization and consistent beliefs (cf. SPE and QRE) are larger in the games of the estimation set than in the games of the prediction set. The introduction of choice errors (cf. QRE) increases both the within-sample fit and the out-of-sample fit (compared to SPE) and the increase in accuracy in both statistics is greater compared to the second mover.

Result F.2 Like in the second mover case, combining choice errors with social preferences increases the within-sample fit *but not* the out-of-sample fit (compared to QRE). Bolton-Ockenfels preferences even decrease the out-of-sample fit.

Result F.3 Assuming a mix of simple strategies with different target values and subjective beliefs (own payoff maximization and self-centered belief (SCB), own payoff maximization and level-1 reasoning, own minimum payoff maximization, joint payoff maximization and SCB, maximization of the payoff of the weaker player and SCB and minimization of payoff differences and SCB) increases the within-sample fit (compared to QRE) and the out-of-sample fit (compared to all other models).

Result F.4 Assuming heterogeneity in skills (more versus less choice errors), subjective one-step below beliefs about the skills of others (others make more choice errors), heterogeneity in preferences (selfishness or inequality aversion) and heterogeneity in subjective beliefs about the preferences of others (self-centered versus pessimistic; cf. SQRE) increases the within-sample fit (compared to all baselines models) and the out-of sample fit (compared to all baseline models but 7S).

Result F.5 Assuming heterogeneity in skills combined with subjective one-step below beliefs, heterogeneity in preferences (proselfs versus prosocials) and heterogeneity in subjective preference beliefs (cf. SVO-SQRE) increases the within-sample fit (compared to all baselines models) and the out-of sample fit (compared to all baseline models but 7S).

5.5.2 Reliability of Competition Results

Training a model and evaluating its predictive performance on the same data may imply results that are overoptimistic (Larson, 1931). The validation procedure of the competition addresses this problem by using part of data (the estimation set) for training each model and the other part (prediction set) for validation.

Nevertheless, one could still be concerned about the reliability of the competition validation results that we consider in Section 5.5.1 and ask questions of the following type: What would have happened if the same games were split differently, or if other games were used for training and validating the models?

We address these types of questions by conducting more than one data split with the given data sets. Each data split yields a new validation result for each model. The mean of the obtained validation results yields a cross validation result (Stone, 1974; Geisser, 1975; Arlot and Celisse, 2010). In particular, we apply two different leave-one-out cross validations (LOOCV I and LOOCV II) to the data and compare the obtained cross validation results to the validation results of the competition procedure.

Validation Procedures Table 5.10 contrasts the cross validation procedures to the competition procedure.

Table 5.10
Comparison of Validation Procedures

Validation Procedure	Competition	LOOCV I	LOOCV II
Number of Splits	1	120	120
Size of Training Set	120	239	119
Composition of Training Set	est	$est \wedge pre$	pre
Size of Validation Set	120	1	1
Composition of Validation Set	pre	pre	pre

What we do in the competition procedure is to use the estimation set est for training each model. Then we compute for each model the squared deviations for the 120 games of the prediction set pre and take the mean to obtain a validation result.

In contrast, the LOOCV procedure I works as follows: We compute for each model the squared deviation for the first game of pre after using all games of est and remaining games 2 to 120 of pre for training the models. Then we compute for each model the squared deviation for the second game of pre after using all games of est and remaining games 1 and 3 to 120 of pre for training the models. We repeat this leave-one-out procedure with the third game of pre, forth game of pre, and so on. Then, we obtain for each model a cross validation result by taking the mean of the obtained 120 squared deviations. The LOOCV procedure II works analogously but there we use only pre for training the models.

All three validation procedures have in common that they predict the same 120 games of *pre*, i.e., none of the validation sets contains games that may be used for formulating the models. However, there is enough variation within the validation procedures for addressing the reliability question that we pose. The competition procedure splits the data only once and predicts therefore the 120 games of *pre* at once. The LOOCV procedures conduct 120 data splits, reestimate the models each time and predict only one game of *pre* at the time; the latter also differ in the size (239 vs 119) and the composition (*est* and *pre* vs *pre*) of the training set.[29]

First Mover Models Table 5.11 presents the out-of-sample fit of each first mover model for each validation procedure.

Table 5.11
Out-of-Sample Fit of First Mover Models For Each Validation Procedure

Section	MSD^F_{pre}	Competition	LOOCV I	LOOCV II
5.3.1	SPE	0.0532	0.0532	0.0532
5.3.2	QRE	0.0141	0.0137	0.0139
5.3.3	FS-QRE	0.0140	0.0121	0.0123
5.3.4	BO-QRE	0.0172	0.0146	0.0144
5.3.5	CR-QRE	0.0143	0.0121	0.0118
5.3.6	7S	0.0083	0.0085	0.0090
5.4.3	SQRE	0.0094	0.0082	0.0086
5.4.4	SVO-SQRE	0.0090	0.0088	0.0079

A comparison of the out-of-sample fits shows that the ranking of a first mover model may depend on the validation procedure if its out-of-sample fit in the competition procedure is close to other models (F.6) and otherwise not, i.e., SPE is always worse than each QRE; and each QRE is always worse than 7S or each SQRE (F.7).

Result F.6 If the out-of-sample fits are very close in the competition procedure, the models change ranks in the cross validations. The social preference QRE models perform better in both cross validations and the ranking of the best three models changes depending on the validation procedure in the following way:

- 7S > SVO-SQRE > SQRE (Competition)
- SQRE > 7S > SVO-SQRE (LOOCV I)
- SVO-SQRE > SQRE > 7S (LOOCV II)

Result F.7 If the out-of-sample fits are far away in the competition procedure,

[29] The MATLAB code that was used to conduct both LOOCV is attached in Appendix 16.30.

the models do not change ranks, i.e., in each validation procedure SPE is worse than each QRE; and each QRE is worse than 7S and each SQRE.

Second Mover Models In Table 5.12 we present the out-of-sample fit of each second mover model for each validation procedure.

Table 5.12
Out-of-Sample Fit of Second Mover Models For Each Validation Procedure

Section	MSD^S_{pre}	Competition	LOOCV I	LOOCV II
5.3.1	SPE	0.0071	0.0071	0.0071
5.3.2	QRE	0.0057	0.0057	0.0058
5.3.3	FS-QRE	0.0056	0.0056	0.0058
5.3.4	BO-QRE	0.0056	0.0055	0.0056
5.3.5	CR-QRE	0.0067	0.0051	0.0056
5.3.6	7S	0.0043	0.0042	0.0044
5.4.1	SUM	0.0038	0.0038	0.0040
5.4.2	TTB	0.0038	0.0040	0.0047

A comparison of the out-of-sample fits confirms the results that we obtain for the first mover models. In particular, it is for each validation procedure true that SPE is worse than each QRE; and that each QRE is worse than 7S, SUM or TTB.

Result S.6 The ranks of the models in the cross validations change if their out-of-sample fits are very close in the competition procedure; although the out-of-sample fits of most second mover models do not change considerably. In particular, the CR-QRE model performs a lot better in both cross validations and the ranking of the best three models depends in the following way on the validation procedure:

- SUM = TTB > 7S (Competition)
- SUM > TTB > 7S (LOOCV I)
- SUM > 7S > TTB (LOOCV II)

Result S.7 If the out-of-sample fits are far away in the competition procedure, the models do not change ranks in the cross validations. In particular, SPE is always worse than each QRE; and each QRE is always worse than 7S, SUM or TTB.

5.5.3 Averaging Good Predictions Yields Better Predictions

In Section 5.5.2 three first mover models (7S, SQRE, SVO SQRE) and three second mover models (7S, SUM, TTB) perform best between different validation procedures. Inspired by the idea of forecast combination, we show in this Section that averaging predictions of good models yields better (but not optimal) predictions.

Forecast combination (Bates and Granger, 1969) is widely used in econometrics for improving forecast accuracy (Clemen, 1989; Hendry and Clements, 2004). It is still controversial how to pick the forecast weights (Hansen, 2008); but simple averaging (equal weights) works reasonably well (Stock and Watson, 2004). Since our main concern is not about showing how to average properly but about improving predictive accuracy by averaging, we use equal weights to keep things simple.

Simple Averaging We consider $n = 8$ models for each individual $i \in \{F, S\}$. This implies that a model set M_j with $j \in \{1, 2, ..., m\}$ can include $k \in \{1, 2, ..., n\}$ models. The number of possible model sets is given by

$$m = \sum_{k=1}^{n} \frac{n!}{k!(n-k)!}.$$

Given $n = 8$, $m = 255$.

We use a simple averaging method that weighs the prediction vectors of a model set equally. Each model $l \in \{1, ..., n\}$ generates a vector of predictions p_l. Thus, the prediction vector of each model set M_j is given by the average of the prediction vectors

$$p_j = \sum_{l=1}^{k} \frac{1}{k} p_l.$$

We evaluate for each splitting procedure the performance of the best three models in comparison to the best individual model within a procedure, the optimal model set within a procedure and the optimal model set between procedures. The model set of the *best three models* contains the three models that performed best individually between the three validation procedures. The model set that is *optimal within a procedure* has the lowest mean predictive error for a given validation procedure (relative to the other $m-1$ model sets). The model set that is *optimal between procedures* has the lowest average of the three within procedure mean predictive errors (relative to the other $m-1$ model sets).[30]

First Mover Models Table 5.13 presents the results for the first mover models. The results show that averaging good predictions yields better but not optimal predictions.

Result F.8 The set of the best three models (7S, SQRE, SVO-SQRE) outperforms

[30] The MATLAB code that was used to conduct both averaging of predictions is attached in Appendix 16.31.

the best performing model of each splitting procedure; however it is not the optimal model set within a procedure and between procedures.

Result F.9 The optimal model set between procedures contains a model (CR-QRE) that is not one of the best three models; within a procedure the optimal model set is either smaller or contains models (CR-QRE or FS-QRE) that are not under the best three models.

Table 5.13
First Mover Models: Simple Prediction Averaging

MSD^F_{pre}	Competition	LOOCV-I	LOOCV-II
Best Model	$.0083^1$	$.0082^2$	$.0079^3$
Best Three Models	$.0072^4$	$.0071^4$	$.0074^4$
Optimal Within A Procedure	$.0068^5$	$.0067^6$	$.0063^7$
Optimal Between Procedures	$.0071^7$	$.0069^7$	$.0063^7$
VSD^F_{pre}	Competition	LOOCV-I	LOOCV-II
Best Model	.000207	.000230	.000198
Best Three Models	.000164	.000157	.000175
Optimal Within A Procedure	.000146	.000150	.000148
Optimal Between Procedures	.000153	.000157	.000148

VSD^F_{pre} denotes the variance of the predictive errors of a first mover model set. Each first mover model set M^F_j is indicated by a superscript j with M^F_1={7S}, M^F_2={SQRE}, M^F_3={SVO-SQRE}, M^F_4={7S, SQRE, SVO-SQRE}, M^F_5={7S, SVO-SQRE}, M^F_6={FS-QRE, 7S, SQRE} and M^F_7={CR-QRE, 7S, SVO-SQRE}.

Both results can be explained by considering that the predictions of the best three models (set M^F_4) have a similar mean predictive error (see Table 5.11) but are not perfectly correlated (see Table 8). The basic mechanism that is exploited works as follows: If two models with the same mean predictive error are not perfectly correlated in their predictions than one model performs better in some games and the other model performs better in other games. Averaging predictions that are equally good between games but differ in their goodness within each game decreases thus the variance and the mean predictive error.

However, set M^F_4 contains only the models with the lowest mean predictive error but not the ones with the lowest correlation of predictions. Since the mean predictive error of the average predictions depends on both, M^F_4 must not be the optimal set which is actually the case within each procedure and between procedures. In particular, the optimal sets M^F_5, M^F_6 and M^F_7 contain only one of two SQRE models that have highly correlated predictions and may contain a social preference QRE model that compensates its higher mean predictive error with a lower correlation to the predictions of each other model in the set.

Second Mover Models Table 5.14 presents the results for the second mover models. The second mover results replicate the first mover results and show again that averaging good predictions yields better but not optimal predictions.

> **Result S.8** The set of the best three models (7S, SUM, TTB) is not worse than the best model of each validation procedure; however it is not the optimal model set within each procedure and between procedures. **Result S.9** The optimal model set between procedures contains a model (QRE) that is not one of the best three models; within a procedure each optimal model set is bigger and contains models (QRE or CR-QRE) that are not under the best three models.

Table 5.14
Second Mover Models: Simple Prediction Averaging

MSD^S_{pre}	Competition	LOOCV-I	LOOCV-II
Best Model	$.0038^1$	$.0038^1$	$.0040^1$
Best Three Models	$.0037^2$	$.0038^2$	$.0041^2$
Optimal Within A Procedure	$.0033^3$	$.0034^3$	$.0036^4$
Optimal Between Procedures	$.0033^5$	$.0034^5$	$.0037^5$
VSD^S_{pre}	Competition	LOOCV-I	LOOCV-II
Best Model	.000111	.000111	.000128
Best Three Models	.000101	.000101	.000123
Optimal Within A Procedure	.000075	.000081	.000100
Optimal Between Procedure	.000073	.000079	.000106

VSD^S_{pre} denotes the variance of the predictive errors of a second mover model set. Each second mover model set M^S_j is indicated by a superscript j with M_1 ={SUM}, M_2 ={7S, SUM, TTB}, M_3 ={QRE, 7S, SUM, TTB}, M_4 ={CR-QRE, 7S, SUM, TTB} and M_5 ={QRE, 7S, SUM}. Notice that TTB is beside SUM the best model in the competition procedure with a variance of 0.000113.

Both second mover results replicate the first mover results and depend on the same mechanism.[31] The bottom line for both cases is that it seems to be a good starting point to select a set of good models and average their predictions for making more accurate and less volatile predictions. However, one can do better by adding a semi-good model that compensates its higher mean predictive error with lower correlations to the predictions of the other models.

[31] The correlation matrix and the predictive error matrix for the second mover models are given in Tables 10 and 11.

5.6 Summary

Ert et al. (2011) designed both choice prediction competitions with the aim to answer two unresolved questions: Which quantitative model is the best for predicting behavior in a broad set of extensive form games? And, what is the relative importance of each behavioral tendency captured in a model? They fitted for each sub-competition three social preference models and a strategy mix model (seven strategies). The social preference models consider the behavioral tendencies by means of social utility functions and stochastic choice functions; whereas the seven strategies model assumes the use of simple strategies. The surprising result of their study was that the seven strategies model outperformed the popular social preference models in fitting the choice behavior in both sub-competitions. This preliminary result indicated that the strategy mix model might also predict better. Against this background our study contributes four new insights.

(1) We determined – based on the validation procedure of the competition – within-sample fits and out-of-sample fits for quantal response equilibrium models that differ from the ones used by Ert et al. (2011). The models achieved better within-sample fits than the seven strategies model but not better out-of-sample fits.

(2) We determined – based on the validation procedure of the competition — the within-sample fit and the out-of-sample fit of each model that we submitted to the competition. The submitted second mover models have higher within-sample and out-of-sample fits than the seven strategies model. Thus we conclude that one can predict better with an alternative strategy mix model (take-the-best) as well as a stochastic preference model (stochastic utility maximizer). The first mover subjective quantal response equilibrium models have higher within-sample fits but not higher out-of-sample fits than the seven strategies model; however the out-of-sample fits of our submitted models are still higher than the ones of the quantal response equilibrium models. Thus we conclude that one can achieve better predictions with subjective quantal response equilibrium models compared to quantal response equilibrium models but not better predictions compared to the seven strategies model.

(3) We assessed the reliability of the results of the competition validation procedure by comparing them to out-of-sample fits of two cross validation procedures. The main new insight is that only groups of models with substantially different out-of-sample fits in the competition procedure do not change ranks in the cross validations. The choice behavior of second movers is predicted by subgame perfect equilibrium always worse than by each quantal response equilibrium and each quantal equilibrium model predicts always worse than seven strategies, take-the-best and stochastic utility maximizer. The choice behavior of the first movers is predicted by sub game perfect equilibrium always worse than each quantal response equilibrium and each quantal response equilibrium model predicts always worse than seven strategies and each subjective quantal response equilibrium.

(4) We determined the out-of-sample fit of averaged predictions of each possible model set with the validation procedure of the competition and the two cross validation procedures. The set of the best three models yields better predictions than any single model. The goodness of

the averaged predictions depend thereby on the goodness of the predictions of each model and the correlation to the predictions of the other models in the set. The main new insights are that averaging predictions of good models yields better and less volatile predictions; moreover models with low prediction correlations do not need to have the highest predictive power to be useful for making better predictions.

Table 5.15
Behavioral Tendencies Modeled by Best Second Mover Models

Explanation for observed behavior	7S	SUM	TTB
Interest in own welfare	√	√	√
Interest in social welfare	√	√	√
Higher interest in social welfare if costless	√	√	√
Interest in equality	√	√	√
Higher interest in equality if costless		√	√
Tendency to make choice errors		√	√
Interest in helping the weaker individual	√		

The best three second mover models are seven strategies, stochastic utility maximizer and take-the-best (7S, SUM, TTB). As depicted in Table 5.15 some behavioral tendencies are modeled by all three models while others are not.

Table 5.16
Behavioral Tendencies Modeled by Best First Mover Models

Explanation for observed behavior	7S	SQRE	SVO-SQRE
Interest in own welfare	√	√	√
Interest in equality	√	√	√
Pessimistic beliefs	√	√	√
Self-centered beliefs	√	√	√
Level-1 reasoning	√	√	√
Level-2 reasoning		√	√
Choice errors		√	√
Interest in social welfare	√		√
Interest in helping the weaker individual	√		

Each model considers an interest in own welfare, social welfare and equality as well as a higher interest in social welfare if both alternatives yield the same own payoff. 7S includes additionally an interest in helping the weaker individual. SUM and TTB include instead a tendency to make choice errors and a higher interest in equality if both alternatives yield the same own payoff. Although the latter ones capture the same behavioral tendencies they do it very differently: SUM assumes an average decision-maker who weighs each aspect of the payoff distribution, while TTB assumes a mix of types that use the same lexicographic heuristic but take different aspects of the payoff distribution into account.

The best three first mover models are seven strategies and both subjective quantal response equilibrium models (7S, SQRE, SVO-SQRE). Table 5.16 summarizes the behavioral tendencies that are modeled.

Each model contains an interest in own welfare and equality, pessimistic and self-centered beliefs as well as level-1 reasoning. SQRE and SVO-SQRE include additionally level-2 reasoning and choice errors. 7S and SVO-SQRE include an interest in social welfare and 7S includes furthermore an interest in helping the weaker individual.

Our study shows that subjective quantal response equilibrium models and strategy mix models make the most useful predictions. Moreover we show that better and less volatile predictions can be achieved by simple prediction averaging. Our most surprising result is that the goodness of averaged predictions can even be increased if "weaker" but less correlated predictions of standard equilibrium models are considered. We hope that our study indicates the value of each behavioral model and we look forward to future research on this topic.

Bibliography

Andreoni, J. (1990). Impure Altruism and Donations to Public Goods: A Theory of Warm-Glow Giving. *The Economic Journal*, 100(401):464–477.

Andreoni, J. and Miller, J. (2002). Giving According to GARP: An Experimental Test of the Consistency of Preferences for Altruism. *Econometrica*, 70(2):737–753.

Arlot, S. and Celisse, A. (2010). A Survey of Cross-Validation Procedures for Model Selection. *Statistics Surveys*, 4:40–79.

Barron, G. and Erev, I. (2003). Small Feedback-Based Decisions and Their Limited Correspondence to Description-Based Decisions. *Journal of Behavioral Decision Making*, 16:215–233.

Bates, J. M. and Granger, C. W. J. (1969). The Combination of Forecasts. *Operational Research Society*, 20(4):451–468.

Benartzi, S. and Thaler, R. H. (2001). Naive Diversification Strategies in Defined Contribution Saving Plans. *American Economic Review*, 91(1):79–98.

Berg, J., Dickhaut, J., and McCabe, K. (1995). Trust, Reciprocity, and Social History. *Games and Economic Behavior*, 10:122–142.

Berg, N. and Gigerenzer, G. (2010). As-if Behavioral Economics: Neoclassical Economics in Disguise. *History of Economic Ideas*, 18(1):133–166.

Betsch, T., Haberstroh, S., Glöckner, A., Haar, T., and Fiedler, K. (2001). The effects of routine strength on adaptation and information search in recurrent decision making. *Organizational Behavior and Human Decision Processes*, 84(1):23–53.

Binmore, K. (1992). *Fun and Games*. Heath, Lexington,MA.

Binmore, K. (2007). *Does Game Theory Work? The Bargaining Challenge*. MIT Press, Cambridge.

Binmore, K. and Shaked, A. (2010). Experimental Economics: Where Next? *Journal of Economic Behavior & Organization*, 73(1):87–100.

Bolton, G. E. (1991). A Comparative Model of Bargaining: Theory and Evidence. *The American Economic Review*, 81(5):1096–1136.

Bolton, G. E. and Ockenfels, A. (2000). ERC: A Theory of Equity, Reciprocity, and Competition. *American Economic Review*, 90(1):166–193.

Bolton, G. E. and Zwick, R. (1995). Anonymity versus Punishment in Ultimatum Bargaining. *Games and Economic Behavior*, 10:95–121.

Brandstätter, E., Gigerenzer, G., and Hertwig, R. (2006). The Priority Heuristic: Making Choices Without Trade-Offs. *Psychological Review*, 113(2):409–432.

Camerer, C. and Ho, T.-H. (1999). Experience-Weighted Attraction Learning in Normal Form Games. *Econometrica*, 67(4):827–874.

Camerer, C. F. (2003). *Behavioral Game Theory: Experiments in Strategic Interaction.* Princeton University Press, Princeton.

Charness, G. and Rabin, M. (2002). Understanding Social Preferences with Simple Tests. *The Quarterly Journal of Economics*, 117(3):817–869.

Cheung, Y.-W. and Friedman, D. (1997). Individual Learning in Normal Form Games: Some Laboratory Results. *Journal of Economic Literature*, 19:46–76.

Clemen, R. T. (1989). Combining Forecasts : A Review and Annotated Bibliography. *International Journal of Forecasting*, 5:559–583.

Collel, M., Andreu, M. W., and Green, J. (1995). *Microeconomic Theory.* Oxford: Oxford University Press.

Cooper, D. J., Garvin, S., and Kagel, J. H. (1997). Signalling and Adaptive Learning in an Entry Limit Pricing Game. *The RAND Journal of Economics*, 28(4):662–683.

Crook, P. A. and Hayes, G. (2003). Learning in a State of Confusion: Perceptual Aliasing in Grid World Navigation. In *Proceedings of Towards Intelligent Mobile Robots (TIMR 2003), 4th British Conference on (Mobile) Robotics.*

De Bruyn, A. and Bolton, G. E. (2008). Estimating the Influence of Fairness on Bargaining Behavior. *Management Science*, 54(10):1774–1791.

De Croon, G., Van Dartel, M. F., and Postma, E. O. (2005). Evolutionary Learning Outperforms Reinforcement Learning on Non-Markovian Tasks. In *Proceedings of the 8th European Conference on Artificial Life, Workshop on Memory and Learning Mechanisms in Autonomous Robots*, Cantebury.

Dieckmann, A., Dippold, K., and Dietrich, H. (2009). Compensatory versus Noncompensatory Models for Predicting Consumer Preferences. *Judgment and Decision Making*, 4(3):200–213.

Erev, I., Bereby-Meyer, Y., and Roth, A. E. (1999). The Effect of Adding a Constant to All Payoffs: Experimental Investigation, and Implications for Reinforcement Learning Models. *Journal of Economic Behavior & Organization*, 39(1):111–128.

Erev, I., Ert, E., and Roth, A. E. (2010). A Choice Prediction Competition for Market Entry Games: An Introduction. *Games*, 1(2):117–136.

Erev, I. and Roth, A. E. (1998). Predicting How People Play Games: Reinforcement Learning in Experimental Games with Unique, Mixed Strategy Equilibria. *American Economic Review*, 88(4):848–881.

Ert, E. and Erev, I. (2007). Replicated Alternatives and the Role of Confusion, Chasing, and Regret in Decisions from Experience. *Journal of Behavioral Decision Making*, 20:305–322.

Ert, E., Erev, I., and Roth, A. E. (2011). A Choice Prediction Competition for Social Preferences in Simple Extensive Form Games: An Introduction. *Games*, 2(3):257–276.

Falk, A. and Fischbacher, U. (2006). A Theory of Reciprocity. *Games and Economic Behavior*, 54(2):293–315.

Fehr, E., Kirchler, E., Weichbold, A., and Gächter, S. (1998). When Social Norms Overpower Competition: Gift Exchange in Experimental Labor Markets. *Journal of Labor Economics*, 16(2):324–351.

Fehr, E. and Schmidt, K. M. (1999). A Theory of Fairness, Competition, and Cooperation. *The Quarterly Journal of Economics*, 114(3):817–868.

Forsythe, R., Horowitz, J. L., Savin, N. E., and Sefton, M. (1994). Fairness in Simple Bargaining Experiments. *Games and Economic Behavior*, 6:347–369.

Fudenberg, D. and Levine, D. K. (1998). *The Theory of Learning in Games.* The MIT Press, Cambridge, MA.

Gale, J., Binmore, K., and Samuelson, L. (1995). Learning to Be Imperfect: The Ultimatum Game. *Games and Economic Behavior*, 8:56–90.

Geisser, S. (1975). The Predictive Sample Reuse Method with Applications. *Journal of the American Statistical Association*, 70(350):320–328.

Gigerenzer, G. (2004). Striking a Blow for Sanity in Theories of Rationality. In Augier, M. and March, J. G., editors, *Models of a Man: Essays in Memory of Herbert A. Simon*, pages 389–409. MIT Press, Cambridge.

Gigerenzer, G. and Goldstein, D. G. (1996). Reasoning the Fast and Frugal Way: Models of Bounded Rationality. *Psychological Review*, 103(4):650–669.

Gigerenzer, G. and Selten, R., editors (2001). *Bounded Rationality: The Adaptive Toolbox.* MIT Press, Cambridge, MA.

Gigerenzer, G., Todd, P. M., and Group, A. R., editors (1999). *Simple Heuristics That Make Us Smart.* Oxford University Press, New York.

Glöckner, A. and Hochman, G. (2011). The Interplay of Experience-Based Affective and Probabilistic Cues in Decision Making: Arousal Increases When Experience and Additional Cues Conflict. *Experimental Psychology*, 58(2):132–141.

Goeree, J. K. and Holt, C. A. (1999). Stochastic Game Theory: For Playing Games, Not Just for Doing Theory. *Proceedings of the National Academy of Sciences of the United States of America*, 96(19):10564–7.

Goldstein, D. G. and Gigerenzer, G. (2002). Models of Ecological Rrationality: The Recognition Heuristic. *Psychological Review*, 109(1):75–90.

Green, R. T. (1956). Surprise as a Factor in the Von Restorff Effect. *Journal of Experimental Psychology*, 52:340–344.

Güth, W., Huck, S., and Müller, W. (2001). The Relevance of Equal Splits in Ultimatum Games. *Games and Economic Behavior*, 37(1):161–169.

Güth, W., Schmidt, C., and Sutter, M. (2003). Fairness in the Mail and Opportunism in the Internet: A Newspaper Experiment on Ultimatum Bargaining. *German Economic Review*, 4(2):243–265.

Güth, W., Schmidt, C., and Sutter, M. (2007). Bargaining Outside the Lab: A Newspaper Experiment of a Three-Person Ultimatum Game. *The Economic Journal*, 117(2002):449–469.

Güth, W., Schmittberger, R., and Schwarze, B. (1982). An Experimental Analysis of Ultimatum Bargaining. *Journal of Economic Behavior & Organization*, 3(4):367–388.

Hansen, B. E. (2008). Least-Squares Forecast Averaging. *Journal of Econometrics*, 146(2):342–350.

Hariskos, W. and Königstein, M. (2015). Asymmetric Cooperation Equilibria in N-player Public Good Games. *Working Paper: Max Planck Institute for Human Development and University of Erfurt*.

Hariskos, W., Leder, J., and Teodorescu, K. (2011). Market Entry Prediction Competition 2010. *Games*, 2(2):200–208.

Hendry, D. F. and Clements, M. P. (2004). Pooling of Forecasts. *The Econometrics Journal*, 5:1–26.

Hertwig, R., Barron, G., Weber, E. U., and Erev, I. (2004). Decisions from experience and the effect of rare events in risky choice. *Psychological Science*, 15(8):534–539.

Hertwig, R., Davis, J. N., and Sulloway, F. J. (2002). Parental Investment: How An Equity Motive Can Produce Inequality. *Psychological Bulletin*, 128(5):728–745.

Hertwig, R., Urs, F., and Bruhin, A. (2012). Simple Heuristics in a Social Game. In Hertwig, R., Hoffrage, U., and ABC Research Group, editors, *Simple Heuristics in a Social World*, volume 2010. Oxford University Press, New York.

Hunt, R. R. and Lamb, C. A. (2001). What Causes the Isolation Effect? *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 27(6):1359–1366.

Kelley, J. W. and Thibaut, H. H. (1978). *Interpersonal Relations: A Theory of InterdependenceTitle*. Wiley, New York.

Larson, S. C. (1931). The Shrinkage of the Coefficient of Multiple Correlation. *Journal of Educational Psychology*, 22(1):45–55.

Ledyard, J. O. (1995). Public Goods: A Survey of Experimental Research. In Kagel, J. H. and Roth, A. E., editors, *The Handbook of Experimental Economics*, pages 111–194. Princeton University Press.

Lee, M. D., Zhang, S., Munro, M., and Steyvers, M. (2011). Psychological Models of Human and Optimal Performance in Bandit Problems. *Cognitive Systems Research*, 12(2):164–174.

Loch, J. and Singh, S. (1998). Using Eligibility Traces to Find the Best Memoryless Policy in Partially Observable Markov Decision Processes. In *Proceedings of the 15th International Conference on Machine Learning*, pages 323–331, Madison.

Loewenstein, G. F., Thompson, L., and Bazerman, M. H. (1989). Social Utility and Decision Making in Interpersonal Contexts. *Journal of Personality and Social Psychology*, 57(3):426–441.

Loewenstein, G. F., Weber, E. U., Hsee, C. K., and Welch, N. (2001). Risk as Feelings. *Psychological Bulletin*, 127(2):267–286.

López-Pérez, R. (2008). Aversion to Norm-Breaking: A Model. *Games and Economic Behavior*, 64(1):237–267.

Luce, R. D. (1959). *Individual Choice Behavior: A Theoretical Analysis*. Wiley, New York.

Marks, G. and Miller, N. (1987). Ten Years of Research on the False-Consensus Effect: An Empirical and Theoretical Review. *Psychological Bulletin*, 102(1):72–90.

McKelvey, R. D. and Palfrey, T. R. (1995). Quantal Response Equilibria for Normal Form Games. *Games and Economic Behavior*, 10(1):6–38.

McKelvey, R. D. and Palfrey, T. R. (1998). Quantal Response Equilibria for Extensive Form Games. *Humanities*, 1:9–41.

McKelvey, R. D., Palfrey, T. R., and Weber, R. A. (2000). The Effects of Payoff Magnitude and Heterogeneity on Behavior in 2 x 2 Games with Unique Mixed Strategy Equilibria. *Journal of Economic Behavior & Organization*, 42(4):523–548.

Messick, D. M. and McClintock, C. G. (1968). Motivational Bases of Choice in Experimental Games. *Journal of Experimental Social Psychology*, 4(1):1–25.

Murphy, R. O., Ackermann, K. A., and Handgraaf, M. J. J. (2011). Measuring Social Value Orientation. *Judgment and Decision Making*, 6(8):771–781.

Nagel, R. (1995). Unraveling in Guessing Games : An Experimental Study. *American Economic Review*, 85(5):1313–1326.

Nash, J. (1951). Non-Cooperative Games. *Annals of Mathematics*, 54(2):286–295.

Nevo, I. and Erev, I. (2012). On Surprise, Change, and the Effect of Recent Outcomes. *Frontiers in Psychology*, 3:1–9.

Payne, J. W., Bettman, J. R., and Johnson, E. J. (1993). *The Adaptive Decision Maker*. Cambridge University Press, Camebridge, UK.

Pillutla, M. M. and Murnighan, K. J. (1996). Unfairness, Anger, and Spite: Emotional Rejections of Ultimatum Offers. *Organizational Behavior and Human Decision Processes*, 68(3):208–224.

Rabin, M. (1993). Incorporating fairness into Game Theory and Economics. *The American Economic Review*, 83(5):1281–1302.

Rogers, B. W., Palfrey, T. R., and Camerer, C. F. (2009). Heterogeneous Quantal Response Equilibrium and Cognitive Hierarchies 1. *Journal of Economic Theory*, 144(4):1440–1467.

Ross, L., Greene, D., and House, P. (1977). The "False in Social Consensus Perception Effect": An Egocentric Bias and Attribution Processes. *Journal of Experimental Social Psychology*, 13(3):279–301.

Roth, A. E. and Erev, I. (1995). Learning in Extensive-Form Games: Experimental Data and Simple Dynamic Models in the Intermediate Term. *Games and Economic Behavior*, 212(8):164–212.

Rottenstreich, Y. and Hsee, C. K. (2001). Money, Kisses, and Electric Shocks: On the Affective Psychology of Risk. *Psychological Science*, 12(3):185–90.

Selten, R. (1965). Spieltheoretische Behandlung eines Oligopolmodells mit Nachfrageträgheit. *Zeitschrift für die gesamte Staatswissenschaft / Journal of Institutional and Theoretical Economics*, 121(2):301–324 and 667–689.

Selten, R. (1973). A Simple Model of Imperfect Competition, Where 4 Are Few and 6 Are Many. *International Journal of Game Theory*, 2(1):141–201.

Selten, R. (1975). Reexamination of the Perfectness Concept for Equilibrium Points in Extensive Games. *International Journal of Game Theory*, 4(1):25–55.

Simon, H. A. (1956). Rational Choice and the Structure of the Environment. *Psychological Review*, 63(2):129–38.

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

Stock, J. H. and Watson, M. W. (2004). Combination Forecasts of Output Growth in a Seven-Country Data Set. *Journal of Forecasting*, 23(6):405–430.

Stone, M. (1974). Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of Royal Statistical Society*, 36(2):111–147.

Tulving, E. (1969). Retrograde Amnesia in Free Recall. *Science*, 164(3875):88–90.

Tversky, A. (1972). Choice by Elimination. *Journal of Mathematical Psychology*, 9:341–367.

Van Lange, P. A. M. (1999). The Pursuit of Joint Outcomes and Equality in Outcomes: An Integrative Model of Social Value Orientation. *Journal of Personality and Social Psychology*, 77(2):337–349.

Van Lange, P. A. M., De Cremer, D., Van Dijk, E., and Van Vugt, M. (2007). Self-Interest and Beyond: Basic Principles of Social Interaction. In Kruglanski, A. W. and Higgings, E. T., editors, *Social Psychology: Handbook of Basic Principles*, pages 540–561. Guilford, New York.

Von Restorff, H. (1933a). Analyse von Vorgängen im Spurenfeld: Über die Wirkung von Bereichsbildungen im Spurenfeld. *Psychological Research*, 61(1933):299–342.

Von Restorff, H. (1933b). Über die Wirkung von Bereichsbildungen im Spurenfeld. *Psychologische Forschung*, 18:299–342.

Appendix A

1 hariskos.m

The MATLAB program hariskos.m computes the MSD of the estimation set for the Hariskos et al. model (see Section 2.5.2).

```
%%%PART 1: INPUT %%%
load('ent_est_par.dat'); data = ent_est_par;
nt = 40; %%% NUMBER OF PROBLEMS %%%
x = data(1:nt,[1 2 3 4 5 6]); %%% PROBLEM K PH H L S
%%% PART 2: PREDICTIONS %%%
his=zeros(4,2,50); tot=zeros(2); gm=zeros(4,2);
dec=zeros(4,50); lsurp=zeros(4); asurp=zeros(4);
ipie=zeros(4); ieps=zeros(4); iro=zeros(4);
iwgm=zeros(4); icas=zeros(4); ilamda=zeros(4);
sent=zeros(40,2); %mean entry rates in the simulation;
spay=zeros(40,2); %mean efficiencies in the simulation;
salt=zeros(40,2); %mean alteration rates in the simulation;
%%% PARAMETERS %%%
initial=0.66; ro=0.2; wgm=0.8; kappa=3; eps=0.24; pie=0.6;
lamda=0.5; %%% NEW PARAMETER
nsim=10000; %%% NUMBER OF SIMULATIONS %%%
for ss=1:nsim %%% START SIMULATION %%%
for prob=1:nt %%% START PROBLEM %%%
k=x(prob,2); block=1;
for player=1:4 %%%PLAYER TRAITS %%%
ipie(player)=rand*pie; ieps(player)=rand*eps; iro(player)=rand*ro;
iwgm(player)=rand*wgm; icas(player)=round(.5+rand*kappa);
%%% NEW TRAIT: CAPTURES SENSITIVITY TO THE MOST RECENT & POSITIVE ... %%%
%%% FORGONE PAYOFF IF IT IS MUCH LARGER THAN THE OBTAINED PAYOFF %%%
ilamda(player)=rand*lamda;
end
for t=1:50; %%% TRIALS%%%
for player=1:4; %%% PLAYER DECISIONS %%%
explor=0; inertia=0; exploit=0;
if t==1 %%% STAGE 1: EXPLORATION %%%
explor=1;
if rand<initial; dec(player,t)=2;
else dec(player,t)=1;
end
end
```

```
if t>1 && rand<ieps(player); explor=1;
if rand<initial; dec(player,t)=2;
else dec(player,t)=1;
end
end %%% END EXPLORATION %%%
if explor==0 %%% STAGE 2: INERTIA %%%
rsurp=lsurp(player)/(asurp(player)+lsurp(player));
if rand<ipie(player)^rsurp; inertia=1; dec(player,t)=dec(player,t-1);
end
end %%% END INERTIA %%%
if inertia==0 && explor==0 %%% STAGE 3: EXPLOITATION %%%
exploit=1; tot(1)=0; tot(2)=0;
ncas=icas(player); %%% NUMBER OF CASES = INDIVIDUAL CASES %%%
for i=1:ncas %%% SAMPLING %%%
if rand<iro(player); Case=t-1;
else Case=round(0.5+(t-1)*rand);
end
%%% ***REFINEMENT NUMBER ONE*** %%%
%%% REVISION OF EARLY CASES %%%
%%% UNRELIABLE CASES FROM THE FIRST TRIALS ARE MORE
%%% LIKELY TO BE EXCLUDED FROM THE SAMPLE %%%
if Case<9
if rand<iro(player); Case=t-1;
else Case=round(0.5+(t-1)*rand);
end
end
if Case<7
if rand<iro(player); Case=t-1;
else Case=round(0.5+(t-1)*rand);
end
end
if Case<5
if rand<iro(player); Case=t-1;
else Case=round(0.5+(t-1)*rand);
end
end
if Case<3
if rand<iro(player); Case=t-1;
else Case=round(0.5+(t-1)*rand);
end
end
%%% ***END OF REFINEMENT NUMBER ONE*** %%%
draw=his(player,1,Case); %%% WEIGHTING %%%
smem=(1-iwgm(player))*draw+iwgm(player)*gm(player,1); tot(1)=tot(1)+smem;
draw=his(player,2,Case);
smem=(1-iwgm(player))*draw+iwgm(player)*gm(player,2); tot(2)=tot(2)+smem;
end
if tot(1)>tot(2) %%% WEIGHTED & SAMPLE-BASED CHOICE %%%
dec(player,t)=1;
```

```
else
dec(player,t)=2; %%%minor modification; if tie, then enter (no coin toss)
end
%%% ***REFINEMENT NUMBER TWO*** %%%
%%% REVISION OF WEIGHTED & SAMPLE-BASED CHOICE
%%% THE PLAYERS CONSIDER ONLY POSITIVE(!) & MORE RECENT(!) FORGONE PAYOFFS
%%%
%%% SCENARIO 1A: NEGATIVE(!) AND MUCH SMALLER OBTAINED PAYOFF
%%% FROM STAYING OUT(!) IN THE LAST TRIAL
%%% DO I REALLY WANT TO STAY OUT AGAIN?
if his(player,1,t-1) < 0 && his(player,2,t-1) > 0
if 3*his(player,1,t-1) > -his(player,2,t-1) && dec(player,t)==1
if rand<ilamda(player)
dec(player,t)=2;
end
end
end
%%% SCENARIO 2A: NEGATIVE(!) AND MUCH SMALLER OBTAINED PAYOFF
%%% FROM ENTERING(!) IN THE LAST TRIAL
%%% DO I REALLY WANT TO ENTER AGAIN?
if his(player,1,t-1) > 0 && his(player,2,t-1) < 0
if 3*his(player,2,t-1) > -his(player,1,t-1) && dec(player,t)==2
if rand<ilamda(player)
dec(player,t)=1;
end
end
end
%%% SCENARIO 1B: POSITIVE(!) BUT MUCH SMALLER OBTAINED PAYOFF
%%% FROM STAYING OUT(!) IN THE LAST TRIAL
%%% DO I REALLY WANT TO STAY OUT AGAIN?
if his(player,1,t-1) > 0 && his(player,2,t-1) > 0 && ...
   his(player,1,t-1)<his(player,2,t-1)
if 3*his(player,1,t-1) < his(player,2,t-1) && dec(player,t)==1
if rand<ilamda(player)
dec(player,t)=2;
end
end
end
%%% SCENARIO 2B: POSITIVE(!) BUT MUCH SMALLER OBTAINED PAYOFF
%%% FROM ENTERING(!) IN THE LAST TRIAL
%%% DO I REALLY WANT TO ENTER AGAIN?
if his(player,1,t-1) > 0 && his(player,2,t-1) > 0 && ...
   his(player,1,t-1)>his(player,2,t-1)
if 3*his(player,2,t-1) < his(player,1,t-1) && dec(player,t)==2
if rand<ilamda(player)
dec(player,t)=1;
end
end
end %%% ***END REFINEMENT NUMBER TWO*** %%%
```

```
end %%% END EXPLOITATION %%%
end %%% END PLAYER DECISIONS %%%
%%% STATISTICS %%%
ent=sum(dec(1:4,t))-4;
sent(prob,block)=sent(prob,block) + ent/(4*nsim*25);
if ent>0; spay(prob,block)=spay(prob,block) + ent*(10-k*ent)/(4*nsim*25);
end
if t>1
for player=1:4; altt=0;
if dec(player,t) ~= dec(player,t-1); altt=1;
end
if block==1; salt(prob,block)=salt(prob,block) + altt/(4*nsim*24);
else salt(prob,block)=salt(prob,block) + altt/(4*nsim*25);
end
end
end
%%% PAYOFFS %%%
ph=x(prob, 3); h=x(prob, 4); l=x(prob, 5); sf=x(prob, 6); state=h;
if rand>ph; state=l;
end
sft=sf;
if rand<0.5; sft=-sf;
end
%%% PLAYERS' SURPRIZE AND GRAND MEAN %%%
for player=1:4; his(player,1,t)=round(state/sft);
if dec(player,t)==1; his(player,2,t)=state+10-k*(ent+1);
else his(player,2,t)=state+10-k*(ent);
end
%%% SURPRIZE %%%
if t==1
lsurp(player)=(abs(his(player,2,t)-gm(player,2)) + ...
               abs(his(player,1,t)-gm(player,1)))/2; asurp(player)=0.00001;
end
if t>1
lsurp(player)= (abs(his(player,2,t)-gm(player,2)) + ...
                abs(his(player,1,t)-gm(player,1)) + ...
                abs(his(player,1,t)-his(player,1,t-1)) + ...
                abs(his(player,2,t)-his(player,2,t-1)))/4;
asurp(player)=asurp(player)*(1-1/50)+lsurp(player)*(1/50);
end
%%% UPDATING GRAND MEAN %%%
gm(player,1)=(1-1/t)*gm(player,1)+(1/t)*his(player,1,t);
gm(player,2)=(1-1/t)*gm(player,2)+(1/t)*his(player,2,t);
end %%% END PLAYERS' SURPRIZE AND GRAND MEAN %%%
if t/25==round(t/25)
block=block+1;
end
end %%%END TRIALS %%%
end %%% END PROBLEM %%%
```

```
end %%% END SIMULATION %%%
%%% PART 3: THE OUTPUT %%%
load('ent_est_results.dat'); data1 = ent_est_results;
% prob, k, ph, h, l, sf, ent1, ent2, pay1, pay2, alt1, alt2;
res = data1(1:nt,[1 2 3 4 5 6 7 8 9 10 11 12]);
x1=data1(1:nt, [1 2 3 4 5 6]);
ent1=data1(:,7); ent2=data1(:,8);
pay1=data1(:,9); pay2=data1(:,10);
alt1=data1(:,11); alt2=data1(:,12);
gent1=ent1-sent(:,1); gent2=ent2-sent(:,2);
gpay1=pay1-spay(:,1); gpay2=pay2-spay(:,2);
galt1=alt1-salt(:,1); galt2=alt2-salt(:,2);
msdent1=(gent1.^2)/0.0016490; msdent2=(gent2.^2)/0.0014957;
msdpay1=(gpay1.^2)/0.1371033; msdpay2=(gpay2.^2)/0.1188298;
msdalt1=(galt1.^2)/0.0011904; msdalt2=(galt2.^2)/0.0014519;
msd=(msdent1 +msdent2+msdpay1+msdpay2 + msdalt1+msdalt2)/6;
%%% Output
SimulatedData=horzcat(x1, sent, spay, salt, msd)
AverageMSD=mean(msd)
```

2 leder.m

The MATLAB program leder.m computes the MSD of the estimation set for the Leder et al. model (see Section 2.5.3).

```
%%% PART 1: INPUT %%%
load('ent_est_par.dat'); data = ent_est_par;
nt = 40; %%% NUMBER OF PROBLEMS %%%
x = data(1:nt,[1 2 3 4 5 6]); %%% PROBLEM K PH H L S
%%% PART 2: PREDICTION %%%
his=zeros(4,2,50); tot=zeros(2); gm=zeros(4,2);
dec=zeros(4,50); lsurp=zeros(4); asurp=zeros(4);
ipie=zeros(4); ieps=zeros(4); iro=zeros(4);
iwgm=zeros(4); icas=zeros(4); ilamda=zeros(4);
sent=zeros(40,2); %mean entry rates in the simulation;
spay=zeros(40,2); %mean efficiencies in the simulation;
salt=zeros(40,2); %mean alteration rates in the simulation;
%%% PARAMETERS %%%
initial=0.66; ro=0.2; wgm=0.8; kappa=3; eps=0.24; pie=0.6;
lamda=0.5; %%% NEW PARAMETER
nsim=10000; %%% NUMBER OF SIMULATIONS %%%
for ss=1:nsim %%% START SIMULATION %%%
for prob=1:nt  %%% START PROBLEM %%%
k=x(prob,2); block=1;
for player=1:4 %%%PLAYER TRAITS %%%
ipie(player)=rand*pie; ieps(player)=rand*eps; iro(player)=rand*ro;
iwgm(player)=rand*wgm; icas(player)=round(.5+rand*kappa);
```

```
%%% NEW TRAIT: CAPTURES SENSITIVITY TO THE MOST RECENT & POSITIVE ... %%%
%%% FORGONE PAYOFF IF IT IS MUCH LARGER THAN THE OBTAINED PAYOFF %%%
ilamda(player)=rand*lamda;
end
for t=1:50; %%% TRIALS%%%
for player=1:4; %%% PLAYER DECISIONS %%%
explor=0; inertia=0; exploit=0;
if t==1 %%% STAGE 1: EXPLORATION %%%
explor=1;
if rand<initial; dec(player,t)=2;
else dec(player,t)=1;
end
end
%%% ***REFINEMENT NUMBER ONE*** %%%
if t>1 && t<10
pexp=9/t*ieps(player);
end
if t>9 && t<31
pexp=0.95*ieps(player);
end
if t>30
pexp=0.9*ieps(player);
end
%%% ***END OF REFINEMENT NUMBER ONE*** %%%
if t>1 && rand<pexp; explor=1;
if rand<initial; dec(player,t)=2;
else dec(player,t)=1;
end
end %%% END EXPLORATION %%%
if explor==0 %%% STAGE 2: INERTIA %%%
rsurp=lsurp(player)/(asurp(player)+lsurp(player));
if rand<ipie(player)^rsurp; inertia=1; dec(player,t)=dec(player,t-1);
end
end %%% END INERTIA %%%
if inertia==0 && explor==0 %%% STAGE 3: EXPLOITATION %%%
exploit=1; tot(1)=0; tot(2)=0;
ncas=icas(player); %%% NUMBER OF CASES = INDIVIDUAL CASES %%%
for i=1:ncas %%% SAMPLING %%%
if rand<iro(player); Case=t-1;
else Case=round(0.5+(t-1)*rand);
end
%%% ***REFINEMENT NUMBER TWO*** %%%
%%% REVISION OF EARLY CASES %%%
%%% UNRELIABLE CASES FROM THE FIRST TRIALS ARE MORE
%%% LIKELY TO BE EXCLUDED FROM THE SAMPLE %%%
if Case<9
if rand<iro(player); Case=t-1;
else Case=round(0.5+(t-1)*rand);
end
```

```
end
if Case<7
if rand<iro(player); Case=t-1;
else Case=round(0.5+(t-1)*rand);
end
end
if Case<5
if rand<iro(player); Case=t-1;
else Case=round(0.5+(t-1)*rand);
end
end
if Case<3
if rand<iro(player); Case=t-1;
else Case=round(0.5+(t-1)*rand);
end
end
%%% ***END OF REFINEMENT NUMBER TWO*** %%%
draw=his(player,1,Case); %%% WEIGHTING %%%
smem=(1-iwgm(player))*draw+iwgm(player)*gm(player,1); tot(1)=tot(1)+smem;
draw=his(player,2,Case);
smem=(1-iwgm(player))*draw+iwgm(player)*gm(player,2); tot(2)=tot(2)+smem;
end
if tot(1)>tot(2) %%% WEIGHTED & SAMPLE-BASED CHOICE %%%
dec(player,t)=1;
else
dec(player,t)=2; %%%minor modification; if tie, then enter (no coin toss)
end
%%% ***REFINEMENT NUMBER THREE*** %%%
%%% REVISION OF WEIGHTED & SAMPLE-BASED CHOICE
%%% THE PLAYERS CONSIDER ONLY POSITIVE(!) & MORE RECENT(!) FORGONE PAYOFFS
%%%
%%% SCENARIO 1A: NEGATIVE(!) AND MUCH SMALLER OBTAINED PAYOFF
%%% FROM STAYING OUT(!) IN THE LAST TRIAL
%%% DO I REALLY WANT TO STAY OUT AGAIN?
if his(player,1,t-1) < 0 && his(player,2,t-1) > 0
if 3*his(player,1,t-1) > -his(player,2,t-1) && dec(player,t)==1
if rand<ilamda(player)
dec(player,t)=2;
end
end
end
%%% SCENARIO 2A: NEGATIVE(!) AND MUCH SMALLER OBTAINED PAYOFF
%%% FROM ENTERING(!) IN THE LAST TRIAL
%%% DO I REALLY WANT TO ENTER AGAIN?
if his(player,1,t-1) > 0 && his(player,2,t-1) < 0
if 3*his(player,2,t-1) > -his(player,1,t-1) && dec(player,t)==2
if rand<ilamda(player)
dec(player,t)=1;
end
```

```
end
end
%%% SCENARIO 1B: POSITIVE(!) BUT MUCH SMALLER OBTAINED PAYOFF
%%% FROM STAYING OUT(!) IN THE LAST TRIAL
%%% DO I REALLY WANT TO STAY OUT AGAIN?
if his(player,1,t-1) > 0 && his(player,2,t-1) > 0 &&...
   his(player,1,t-1)<his(player,2,t-1)
if 3*his(player,1,t-1) < his(player,2,t-1) && dec(player,t)==1
if rand<ilamda(player)
dec(player,t)=2;
end
end
end
%%% SCENARIO 2B: POSITIVE(!) BUT MUCH SMALLER OBTAINED PAYOFF
%%% FROM ENTERING(!) IN THE LAST TRIAL
%%% DO I REALLY WANT TO ENTER AGAIN?
if his(player,1,t-1) > 0 && his(player,2,t-1) > 0 && ...
   his(player,1,t-1)>his(player,2,t-1)
if 3*his(player,2,t-1) < his(player,1,t-1) && dec(player,t)==2
if rand<ilamda(player)
dec(player,t)=1;
end
end
end %%% ***END REFINEMENT NUMBER THREE*** %%%
end %%% END EXPLOITATION %%%
end %%% END PLAYER DECISIONS %%%
%%% STATISTICS %%%
ent=sum(dec(1:4,t))-4;
sent(prob,block)=sent(prob,block) + ent/(4*nsim*25);
if ent>0; spay(prob,block)=spay(prob,block) + ent*(10-k*ent)/(4*nsim*25);
end
if t>1
for player=1:4; altt=0;
if dec(player,t) ~= dec(player,t-1); altt=1;
end
if block==1; salt(prob,block)=salt(prob,block) + altt/(4*nsim*24);
else salt(prob,block)=salt(prob,block) + altt/(4*nsim*25);
end
end
end
%%% PAYOFFS %%%
ph=x(prob, 3); h=x(prob, 4); l=x(prob, 5); sf=x(prob, 6); state=h;
if rand>ph; state=l;
end
sft=sf;
if rand<0.5; sft=-sf;
end
%%% PLAYERS' SURPRIZE AND GRAND MEAN %%%
for player=1:4; his(player,1,t)=round(state/sft);
```

```
if dec(player,t)==1; his(player,2,t)=state+10-k*(ent+1);
else his(player,2,t)=state+10-k*(ent);
end
%%% SURPRIZE %%%
if t==1
lsurp(player)=(abs(his(player,2,t)-gm(player,2)) + ...
               abs(his(player,1,t)-gm(player,1)))/2; asurp(player)=0.00001;
end
if t>1
lsurp(player)= (abs(his(player,2,t)-gm(player,2)) + ...
                abs(his(player,1,t)-gm(player,1)) + ...
                abs(his(player,1,t)-his(player,1,t-1)) + ...
                abs(his(player,2,t)-his(player,2,t-1)))/4;
asurp(player)=asurp(player)*(1-1/50)+lsurp(player)*(1/50);
end
%%% UPDATING GRAND MEAN %%%
gm(player,1)=(1-1/t)*gm(player,1)+(1/t)*his(player,1,t);
gm(player,2)=(1-1/t)*gm(player,2)+(1/t)*his(player,2,t);
end %%% END PLAYERS' SURPRIZE AND GRAND MEAN %%%
if t/25==round(t/25)
block=block+1;
end
end %%%END TRIALS %%%
end %%% END PROBLEM %%%
end %%% END SIMULATION %%%
%%% PART 3: THE OUTPUT %%%
load('ent_est_results.dat'); data1 = ent_est_results;
% prob, k, ph, h, l, sf, ent1, ent2, pay1, pay2, alt1, alt2;
res = data1(1:nt,[1 2 3 4 5 6 7 8 9 10 11 12]);
x1=data1(1:nt, [1 2 3 4 5 6]);
ent1=data1(:,7); ent2=data1(:,8);
pay1=data1(:,9); pay2=data1(:,10);
alt1=data1(:,11); alt2=data1(:,12);
gent1=ent1-sent(:,1); gent2=ent2-sent(:,2);
gpay1=pay1-spay(:,1); gpay2=pay2-spay(:,2);
galt1=alt1-salt(:,1); galt2=alt2-salt(:,2);
msdent1=(gent1.^2)/0.0016490; msdent2=(gent2.^2)/0.0014957;
msdpay1=(gpay1.^2)/0.1371033; msdpay2=(gpay2.^2)/0.1188298;
msdalt1=(galt1.^2)/0.0011904; msdalt2=(galt2.^2)/0.0014519;
msd=(msdent1 +msdent2+msdpay1+msdpay2 + msdalt1+msdalt2)/6;
%%% Output
SimulatedData=horzcat(x1, sent, spay, salt, msd)
AverageMSD=mean(msd)
```

3 teodorescu.sas

The SAS program teodorescu.sas computes the MSD of the estimation set for the Teodorescu et al. model (see Section 2.5.1).

```
*************************Part 1: Input***************************************;
**** This part should appear without any change in all the SAS submissions;
options linesize=78 pagesize=999;
libname out '';
data a; infile 'c:\ent_est_par.dat';
input problem k ph h l sf;
***************************Part 2: The derivation of the prediction**********;
************simulation of the I-SAW model**************;
********The parameters*******************;
data a; set a;
initial=.66;
do ro=.1;
 do wgm= .4;
  do kapa=1.5;
   do eps=.12;
 do pie=.3;
output;
end; end; end; end; end;
data a; set a;
nsim=2000;
array sampsurp{4} sampsurp1-sampsurp4;
array his{4,2,50} his1-his400;
array tot{2} tot1-tot2;
array gm{4,2} gm1-gm8;
array dec{4} dec1-dec4;
array lagdec{4} ldec1-ldec4;
array lsurp{4} lsurp1-lsurp4;
array asurp{4} asurp1-asurp4;
array ipie{4} ipie1-ipie4;
array ieps{4} ieps1-ieps4;
array iro{4} iro1-iro4;
array iwgm{4} iwgm1-iwgm4;
array icas{4} icas1-icas4;
 **Stored statistics;
 ARRAY sent{2} sent1-sent2; *mean entry rates in the simulation;
 ARRAY spay{2} spay1-spay2; *mean efficiencies in the simulation;
 array salt{2} salt1-salt2; *mean alteration rates in the simulation;
***new sim****************************;
do ss=1 to nsim;
 do player=1 to 4;
 ipie{player}=ranuni(0)*2*pie;
 ieps{player}=ranuni(0)*2*eps;
 iro{player}=ranuni(0)*2*ro;
 iwgm{player}=ranuni(0)*2*wgm;
 icas{player}=round(.5+ranuni(0)*2*kapa);
 sampsurp{player}=0;
 do st=1 to 2;  gm{player,st}=0;
   do tt=1 to 50; his{player,st,tt}=.; end;
```

```
 end;**st;
end; *of player;
block=1;
******* the 100 trials ******************************;
do t=1 to 50;
 do player=1 to 4;
*******************decisions *************************;
lagdec{player}=dec{player};
explor=0; inertia=0; exploit=0;
*****exploration******;
if t=1 then pexp=1;
if t>1  and t<6 then pexp=(6*ieps{player})/t;
if t>5  and t<31 then pexp= ieps{player};
if t>30 then pexp=0.9*ieps{player};
 if ranuni(0)<pexp then do;
  explor=1;
  dec{player}=1; if ranuni(0)<initial then dec{player}=2; ** exploration;
 end;
******inertia*****************************;
if explor=0  then do;
 rsurp=lsurp{player}/(asurp{player}+lsurp{player});
 pinertia=ipie{player}**(rsurp);
 if ranuni(0)<pinertia then do; inertia=1; dec{player}=lagdec{player}; end;
end;
if rsurp>0.85 then sampsurp{player}=t;
***exploit*********************************** ;
if inertia=0 and explor=0 then do;
 exploit=1;
 ncas=icas{player};**** sample size***;
do st=1 to 2; tot{st}=0; end;;
 do i=1 to ncas; **sampling**;
    case=round(.5+(t-1)*ranuni(0));
 if ranuni(0)<iro{player} then case=t-1;
 else if sampsurp{player}^=0 then do;
if ranuni(0)<iro{player} then case=sampsurp{player};
 end;
   do st=1 to 2;
     draw=his{player,st,case};
     smem=(1-iwgm{player})*draw+iwgm{player}*gm{player,st};
     tot{st}=sum(tot{st},smem);
   end;
 end;
 dec{player}=1; if tot2>tot1 then dec{player}=2;
 if tot1=tot2 then dec{player}=1+round(ranuni(0));**choice;
end; **of exploit;
end; **of pl;
***statistics***;
  ent=sum(of dec1-dec4)-4;
  sent{block}=sum(0,sent{block},ent/(4*nsim*25));
  if ent>0 then spay{block}=sum(spay{block},ent*(10-k*(ent))/(4*nsim*25));
if t>1 then do player=1 to 4; altt=0;
  if dec{player} ne lagdec{player} then altt=1;
  if block=1 then salt{block} =sum(salt{block},altt/(4*nsim*24));
```

```
  if block=2 then salt{block} =sum(salt{block},altt/(4*nsim*25));
end;
**PAYOFFS***;
state=h; if ranuni(0)>ph then state=l;
sft=sf; if ranuni(0)<.5 then sft=-sf;
do player=1 to 4;
  his{player,1,t}=round(state/sft);
  if dec{player}=1 then his{player,2,t}=state+10-k*(ent+1);
  if dec{player}=2 then his{player,2,t}=state+10-k*(ent);
**surprize;
if t=1 then lsurp{player}=mean(abs(his{player,2,t}-gm{player,2}),
abs(his{player,1,t}-gm{player,1}));
if t>1 then lsurp{player}=mean(abs(his{player,2,t}-gm{player,2}),
abs(his{player,1,t}-gm{player,1}),
abs(his{player,1,t}-his{player,1,t-1}),abs(his{player,2,t}-his{player,2,t-1}));
if t=1 then do; asurp{player}=.00001;  end;
if t>1 then asurp{player}=asurp{player}*(1-1/(50))+lsurp{player}*(1/(50));
***updating grand mean******;
do st=1 to 2;
 gm{player,st}=(1-1/(t))*gm{player,st}+(1/(t))*his{player,st,t};
end;
end; ***of player;
IF (T)/(25)=ROUND((T)/(25)) THEN  do;
    block=block+1;
end;
end; **of trials***;
end; **of sim***;
*********************Part 3: The output ******************************************;
******** This part should appear without any change in all the SAS submissions*******;
data out.model_est; set a;
data c; infile 'c:\ent_est_results.dat';
input problem k    ph     h     l   sf   ent1   ent2    pay1    pay2   alt1   alt2;
data test; merge out.model_est c;
by problem  k ph h l;
data test; set test;
msdent1=((ent1-sent1)**2)/0.0016490;
msdent2=((ent2-sent2)**2)/0.0014957;
msdpay1=((pay1-spay1)**2)/0.1371033;
msdpay2=((pay2-spay2)**2)/0.1188298;
msdalt1=((alt1-salt1)**2)/0.0011904;
msdalt2=((alt2-salt2)**2)/0.0014519;
msd=(msdent1 +msdent2+msdpay1+msdpay2 + msdalt1+msdalt2)/6;
proc sort; by ro wgm kapa eps pie;
proc means noprint;
by ro wgm kapa eps pie;
var msdent1 msdent2 msdpay1 msdpay2 msdalt1 msdalt2 msd;
output out=o mean=msdent1 msdent2 msdpay1 msdpay2 msdalt1 msdalt2 msd;
  proc print;
  var msdent1 msdent2 msdpay1 msdpay2 msdalt1 msdalt2 msd;
by ro wgm kapa eps pie;
proc means data=test noprint;
by ro wgm kapa eps pie;
var msd;
```

```
output out=o mean=msd;
proc sort; by msd;
proc print;
run;
```

4 Experimental Data

Table 1
Estimation Set: Entry Rates, Efficiency and Alternation Rates

Game						Entry Rate		Efficiency		Alternation Rate	
Γ_{est}	k	p	H	L	S	B1	B2	B1	B2	B1	B2
1	2	0.04	70	-3	5	0.71	0.80	2.77	2.66	0.16	0.16
2	2	0.23	30	-9	4	0.55	0.62	2.64	2.75	0.25	0.23
3	2	0.67	1	-2	3	0.88	0.94	2.39	2.24	0.10	0.04
4	2	0.73	30	-80	4	0.71	0.64	2.58	2.57	0.28	0.27
5	2	0.80	20	-80	5	0.66	0.67	2.50	2.67	0.29	0.27
6	2	0.83	4	-20	3	0.73	0.82	2.45	2.50	0.24	0.18
7	2	0.94	6	-90	5	0.86	0.87	2.34	2.38	0.13	0.11
8	2	0.95	1	-20	5	0.86	0.91	2.48	2.31	0.12	0.08
9	2	0.96	4	-90	3	0.87	0.90	2.36	2.34	0.14	0.08
10	3	0.10	70	-8	4	0.42	0.48	1.22	1.11	0.29	0.25
11	3	0.90	9	-80	4	0.80	0.73	-0.33	0.29	0.18	0.25
12	3	0.91	7	-70	6	0.76	0.83	0.10	-0.41	0.19	0.12
13	4	0.06	60	-4	2	0.42	0.41	0.52	0.84	0.22	0.15
14	4	0.20	40	-10	4	0.48	0.46	-0.34	0.04	0.31	0.31
15	4	0.31	20	-9	4	0.49	0.44	-0.07	0.30	0.34	0.38
16	4	0.60	4	-6	2	0.56	0.58	-0.27	-0.26	0.22	0.26
17	4	0.60	40	-60	3	0.58	0.55	-0.96	-0.20	0.28	0.25
18	4	0.73	3	-8	2	0.57	0.55	-0.29	0.09	0.24	0.20
19	4	0.80	20	-80	2	0.64	0.63	-1.30	-1.21	0.28	0.27
20	4	0.90	1	-9	6	0.53	0.48	0.12	0.63	0.21	0.16
21	4	0.96	3	-70	3	0.65	0.62	-0.84	-0.38	0.23	0.18
22	5	0.02	80	-2	3	0.36	0.31	0.24	0.64	0.17	0.17
23	5	0.07	90	-7	3	0.39	0.24	-0.81	0.34	0.19	0.13
24	5	0.53	80	-90	5	0.65	0.58	-3.41	-2.44	0.27	0.36
25	5	0.80	1	-4	2	0.45	0.42	-0.31	0.11	0.20	0.18
26	5	0.88	4	-30	3	0.52	0.49	-0.95	-0.57	0.22	0.21
27	5	0.93	5	-70	4	0.57	0.57	-1.63	-1.43	0.27	0.20
28	6	0.10	90	-10	5	0.26	0.27	-0.13	0.07	0.22	0.19
29	6	0.19	30	-7	3	0.39	0.32	-1.35	-0.45	0.27	0.26
30	6	0.29	50	-20	3	0.47	0.48	-2.74	-2.43	0.38	0.36
31	6	0.46	7	-6	6	0.38	0.34	-0.90	-0.38	0.23	0.24
32	6	0.57	6	-8	4	0.44	0.39	-1.56	-0.59	0.26	0.27
33	6	0.82	20	-90	3	0.63	0.55	-5.33	-3.14	0.26	0.21
34	6	0.88	8	-60	4	0.57	0.50	-3.30	-1.96	0.16	0.19
35	7	0.06	90	-6	4	0.31	0.35	-1.40	-1.43	0.29	0.21
36	7	0.21	30	-8	3	0.39	0.31	-2.20	-1.04	0.30	0.23
37	7	0.50	80	-80	5	0.51	0.55	-4.18	-4.78	0.34	0.32
38	7	0.69	9	-20	5	0.46	0.34	-2.62	-0.88	0.25	0.23
39	7	0.81	7	-30	2	0.41	0.34	-2.25	-0.93	0.22	0.21
40	7	0.91	1	-10	2	0.34	0.27	-0.71	-0.30	0.19	0.17
Mean						0.56	0.56	-0.39	0.04	0.27	0.23
Estimated Error Variance						.0016	.0015	.1370	.1188	.0018	.0015

Source. Erev et al. (2010)

Table 2
Predictions Set: Entry Rates, Efficiency and Alternation Rates

Game						Entry Rate		Efficiency		Alternation Rate	
Γ_{pre}	k	p	H	L	S	B1	B2	B1	B2	B1	B2
1	2	0.04	70	-3	3	0.69	0.78	2.85	2.75	0.17	0.15
2	2	0.18	9	-2	3	0.81	0.82	2.48	2.57	0.19	0.21
3	2	0.20	40	-10	2	0.53	0.50	2.62	2.51	0.29	0.28
4	2	0.33	6	-3	6	0.75	0.81	2.75	2.57	0.16	0.14
5	2	0.40	3	-2	5	0.90	0.95	2.29	2.18	0.13	0.08
6	2	0.95	2	-40	3	0.87	0.93	2.24	2.23	0.11	0.06
7	2	0.97	2	-60	5	0.88	0.93	2.33	2.28	0.11	0.04
8	3	0.03	90	-3	3	0.47	0.53	1.43	1.50	0.28	0.26
9	3	0.10	9	-1	2	0.65	0.70	1.06	0.88	0.17	0.16
10	3	0.33	2	-1	5	0.69	0.69	0.70	0.81	0.26	0.20
11	3	0.36	90	-50	3	0.40	0.37	1.27	1.26	0.28	0.30
12	3	0.47	10	-9	2	0.56	0.57	1.26	1.39	0.30	0.26
13	3	0.50	7	-7	5	0.63	0.61	1.01	1.08	0.26	0.25
14	4	0.07	40	-3	3	0.42	0.45	0.74	0.59	0.28	0.21
15	4	0.44	9	-7	2	0.54	0.54	-0.35	-0.12	0.33	0.36
16	4	0.46	7	-6	5	0.56	0.50	-0.51	0.39	0.27	0.26
17	4	0.47	10	-9	5	0.45	0.51	0.37	0.15	0.35	0.34
18	4	0.53	7	-8	6	0.53	0.54	0.01	-0.14	0.27	0.28
19	4	0.82	9	-40	2	0.71	0.65	-1.98	-1.14	0.25	0.26
20	4	0.86	10	-60	2	0.72	0.69	-2.08	-1.72	0.25	0.23
21	4	0.88	8	-60	4	0.77	0.76	-2.72	-2.69	0.24	0.21
22	5	0.29	5	-2	6	0.40	0.37	-0.10	0.25	0.23	0.26
23	5	0.33	80	-40	5	0.42	0.41	-0.92	-1.16	0.33	0.32
24	5	0.36	90	-50	6	0.46	0.36	-1.30	-0.54	0.34	0.31
25	5	0.42	7	-5	6	0.45	0.46	-0.41	-0.39	0.23	0.22
26	5	0.60	2	-3	2	0.39	0.37	0.10	0.22	0.28	0.21
27	5	0.67	4	-8	3	0.50	0.44	-1.04	-0.41	0.26	0.24
28	5	0.91	8	-80	6	0.57	0.56	-1.75	-1.49	0.20	0.22
29	6	0.08	60	-5	5	0.27	0.30	-0.11	-0.04	0.20	0.18
30	6	0.12	50	-7	6	0.41	0.30	-1.40	-0.21	0.27	0.22
31	6	0.40	60	-40	5	0.50	0.46	-3.03	-2.45	0.35	0.36
32	6	0.56	70	-90	2	0.58	0.57	-4.49	-3.98	0.37	0.30
33	6	0.63	6	-10	5	0.39	0.40	-0.84	-0.87	0.31	0.27
34	7	0.20	80	-20	5	0.39	0.34	-3.14	-1.90	0.24	0.23
35	7	0.30	70	-30	5	0.43	0.48	-2.72	-3.47	0.32	0.32
36	7	0.33	20	-10	6	0.40	0.39	-2.59	-1.92	0.32	0.33
37	7	0.44	5	-4	6	0.36	0.30	-1.03	-0.20	0.23	0.16
38	7	0.50	80	-80	3	0.52	0.49	-4.86	-3.63	0.36	0.36
39	7	0.88	1	-7	5	0.33	0.28	-1.01	-0.14	0.24	0.19
40	7	0.98	2	-80	4	0.34	0.32	-0.88	-0.36	0.24	0.24
Mean						0.54	0.54	-0.34	-0.08	0.26	0.24
Estimated Error Variance						.0016	.0018	.2302	.1733	.0020	.0022

Source. https://sites.google.com/site/gpredcomp/study-results/competition-study

Appendix B

No additional material provided.

Appendix C

5 Three-Person Ultimatum Game: Heuristic Decision Process

Table 3 presents for the proposer decision the decision process of each type.

Table 3
Three-Person UG: Type-Dependent Decision Process for Proposer Decision

(d_P, o_{-R}, r)	$A \vee B \vee C$	$D \vee E \vee F$	G
$(1000, 100, 100)$	0	1,0	1,1
$(800, 300, 100)$	0	1,0	1,0
$(800, 200, 200)$	0	1,0	1,0
$(800, 100, 300)$	0	1,0	1,0
$(600, 500, 100)$	0	1,1	1,0
$(600, 400, 200)$	0	1,0	1,0
$(600, 300, 300)$	0	1,0	1,0
$(600, 200, 400)$	0	1,0	1,0
$(600, 100, 500)$	0	1,0	1,0
$(400, 600, 200)$	0	0	1,0
$(400, 500, 300)$	0	0	1,0
$(400, 400, 400)$	1	0	1,0
$(400, 300, 500)$	0	0	1,0
$(400, 200, 600)$	0	0	1,0
$(200, 600, 400)$	0	0	1,0
$(200, 500, 500)$	0	0	1,0
$(200, 400, 600)$	0	0	1,0
$(000, 600, 600)$	0	0	1,0

The absence of a characteristic is indicated by a value of 0 and its presence by a value of 1. If a type considers more than one characteristic, then their values are separated by a comma.

Table 4 presents for the responder decisions the decision process of each type.

Table 4
Three-Person UG: Decision Process for Responder Decisions

$a \to (d_{-P}, o_R, r)$	D	$A \vee E$	B	$C \vee F \vee G$
$accept \to (1000, 100, 100)$	0,0	0,0	0,0	1
$reject \to (0, 0, 0)$	0,1	0,1	0,1	0
$accept \to (800, 300, 100)$	0,0	0,0	1	1
$reject \to (0, 0, 0)$	0,1	0,1	0	0
$accept \to (800, 200, 200)$	0,0	0,0	0,0	1
$reject \to (0, 0, 0)$	0,1	0,1	0,1	0
$accept \to (800, 100, 300)$	0,0	0,0	0,0	1
$reject \to (0, 0, 0)$	0,1	0,1	0,1	0
$accept \to (600, 500, 100)$	1	1	1	1
$reject \to (0, 0, 0)$	0	0	0	0
$accept \to (600, 400, 200)$	0,0	1	1	1
$reject \to (0, 0, 0)$	0,1	0	0	0
$accept \to (600, 300, 300)$	0,0	0,0	1	1
$reject \to (0, 0, 0)$	0,1	0,1	0	0
$accept \to (600, 200, 400)$	0,0	0,0	0,0	1
$reject \to (0, 0, 0)$	0,1	0,1	0,1	0
$accept \to (600, 100, 500)$	0,0	0,0	0,0	1
$reject \to (0, 0, 0)$	0,1	0,1	0,1	0
$accept \to (400, 600, 200)$	1	1	1	1
$reject \to (0, 0, 0)$	0	0	0	0
$accept \to (400, 500, 300)$	1	1	1	1
$reject \to (0, 0, 0)$	0	0	0	0
$accept \to (400, 400, 400)$	0,0	1	1	1
$reject \to (0, 0, 0)$	0,1	0	0	0
$accept \to (400, 300, 500)$	0,0	0,0	1	1
$reject \to (0, 0, 0)$	0,1	0,1	0	0
$accept \to (400, 200, 600)$	0,0	0,0	0,0	1
$reject \to (0, 0, 0)$	0,1	0,1	0,1	0
$accept \to (200, 600, 400)$	1	1	1	1
$reject \to (0, 0, 0)$	0	0	0	0
$accept \to (200, 500, 500)$	1	1	1	1
$reject \to (0, 0, 0)$	0	0	0	0
$accept \to (200, 400, 600)$	0,0	1	1	1
$reject \to (0, 0, 0)$	0,1	0	0	0
$accept \to (0, 600, 600)$	1	1	1	1
$reject \to (0, 0, 0)$	0	0	0	0

The absence of a characteristic is indicated by a value of 0 and its presence by a value of 1. If a type considers more than one characteristic, then their values are separated by a comma.

6 Within-Sample Predictions: Ultimatum Game

```
c=1000; %monetary cake
x=[900 800 700 600 500 400 300 200 100;   % possible demands
   100 200 300 400 500 600 700 800 900 ]; % possible offers
off=[113  39  94 232  518  28   5   0   6]; % actual offers
acc=[361 422 590 820 1015 976 958 939 938]; % actual acceptances
p_obs=[off/sum(off); acc/sum(off)];
```

6.1 Heuristic Mix

```
%joint behavior that can be explained by ...
n=[ 114   %fairness + mirror (o_i=500,m_i=500)
    111   %fairness + anger  (o_i=500,m_i=400)
    134   %fairness + money  (o_i=500,m_i=100)
     84   %fear     + mirror (o_i=400,m_i=400)
     52   %fear     + anger  (o_i=400,m_i=300)
     55   %fear     + money  (o_i=400,m_i=100)
     86]; %money             (o_i=100,m_i=100)
%distribution over proposer criteria
fairness=sum(n(1:3))/sum(n);
fear=sum(n(4:6))/sum(n);
money=n(7)/sum(n);
%chi-square test for equality of responder criteria distribution
od=[n(1)+n(4) n(2)+n(5) n(3)+n(6)]; %observed distribution
ed=[1/3       1/3       1/3      ]; %qual distribution
X2=sum((od-ed).^2./ed);             %chi-square
df=length(od)-1;                    %degrees of freedom
pv=1-chi2cdf(sum(X2),df);           %p-value
%predicted offer probabilities
p_fft(1,:)=[money 0 0 fear fairness 0 0 0 0];
%predicted acceptance probabilities
p_fft(2,:)=[money+1/3*fear+1/3*fairness;
            money+1/3*fear+1/3*fairness;
            money+2/3*fear+1/3*fairness;
            money+fear+2/3*fairness;
            money+fear+fairness;
            money+fear+fairness;
            money+fear+fairness;
            money+fear+fairness;
            money+fear+fairness]';
```

6.2 Subgame Perfect Equilibrium

```
uRA=x(2,:); %acceptance utilities
p_spe(2,:)=uRA>0; %predicted acceptance probabilities
EuP=x(1,:).*p_spe(2,:); %expected utilities
```

```
p_spe(1,:)=(EuP==max(EuP(1,:))); %predicted offer probabilities
```

6.3 Fehr-Schmidt Subgame Perfect Equilibrium

```
%distribution over preferences
dp=[0.3 0.0 0.00;  %type A
    0.3 0.5 0.25;  %type B
    0.3 1.0 0.60;  %type C
    0.1 4.0 0.60]; %type D
uRA=repmat(x(2,:),4,1)... %acceptance utilities
    -repmat(dp(:,2),1,9).*repmat(max(x(1,:)-x(2,:),0),4,1)...
    -repmat(dp(:,3),1,9).*repmat(max(x(2,:)-x(1,:),0),4,1);
p_fsspe(2,:)=sum(repmat(dp(:,1),1,9).*(uRA>0)); %pred. acceptance prob.
EuP=(repmat(x(1,:),4,1)... %expected offer utilities
   -repmat(dp(:,2),1,9).*repmat(max(x(2,:)-x(1,:),0),4,1)...
   -repmat(dp(:,3),1,9).*repmat(max(x(1,:)-x(2,:),0),4,1))...
   .*repmat(p_fsspe(2,:),4,1);
maxEuP=[repmat(max(EuP(1,:)),1,9); %maximum expected utilities of each type
        repmat(max(EuP(2,:)),1,9);
        repmat(max(EuP(3,:)),1,9);
        repmat(max(EuP(4,:)),1,9)];
p_fsspe(1,:)=sum((EuP==maxEuP).*repmat(dp(:,1),1,9)); %pred. offer prob.
```

6.4 Fehr-Schmidt Quantal Response Equilibrium

```
start = [.1;.01;.01]; %start values for alpha = beta, lambda_P, lambda_R
%maximize negative logl by minimizing positive logl (beta <=alpha)
[parameter, logl, exitflag, output, gradient, hessian] = fminunc...
(@fs_logl,start,optimset('Display','off', 'LargeScale','off'),x,off,acc);
%save estimation results
FS_LOGL=logl; % log likelilood
FS_PE=parameter'; %estimates
inv_hessian=inv(hessian);
FS_SD=sqrt(inv_hessian([1 5 9])); %standard deviations
%label estimates
alpha=parameter(1); beta=parameter(1);
lambdaP=parameter(2); lambdaR=parameter(3);
for k=1:length(x) %responder
    utility_accept(k)=x(2,k)... % acceptance utilities
                      -alpha*max(x(1,k)-x(2,k),0)...
                      -beta*max(x(2,k)-x(1,k),0);
    nominator(k)=exp(lambdaR*utility_accept(k));
    denominator(k)=1+exp(lambdaR*utility_accept(k));
    FSQRE_AP(k)=nominator(k)/denominator(k); % acceptance probabilities
end;
for k=1:length(x) %proposer
    utility_offer(k)=x(1,k)... % offer utilities
```

```
                    -alpha*max(x(2,k)-x(1,k),0)...
                    -beta*max(x(1,k)-x(2,k),0);
    nominator(k)=exp(lambdaP*FSQRE_AP(k)*utility_offer(k));
end;
for k=1:length(x)
        FSQRE_OP(k)=nominator(k)/sum(nominator); % offer probabilities
end;
%distribution over predicted offer and acceptance probabilities
p_fsqre=[FSQRE_OP; FSQRE_AP];
%%maximum likelihood estimation (unrestricted)
start = [.1; .1;.01;.01]; %start values for alpha, beta, lambda_P, lambda_R
%maximize negative logl by minimizing positive logl
[parameter, fval, exitflag, output, gradient, hessian] = fminunc...
(@fs_logl,start,optimset('Display','off', 'LargeScale','off'),x,off,acc);
%save estimation results
FS_LOGL_unc=logl; % log likelilood
FS_PE_unc=parameter'; %estimates
inv_hessian=inv(hessian);
FS_SD_unc=sqrt(inv_hessian([1 6 11 16])); %standard deviations
```

6.5 ERC Quantal Response Equilibrium

```
start = [.1;.02;.02]; %start values for alpha = beta, lambda_P, lambda_R
%maximize negative logl by minimizing positive logl
[parameter, logl, exitflag, output, gradient, hessian] = fminunc...
(@erc_logl,start,optimset('Display','off', 'LargeScale','off'),c,x,off,acc);
% save estimation results
ERC_LOGL=logl; % log likelilood
ERC_PE=parameter'; %estimates
inv_hessian=inv(hessian);
ERC_SD=sqrt(inv_hessian([1 5 9])); %standard deviations
%label estimates
alpha=parameter(1); lambdaP=parameter(2); lambdaR=parameter(3);
for k=1:length(x) %responder
    sigma=x(2,k)/c;
    if sigma<1/2 % acceptance utilities
        utility_accept(k)=c*(sigma-alpha/2*(sigma-1/2)^2);
    elseif sigma>=1/2
        utility_accept(k) = c*sigma;
    end;
    nominator(k)=exp(lambdaR*utility_accept(k));
    denominator(k)=1+exp(lambdaR*utility_accept(k));
    ERCQRE_AP(k)=nominator(k)/denominator(k); %acceptance probabilities
end;
for k=1:length(x) %proposer
    sigma=x(1,k)/c;
    if sigma<1/2 % offer utilities
        utility_offer(k)=c*(sigma-alpha/2*(sigma-1/2)^2);
```

```
    elseif sigma>=1/2
        utility_offer(k) = c*sigma;
    end;
    nominator(k)=exp(lambdaP*ERCQRE_AP(k)*utility_offer(k));
end;
for k=1:length(x) % offer probabilities
    ERCQRE_OP(k)=nominator(k)/sum(nominator);
end;
%distribution over predicted offer and acceptance probabilities
p_ercqre=[ERCQRE_OP; ERCQRE_AP];
```

7 Out-of-Sample Predictions: Three Person Ultimatum Game

```
c1=1200; %monetary cake
x1(1,:)=[10 8 8 8 6 6 6 6 6 4 4 4 4 4 2 2 2 0]*100; %possible demands
x1(2,:)=[ 1 3 2 1 5 4 3 2 1 6 5 4 3 2 6 5 4 6]*100; % " offers
x1(3,:)=[1 1 2 3 1 2 3 4 5 2 3 4 5 6 4 5 6 6]*100;  % " remainders
off1=[ 402  161   11    0  776  313   54    3    4 ...%actual offers
             168  184 2764   4    3 9   8   3    1];
acc1=[1069 1595 1273 1074 3102 2830 2031 1294 1098 ...% " acceptances
            3828 3983 4719 2018 1384 4006 4105 3123 4023];
% we use one observation less that was misclassified by Güth 2007, however
% the outcomes do not change because of the high total number of observations
p_obs1=[off1/sum(off1);acc1/sum(off1)]; % observed offer and acceptance prob.
```

7.1 Heuristic Mix

```
%minimum acceptance offer (mao)
%      100   100    100         300         400        400    500
mao=[money fear/3 fairness/3 fairness/3 fairness/3 fear/3 fear/3];
%which offer implied by a proposal do i accept depending on my mao?
acc=[ 1 1 1 0 0 0 0; % 100 is the offer of proposal k=1
      1 1 1 1 0 0 0; % 300
      1 1 1 0 0 0 0; % 200
      1 1 1 0 0 0 0; % 100
      1 1 1 1 1 1 1; % 500
      1 1 1 1 1 1 0; % 400
      1 1 1 1 0 0 0; % 300
      1 1 1 0 0 0 0; % 200
      1 1 1 0 0 0 0; % 100
      1 1 1 1 1 1 1; % 600
      1 1 1 1 1 1 1; % 500
      1 1 1 1 1 1 0; % 400
      1 1 1 1 0 0 0; % 300
      1 1 1 0 0 0 0; % 200
      1 1 1 1 1 1 1; % 600
      1 1 1 1 1 1 1; % 500
```

```
      1 1 1 1 1 1 0; % 400
      1 1 1 1 1 1 1];% 600 is the offer of proporal k=18
p_fft1=[money 0 0 0 fear 0 0 0 0 0 0 fairness 0 0 0 0 0 0;
        sum(repmat(mao,18,1).*acc,2)'];%pred. offer & acceptance prob.
```

7.2 Subgame Perfect Equilibrium

```
uRA1=x1(2,:); %acceptance utilities
p_spe1(2,:)=uRA1>0; %predicted acceptance probabilities
EuP1=x1(1,:).*p_spe1(2,:); %expected utilities
p_spe1(1,:)=(EuP1==max(EuP1(1,:)));%predicted offer probabilities
```

7.3 Fehr-Schmidt Subgame Perfect Equilibrium

```
uRA1=repmat(x1(2,:),4,1)... %acceptance utilities
    -1/2*repmat(dp(:,2),1,18).*repmat(max(x1(1,:)-x1(2,:),0),4,1)...
    -1/2*repmat(dp(:,2),1,18).*repmat(max(x1(3,:)-x1(2,:),0),4,1)...
    -1/2*repmat(dp(:,3),1,18).*repmat(max(x1(2,:)-x1(1,:),0),4,1)...
    -1/2*repmat(dp(:,3),1,18).*repmat(max(x1(2,:)-x1(3,:),0),4,1);
p_fsspe1(2,:)=sum(repmat(dp(:,1),1,18).*(uRA1>0)); %pred. acceptance prob.
EuP1=(repmat(x1(1,:),4,1)... %distribution over expected offer utilities
    -1/2*repmat(dp(:,2),1,18).*repmat(max(x1(2,:)-x1(1,:),0),4,1)...
    -1/2*repmat(dp(:,2),1,18).*repmat(max(x1(3,:)-x1(1,:),0),4,1)...
    -1/2*repmat(dp(:,3),1,18).*repmat(max(x1(1,:)-x1(2,:),0),4,1)...
    -1/2*repmat(dp(:,3),1,18).*repmat(max(x1(1,:)-x1(3,:),0),4,1))...
   .*repmat(p_fsspe1(2,:),4,1);
maxEuP1=[repmat(max(EuP1(1,:)),1,18); %expected utilities of each type
         repmat(max(EuP1(2,:)),1,18);
         repmat(max(EuP1(3,:)),1,18);
         repmat(max(EuP1(4,:)),1,18)];
p_fsspe1(1,:)=sum((EuP1==maxEuP1).*repmat(dp(:,1),1,18)); %pred. offer prob.
```

7.4 Fehr-Schmidt Quantal Response Equilibrium

```
alpha=FS_PE(1); beta=FS_PE(1); %parameter estimates
lambdaP=FS_PE(2); lambdaR=FS_PE(3);
for k=1:length(x1) %responder
    utility_accept(k)=x1(2,k)... %acceptance utilities
                     -1/2*alpha*max(x1(1,k)-x1(2,k),0)...
                     -1/2*alpha*max(x1(3,k)-x1(2,k),0)...
                     -1/2* beta*max(x1(2,k)-x1(1,k),0)...
                     -1/2* beta*max(x1(2,k)-x1(3,k),0);
    nominator(k)=exp(lambdaR*utility_accept(k));
    denominator(k)=1+exp(lambdaR*utility_accept(k));
    FSQRE_AP(k)=nominator(k)/denominator(k); %acceptance probabilities
end;
```

```
for k=1:length(x1)%proposer
    utility_offer(k)=x1(1,k)... %offer utilities
                      -1/2*alpha*max(x1(2,k)-x1(1,k),0)...
                      -1/2*alpha*max(x1(3,k)-x1(1,k),0)...
                      -1/2* beta*max(x1(1,k)-x1(2,k),0)...
                      -1/2* beta*max(x1(1,k)-x1(3,k),0);
    nominator(k)=exp(lambdaP*FSQRE_AP(k)*utility_offer(k));
end;
for k=1:length(x1) %offer probabilities
        FSQRE_OP(k)=nominator(k)/sum(nominator);
end;
p_fsqre1=[FSQRE_OP; FSQRE_AP];%predicted offer and acceptance probabilities
```

7.5 ERC Quantal Response Equilibrium

```
alpha=ERC_PE(1); lambdaP=ERC_PE(2); lambdaR=ERC_PE(3); %parameter estimates
for k=1:length(x1) %responder
    sigma=x1(2,k)/c1;
    if sigma<1/3 %acceptance utilities
        utility_accept(k)=c1*(sigma-alpha/2*(sigma-1/3)^2);
    elseif sigma>=1/3
        utility_accept(k) = c1*sigma;
    end;
    nominator(k)=exp(lambdaR*utility_accept(k));
    denominator(k)=1+exp(lambdaR*utility_accept(k));
    ERCQRE_AP(k)=nominator(k)/denominator(k); %acceptance probabilities
end;
for k=1:length(x1) %proposer
    sigma=x1(1,k)/c1;
    if sigma<1/3 % offer utilities
        utility_offer(k)=c*(sigma-alpha/2*(sigma-1/3)^2);
    elseif sigma>=1/3
        utility_offer(k) = c1*sigma;
    end;
    nominator(k)=exp(lambdaP*ERCQRE_AP(k)*utility_offer(k));
end;
for k=1:length(x1) % offer probabilities
    ERCQRE_OP(k)=nominator(k)/sum(nominator);
end;
p_ercqre1=[ERCQRE_OP; ERCQRE_AP];%predicted offer and acceptance prob.
```

8 Statistics: Ultimatum Game

8.1 Average Offer

```
AVGO=[sum(p_obs(1,:).*x(2,:)/c);     % observations
```

```
      sum(p_fft(1,:).*x(2,:)/c);     % fft
      sum(p_spe(1,:).*x(2,:)/c);     % spe
      sum(p_fsspe(1,:).*x(2,:)/c);   % fs spe
      sum(p_fsqre(1,:).*x(2,:)/c);   % fs qre
      sum(p_ercqre(1,:).*x(2,:)/c);];% erc qre
```

8.2 Average Acceptance Rate

```
%Outcome probabilities (offer pr * acceptance pr & offer pr * rejection pr)
p_out=[p_obs(1,:)   .*p_obs(2,:)    p_obs(1,:).*(1-p_obs(2,:));
       p_fft(1,:)   .*p_fft(2,:)    p_fft(1,:).*(1-p_fft(2,:));
       p_spe(1,:)   .*p_spe(2,:)    p_spe(1,:).*(1-p_spe(2,:));
       p_fsspe(1,:) .*p_fsspe(2,:)  p_fsspe(1,:).*(1-p_fsspe(2,:));
       p_fsqre(1,:) .*p_fsqre(2,:)  p_fsqre(1,:).*(1-p_fsqre(2,:));
       p_ercqre(1,:).*p_ercqre(2,:) p_ercqre(1,:).*(1-p_ercqre(2,:))];
%Average acceptance rate for obs, fft, spe, fs spe, fs qre, erc qre
AVGA=sum(p_out(:,1:9),2);
```

8.3 Sum of Absolute Differences for Predicted and Observed Probabilities

SAD = [offer probabilities acceptance probabilities outcome probalities]

```
SAD=[sum(abs(p_obs-p_fft),2)'    sum(abs(p_out(1,:)-p_out(2,:)),2);
     sum(abs(p_obs-p_spe),2)'    sum(abs(p_out(1,:)-p_out(3,:)),2);
     sum(abs(p_obs-p_fsspe),2)'  sum(abs(p_out(1,:)-p_out(4,:)),2);
     sum(abs(p_obs-p_fsqre),2)'  sum(abs(p_out(1,:)-p_out(5,:)),2);
     sum(abs(p_obs-p_ercqre),2)' sum(abs(p_out(1,:)-p_out(6,:)),2)];
```

8.4 Correlation between Predicted and Observed Probabilities

```
CORR=[corr(p_obs(1,:)',p_fft(1,:)') corr(p_obs(2,:)',p_fft(2,:)')...
      corr(p_out(1,:)',p_out(2,:)'); %fft
      corr(p_obs(1,:)',p_fsspe(1,:)') corr(p_obs(2,:)',p_fsspe(2,:)')...
      corr(p_out(1,:)',p_out(4,:)'); % fs spe
      corr(p_obs(1,:)',p_fsqre(1,:)') corr(p_obs(2,:)',p_fsqre(2,:)')...
      corr(p_out(1,:)',p_out(5,:)'); % fs qre
      corr(p_obs(1,:)',p_ercqre(1,:)') corr(p_obs(2,:)',p_ercqre(2,:)')...
      corr(p_out(1,:)',p_out(6,:)')]; % erc qre
```

8.5 Relative Fit Index (RFI)

```
RFI=(repmat(SAD(2,:),4,1)-SAD([1 3 4 5],:))./repmat(SAD(2,:),4,1);
```

9 Statistics: Three-Person Ultimatum Game

9.1 Average Offer

```
AVGO1=[sum(p_obs1(1,:).*x1(2,:)/c1);     % observations
      sum(p_fft1(1,:).*x1(2,:)/c1);     % fft
      sum(p_spe1(1,:).*x1(2,:)/c1);     % spe
      sum(p_fsspe1(1,:).*x1(2,:)/c1);   % fs spe
      sum(p_fsqre1(1,:).*x1(2,:)/c1);   % fs qre
      sum(p_ercqre1(1,:).*x1(2,:)/c1);];% erc qre
```

9.2 Average Acceptance Rate

```
%Outcome probabilities (offer pr * acceptance pr & offer pr * rejection pr)
p_out1=[p_obs1(1,:)   .*p_obs1(2,:)    p_obs1(1,:).*(1-p_obs1(2,:));
       p_fft1(1,:)   .*p_fft1(2,:)    p_fft1(1,:).*(1-p_fft1(2,:));
       p_spe1(1,:)   .*p_spe1(2,:)    p_spe1(1,:).*(1-p_spe1(2,:));
       p_fsspe1(1,:) .*p_fsspe1(2,:)  p_fsspe1(1,:).*(1-p_fsspe1(2,:));
       p_fsqre1(1,:) .*p_fsqre1(2,:)  p_fsqre1(1,:).*(1-p_fsqre1(2,:));
       p_ercqre1(1,:).*p_ercqre1(2,:) p_ercqre1(1,:).*(1-p_ercqre1(2,:))];
%Average acceptance rate for obs, fft, spe, fs spe, fs qre, erc qre
AVGA1=sum(p_out1(:,1:18),2);
```

9.3 Sum of Absolute Differences for Predicted and Observed Probabilities

SAD = [offer probabilities acceptance probabilities outcome probalities]

```
SAD1=[sum(abs(p_obs1-p_fft1),2)'    sum(abs(p_out1(1,:)-p_out1(2,:)),2);
      sum(abs(p_obs1-p_spe1),2)'    sum(abs(p_out1(1,:)-p_out1(3,:)),2);
      sum(abs(p_obs1-p_fsspe1),2)'  sum(abs(p_out1(1,:)-p_out1(4,:)),2);
      sum(abs(p_obs1-p_fsqre1),2)'  sum(abs(p_out1(1,:)-p_out1(5,:)),2);
      sum(abs(p_obs1-p_ercqre1),2)' sum(abs(p_out1(1,:)-p_out1(6,:)),2)];
```

9.4 Correlation between Predicted and Observed Probabilities

```
CORR1=[corr(p_obs1(1,:)',p_fft1(1,:)') corr(p_obs1(2,:)',p_fft1(2,:)')...
    corr(p_out1(1,:)',p_out1(2,:)'); %fft
    corr(p_obs1(1,:)',p_fsspe1(1,:)') corr(p_obs1(2,:)',p_fsspe1(2,:)')...
    corr(p_out1(1,:)',p_out1(4,:)'); % fs spe
    corr(p_obs1(1,:)',p_fsqre1(1,:)') corr(p_obs1(2,:)',p_fsqre1(2,:)')...
    corr(p_out1(1,:)',p_out1(5,:)'); % fs qre
    corr(p_obs1(1,:)',p_ercqre1(1,:)') corr(p_obs1(2,:)',p_ercqre1(2,:)')...
    corr(p_out1(1,:)',p_out1(6,:)')]; % erc qre
```

9.5 Relative Fit Index (RFI)

```
RFI1=(repmat(SAD1(2,:),4,1)-SAD1([1 3 4 5],:))./repmat(SAD1(2,:),4,1);
```

10 Output

```
display('ULTIMATUM GAME - RESULTS')
display(' ')
display('Offer probabilities for each possible offer o_k of')
disptable(roundn([p_obs(1,:); p_fft(1,:); p_spe(1,:);...
    p_fsspe(1,:); p_fsqre(1,:); p_ercqre(1,:)],-4),...
    '100|200|300|400|500|600|700|800|900', 'obs|fft|spe|fsspe|fsqre|ercqre')
display('Acceptance probabilities for each possible offer o_k of')
disptable(roundn([p_obs(2,:); p_fft(2,:); p_spe(2,:);...
    p_fsspe(2,:); p_fsqre(2,:); p_ercqre(2,:)],-4),...
    '100|200|300|400|500|600|700|800|900', 'obs|fft|spe|fsspe|fsqre|ercqre')
disptable(roundn([AVGO AVGA],-2),'average offer|average acceptance rate',...
    'obs|fft|spe|fsspe|fsqre|ercqre')
display('Sum of absolute differences between pred. and obs. prob. for')
disptable(roundn(SAD,-2),...
    'offers|acceptances|outcomes', 'fft|spe|fsspe|fsqre|ercqre')
display('Correlation between predicted and observed probabilities for')
disptable(roundn(CORR,-2),...
    'offers|acceptances|outcomes', 'fft|fsspe|fsqre|ercqre')
display('Relative fit to subgame perfect equilibrium for')
disptable(roundn(RFI,-2),...
    'offers|acceptances|outcomes', 'fft|fsspe|fsqre|ercqre')

display('THREE-PERSON ULTIMATUM GAME - RESULTS')
display(' ')
display('Offer probabilities for each possible proposal (d_k,o_k) of')

disptable(roundn([x1(2,:); p_obs1(1,:); p_fft1(1,:); p_spe1(1,:);...
p_fsspe1(1,:); p_fsqre1(1,:); p_ercqre1(1,:)],-4),...
'1000|800|800|800|600|600|600|600|600|400|400|400|400|400|200|200|200|0',...
'|obs|fft|spe|fsspe|fsqre|ercqre')
display('Acceptance probabilities for each possible proposal (d_k,o_k) of')
disptable(roundn([x1(2,:);p_obs1(2,:); p_fft1(2,:); p_spe1(2,:);...
p_fsspe1(2,:); p_fsqre1(2,:); p_ercqre1(2,:)],-4),...
'1000|800|800|800|600|600|600|600|600|400|400|400|400|400|200|200|200|0',...
'|obs|fft|spe|fsspe|fsqre|ercqre')
disptable(roundn([AVGO1 AVGA1],-2),...
'average offer|average acceptance rate', 'obs|fft|spe|fsspe|fsqre|ercqre')
display('Sum of absolute differences between pred. and obs. prob. for')
disptable(roundn(SAD1,-2),...
    'offers|acceptances|outcomes', 'fft|spe|fsspe|fsqre|ercqre')
display('Correlation between predicted and observed probabilities for')
```

```
disptable(roundn(CORR1,-2),...
    'offers|acceptances|outcomes', 'fft|fsspe|fsqre|ercqre')
display('Relative fit to subgame perfect equilibrium for')
disptable(roundn(RFI1,-2),...
    'offers|acceptances|outcomes', 'fft|fsspe|fsqre|ercqre')

display('ESTIMATION RESULTS')
display(' ')
display('Heuristic')
display('Distribution of proposer criteria')
disptable(roundn([fairness, fear, money],-2),'fairness|fear|money','=')
display('Equal distribution of (1/3,1/3,1/3) for responder criteria')
disptable(roundn([X2 df pv],-3),'X2|df|p','=')
display('Log-likelihood of the quantal response models')
disptable([-FS_LOGL,-FS_LOGL_unc,-ERC_LOGL],'fs_unc|fs_con|erc','=')
display('Fehr-Schmidt quantal response equilibrium with alpha>=beta')
disptable(roundn([FS_PE; FS_SD],-4),...
'alpha=beta|lambda_P|lambda_R','parameter estimates|standard deviations')
display('Fehr-Schmidt quantal response equilibrium (unrestricted)')
disptable(roundn([FS_PE_unc; FS_SD_unc],-4),...
'alpha|beta|lambda_P|lambda_R','parameter estimates|standard deviations')
display('ERC quantal response equilibrium')
disptable(roundn([ERC_PE; ERC_SD],-4),...
'alpha|lambda_P|lambda_R','parameter estimates|standard deviations')
```

11 Function fs_logl

```
%compute log-likelihood for Fehr-Schmidt QRE model
function logl = fs_logl(parameter,x,off,acc)
for k=1:length(parameter) % non--negativity restriction (lower boundary)
    if parameter(k)<0
        parameter(k)=0;
    end
end
if length(parameter)==3 %alpha>=beta
    alpha=parameter(1); beta=parameter(1);
    lambdaP=parameter(2); lambdaR=parameter(3);
else % unrestricted
    alpha=parameter(1); beta=parameter(2);
    lambdaP=parameter(3); lambdaR=parameter(4);
end
for k=1:length(x) %responder
    utility_accept(k)=x(2,k)... %acceptance utilities
        -alpha*max(x(1,k)-x(2,k),0)-beta*max(x(2,k)-x(1,k),0);
    nominator(k)=exp(lambdaR*utility_accept(k));
    denominator(k)=1+exp(lambdaR*utility_accept(k));
    FSQRE_AP(k)=nominator(k)/denominator(k); %acceptance probabilities
```

```
end;
for k=1:length(x) %proposer
    utility_offer(k)=x(1,k)... %offer utilities
        -alpha*max(x(2,k)-x(1,k),0)-beta*max(x(1,k)-x(2,k),0);
    nominator(k)=exp(lambdaP*FSQRE_AP(k)*utility_offer(k));
end;
for k=1:length(x)
    FSQRE_OP(k)=nominator(k)/sum(nominator); %offer probabilities
end;
logl=0; %% log-likelihood function
for k=1:length(x)
    logl=logl + off(k)*acc(k)/sum(off) * log(FSQRE_OP(k)*FSQRE_AP(k))...
    + off(k)*(sum(off)-acc(k))/sum(off) * log(FSQRE_OP(k)*(1-FSQRE_AP(k)));
end;
logl=-logl;
```

12 Function erc_logl

```
%compute log-likelihood for ERC QRE model
function logl = erc_logl(parameter,c,x,off,acc)
alpha=parameter(1); lambdaP=parameter(2); lambdaR=parameter(3);
for k=1:length(x) %responder
    sigma=x(2,k)/c;
    if sigma<1/2 %acceptance utilities
        utility_accept(k)=c*(sigma-alpha/2*(sigma-1/2)^2);
    elseif sigma>=1/2
        utility_accept(k) = c*sigma;
    end;
    nominator(k)=exp(lambdaR*utility_accept(k));
    denominator(k)=1+exp(lambdaR*utility_accept(k));
    ERCQRE_AP(k)=nominator(k)/denominator(k); %acceptance probabilities
end;
for k=1:length(x) %proposer
    sigma=x(1,k)/c;
    if sigma<1/2 %offer utilities
        utility_offer(k)=c*(sigma-alpha/2*(sigma-1/2)^2);
    elseif sigma>=1/2
        utility_offer(k) = c*sigma;
    end;
    nominator(k)=exp(lambdaP*ERCQRE_AP(k)*utility_offer(k));
end;
for k=1:length(x) %offer probabilities
    ERCQRE_OP(k)=nominator(k)/sum(nominator);
end;
logl=0;%log-likelihood function
for k=1:length(x)
    logl=logl + off(k)*acc(k)/sum(off)*log(ERCQRE_OP(k)*ERCQRE_AP(k))...
```

```
    +off(k)*(sum(off)-acc(k))/sum(off)*log(ERCQRE_OP(k)*(1-ERCQRE_AP(k)));
end;
logl=-logl;
```

Appendix D

13 Fit of Own Baseline Models vs Competition Baseline Models

Table 5 compares the mean squared deviations for F and S of each baseline model that we consider to the baseline models that were provided by the organizers of the competition.

Table 5
Comparison of Fit: Our Baseline Models versus Competition Baseline Models

Section	Model	Ert et al. (2011)		Our Results	
		MSD^F_{est}	MSD^S_{est}	MSD^F_{est}	MSD^S_{est}
5.3.1	SPE	0.0529	0.0105	0.0545	0.0105
5.3.2	QRE	-	-	0.0170	0.0092
5.3.3	FS-QRE	0.0307	0.0099	0.0118	0.0082
5.3.4	BO-QRE	0.0367	0.0100	0.0141	0.0073
5.3.5	CR-QRE	0.0292	0.0041	0.0112	0.0042
5.3.6	7S	0.0121	0.0029	0.0119	0.0029

- Both SPE models are identical. The difference in the MSD^F_{est} may be due to a rounding error. Moreover, we estimated additionally a QRE model without social preferences that allows us to separate the effect of choice errors in comparison to the effect of social preferences.
- All social preference models that we consider have better fits (the only exception is the Charness-Rabin QRE model that performs a little worse in case of S). The reasons for the differences in MSD scores are due to different assumptions and maybe due to the use of different estimation procedures. The estimation procedures used by Ert et al. (2011) are not implemented in the computer programs that are available on the competition homepage, therefore we cannot infer how they estimated the models.[32]
- Our estimation procedure is implemented in the MATLAB codes in Appendix 16 and works as follows: We apply each baseline model first to the second mover and estimate the parameters that minimize the mean of the 120 squared differences between observed and predicted *right* choice probabilities. Then we apply each baseline model to the

[32] `https://sites.google.com/site/extformpredcomp/baseline-models`.

first mover and estimate the parameters that minimize the mean of the 120 squared differences between observed and predicted *in* choice probabilities. This implies that the parameter estimates for S in the second mover model may be different from the parameter estimates for S in the first mover model.

- The most important differences in assumptions between our social preference models and the ones considered by Ert et al. (2011) are: We use average values for the parameter estimates of the Fehr-Schmidt QRE model (instead of probability distributions) and maintain the parameter restrictions $\alpha_F \geq \beta_F$ and $\alpha_S \geq \beta_S$. We use the utility function of Bolton and Ockenfels (2000) in the Bolton-Ockenfels QRE model instead of the modified utility function of De Bruyn and Bolton (2008), we do not permit for learning across games, and we conduct a more accurate payoff transformation (see Section 5.3.4). The remaining differences can be infered by a comparison of our codes and the codes on the competition homepage as well as from the description of our models and the description of the models in Ert et al. (2011).
- Lastly, we consider the seven strategies model as it is described in Ert et al. (2011). The computer program on the competition homepage that was used to compute the MSD scores by the organizers specifies a slightly different model which explains the differences in MSD scores in the case of F. The differences are outlined in the MATLAB code (see Appendix 16.14 and Appendix 16.15) that we use to compute the estimates and MSD scores of the seven strategies model as it is specified in Ert et al. (2011).

14 Experimental Data

Table 6 presents the observed *right* choice probability $p_\Gamma^r \in [0,1]$ and the observed *in* choice probability $p_\Gamma^i \in [0,1]$ for each game $\Gamma \in \{1,2,...,120\}$ of the estimation set.[33]

Table 6
Observed Behavior in Each Game of the Estimation Set

Γ	x_F^o	x_S^o	x_F^l	x_S^l	x_F^r	x_S^r	p_Γ^i	p_Γ^r	Γ	x_F^o	x_S^o	x_F^l	x_S^l	x_F^r	x_S^r	p_Γ^i	p_Γ^r
1	-3	-7	-7	5	-3	6	0.20	0.93	61	0	2	-1	6	2	6	0.79	0.82
2	7	-6	5	3	-4	0	0.10	0.07	62	2	-6	3	-2	-4	5	0.07	0.96
3	-7	-4	-5	8	-5	8	1.00	0.47	63	5	3	-3	6	0	8	0.00	1.00
4	-3	-4	-1	4	3	7	1.00	0.97	64	6	-2	1	1	4	-3	0.07	0.04
5	-2	-7	6	5	-2	8	0.83	0.77	65	-7	-4	-4	-5	-6	-5	0.96	0.29
6	-1	4	-7	3	7	0	0.13	0.13	66	-6	-2	1	-4	-4	-4	0.96	0.50
7	-3	-1	1	-7	-4	6	0.13	0.97	67	-3	-1	-5	-3	6	0	0.96	0.96
8	0	2	-5	1	2	1	0.47	0.83	68	-1	4	2	-4	-3	-2	0.18	0.96
9	1	-7	-4	4	3	4	0.40	0.83	69	-4	8	-3	4	1	-4	0.79	0.00
10	-2	5	-4	7	-2	-2	0.07	0.00	70	-6	-6	0	8	-1	-6	1.00	0.00
11	-2	1	5	-7	5	-7	0.93	0.43	71	7	-1	5	4	-5	7	0.04	0.89
12	2	-7	3	3	4	-6	0.93	0.03	72	0	0	7	0	-5	7	0.18	0.96
13	5	-6	0	1	1	-7	0.07	0.07	73	5	-3	2	0	4	5	0.36	1.00
14	-6	2	-2	-4	6	-5	1.00	0.17	74	-3	7	0	-5	-3	-6	0.89	0.07
15	1	-4	1	2	6	5	0.93	0.97	75	-5	3	-6	5	0	1	0.36	0.11
16	0	8	1	-6	-6	3	0.00	1.00	76	7	-7	5	6	-7	-5	0.29	0.00
17	0	0	5	-4	-5	-5	0.33	0.07	77	-3	2	0	-5	-6	7	0.18	1.00
18	3	-4	-2	8	3	-5	0.00	0.00	78	6	-1	-5	6	8	8	0.75	1.00
19	1	-6	4	5	-1	-6	0.80	0.03	79	0	0	-5	5	-1	-6	0.04	0.00
20	0	-3	-4	6	3	3	0.10	0.20	80	-1	-2	6	0	-3	1	0.39	0.89
21	-6	2	0	-7	8	-7	1.00	0.63	81	-6	-5	0	-1	-4	-6	0.96	0.00
22	-5	7	-6	-2	8	-6	0.33	0.17	82	0	8	6	5	8	3	0.96	0.00
23	-3	-5	3	-3	6	-1	0.97	0.97	83	-6	-6	-1	-1	5	-5	0.96	0.00
24	2	-4	-3	5	3	-2	0.00	0.07	84	-4	1	1	-8	8	0	1.00	1.00
25	-7	1	-6	1	0	3	1.00	0.93	85	7	7	-1	-7	8	-1	0.14	1.00
26	-1	4	1	0	5	-2	0.97	0.03	86	-5	1	0	-6	-4	-4	1.00	1.00
27	4	1	-6	1	7	5	0.50	1.00	87	3	-2	2	0	6	3	0.93	1.00
28	5	-7	-5	-7	8	0	0.57	1.00	88	-5	6	-4	0	-4	0	0.79	0.46
29	0	1	-2	5	5	-5	0.10	0.03	89	6	-7	-3	-3	6	-7	0.04	0.00
30	2	-8	7	-1	6	-7	0.97	0.00	90	-4	-4	0	-7	7	-4	1.00	1.00
31	-6	-6	2	-3	1	2	1.00	1.00	91	2	-6	-7	-1	0	-6	0.04	0.00
32	-3	6	2	-4	5	-3	1.00	0.93	92	0	3	5	-2	0	5	0.86	1.00
33	4	0	-8	-8	5	5	0.57	0.97	93	2	4	-7	7	3	-5	0.00	0.00
34	2	2	3	-7	3	-7	0.67	0.43	94	3	-6	3	-6	6	-5	0.82	1.00
35	0	1	6	1	8	-6	1.00	0.03	95	1	3	5	2	-4	3	0.21	0.89
36	-5	3	7	-7	-4	5	1.00	0.97	96	2	6	-3	6	3	-3	0.00	0.04
37	-6	1	-2	-6	1	-3	1.00	0.97	97	0	6	3	-7	6	-2	1.00	1.00
38	2	8	6	1	-7	-7	0.63	0.03	98	4	8	6	3	1	2	0.61	0.07
39	5	7	7	-7	0	4	0.03	1.00	99	-5	7	-2	-1	4	-4	0.96	0.07
40	4	4	7	2	-3	1	0.33	0.03	100	6	-1	-7	-5	5	4	0.25	1.00
41	1	7	3	2	-4	-6	0.63	0.03	101	2	-6	7	-8	6	-6	0.96	0.96
42	4	-3	-3	7	5	7	0.23	0.93	102	-1	-6	-4	-2	-4	-3	0.07	0.04
43	-7	1	6	4	0	-6	1.00	0.00	103	-7	-5	-5	5	-6	-3	0.96	0.00
44	-7	6	-5	2	-4	7	1.00	0.97	104	-5	-4	4	-7	2	2	1.00	0.96
45	-1	7	4	6	-2	5	0.87	0.07	105	8	-5	-6	-5	-3	6	0.00	1.00
46	-4	-2	7	6	-4	3	1.00	0.03	106	4	-1	6	3	0	-2	0.93	0.00
47	7	-7	-6	-6	-4	-6	0.00	0.63	107	4	-3	5	-7	2	-3	0.11	0.96
48	7	-7	1	-5	5	2	0.17	0.97	108	7	-8	4	0	2	2	0.29	0.93
49	5	-2	4	-5	-3	5	0.00	1.00	109	1	-2	6	6	-3	7	0.50	0.75
50	-1	2	0	1	-5	-3	0.50	0.00	110	1	7	0	2	3	2	0.61	0.79
51	2	-7	-1	3	7	-6	0.27	0.00	111	-5	6	4	-4	5	4	1.00	1.00
52	8	1	0	3	2	2	0.00	0.30	112	5	-6	1	-1	6	-1	0.21	0.61
53	-6	2	-7	-2	0	2	0.97	0.97	113	4	-7	2	-4	7	-4	0.57	0.64
54	-1	3	7	1	-2	8	0.37	0.87	114	-4	1	-3	6	6	1	0.96	0.07
55	6	0	-5	4	0	-2	0.00	0.07	115	3	2	-7	-6	5	-3	0.36	0.96
56	0	-2	8	3	7	-7	1.00	0.00	116	1	1	2	-1	2	-1	0.75	0.57
57	3	0	-4	3	-2	-2	0.00	0.17	117	-3	-6	8	0	7	3	1.00	1.00
58	-6	0	-1	-5	-6	-7	0.93	0.07	118	-5	-6	7	7	4	-6	0.96	0.07
59	-6	5	2	0	7	-8	0.97	0.07	119	2	-4	-7	-6	2	-3	0.29	0.93
60	-8	3	6	4	-3	-2	0.93	0.07	120	-2	-7	-3	-6	0	3	0.96	0.93

Source. Ert et al. (2011)

[33] The text file that contains the data can be downloaded from `https://sites.google.com/site/extformpredcomp/baseline-models/erc/ext_est_res.txt?attredirects=0&d=1`.

Table 6 presents the observed *right* choice probability $p^r_\Gamma \in [0,1]$ and the observed *in* choice probability $p^i_\Gamma \in [0,1]$ for each game $\Gamma \in \{1, 2, ..., 120\}$ of the prediction set.[34]

Table 7
Observed Behavior in Each Game of the Prediction Set

Γ	x^o_F	x^o_S	x^l_F	x^l_S	x^r_F	x^r_S	p^i_Γ	p^r_Γ	Γ	x^o_F	x^o_S	x^l_F	x^l_S	x^r_F	x^r_S	p^i_Γ	p^r_Γ
1	-1	1	-4	6	-4	6	0.03	0.52	61	2	-6	6	3	-1	2	0.59	0.15
2	1	7	-5	1	7	-4	0.06	0.09	62	1	-5	-5	5	1	-1	0.07	0.15
3	-4	5	-3	4	-4	5	0.88	0.79	63	3	-1	3	2	1	-4	0.44	0.07
4	-4	2	5	-4	8	5	0.97	1.00	64	5	-2	5	-6	-4	0	0.00	0.96
5	-4	-2	-6	-7	6	-3	0.94	1.00	65	-6	7	-5	-3	0	1	1.00	0.96
6	0	-7	-6	5	-6	6	0.03	0.97	66	5	6	-5	-6	6	0	0.22	0.96
7	-1	-5	-6	6	-6	-3	0.09	0.00	67	1	-6	-6	2	4	-6	0.22	0.07
8	0	3	7	-7	7	-4	0.88	1.00	68	2	4	3	-6	8	5	0.93	0.96
9	7	0	6	4	-1	-7	0.21	0.00	69	1	-4	3	5	-3	1	0.74	0.04
10	0	-6	-5	7	8	-4	0.03	0.06	70	6	-1	-3	4	-2	-7	0.04	0.07
11	8	-3	-4	-3	6	1	0.21	1.00	71	-7	-2	-4	-3	-4	-1	1.00	0.96
12	-6	-2	-7	4	0	-4	0.27	0.00	72	-4	5	-7	-8	-2	2	0.81	1.00
13	2	-7	-3	-6	-8	2	0.06	0.97	73	6	-7	-8	7	-3	3	0.04	0.11
14	6	-1	1	-3	8	0	0.58	0.94	74	8	-3	1	0	-3	-1	0.00	0.15
15	-1	5	4	1	8	-1	0.91	0.03	75	2	-8	0	0	4	-4	0.26	0.04
16	-3	-4	2	1	-5	1	0.67	0.24	76	4	0	4	-8	-1	5	0.00	1.00
17	-7	-6	8	3	-1	3	0.97	0.21	77	8	-8	7	2	4	-8	0.26	0.00
18	2	-3	-1	-8	8	7	0.85	1.00	78	5	0	3	-6	-7	3	0.00	0.93
19	-1	-3	1	-7	-8	-5	0.03	0.85	79	4	-7	-1	-5	-4	7	0.04	1.00
20	-3	2	0	4	-3	-3	0.85	0.00	80	-6	1	-7	6	-1	-4	0.33	0.04
21	0	6	-6	-7	0	7	0.24	1.00	81	-4	-1	-3	-6	8	-7	0.81	0.11
22	-4	-5	-8	-5	5	3	0.79	1.00	82	2	3	0	7	8	5	0.30	0.19
23	-8	3	8	4	-6	-1	0.97	0.00	83	-4	6	-7	6	8	-8	0.22	0.00
24	7	-4	6	0	4	-4	0.27	0.00	84	-7	-4	2	2	8	4	0.96	0.85
25	4	-3	8	6	-1	-4	0.73	0.00	85	-3	1	5	-8	6	-2	0.96	0.96
26	4	1	5	7	-7	4	0.55	0.00	86	7	-8	6	-7	7	6	0.59	1.00
27	4	2	-3	-8	8	-5	0.30	0.94	87	-3	2	-8	1	-8	4	0.00	0.96
28	-2	-3	-3	3	6	-4	0.27	0.06	88	-2	1	-2	-6	7	-5	0.96	0.85
29	-5	2	5	-5	-4	-8	0.94	0.06	89	-4	-3	-1	5	-3	2	0.85	0.00
30	0	2	2	-6	-4	-2	0.03	0.94	90	-6	0	7	1	-1	-8	0.96	0.04
31	6	-3	4	-8	7	-7	0.39	0.94	91	-4	-1	-3	1	-6	3	0.19	0.89
32	1	-1	-4	2	1	3	0.48	1.00	92	-2	-2	7	2	7	-5	0.96	0.04
33	-7	-5	-1	-4	4	2	0.97	1.00	93	-2	-2	8	-6	-5	-4	0.19	0.81
34	-4	3	7	4	3	2	0.97	0.00	94	-7	-6	-7	0	1	-8	0.81	0.00
35	1	-3	6	3	-6	-2	0.64	0.00	95	4	1	-8	2	-6	7	0.04	1.00
36	4	1	8	-4	1	-7	0.58	0.03	96	-8	-8	2	-5	-3	3	0.93	1.00
37	0	0	-1	-7	4	-6	0.55	0.97	97	2	-6	5	-1	1	5	0.26	0.96
38	4	-6	3	-4	-6	7	0.06	0.97	98	3	5	4	-3	-4	-2	0.00	0.85
39	6	4	-1	6	7	-2	0.00	0.00	99	-4	3	-6	-8	-3	5	0.78	0.96
40	1	2	-7	4	-5	2	0.00	0.09	100	-4	-3	-6	5	-6	5	0.04	0.26
41	3	7	2	-5	8	-2	0.61	0.97	101	-4	-1	4	1	3	-6	0.96	0.00
42	-4	4	1	-3	1	-3	0.88	0.42	102	-7	-1	-1	-3	-4	-8	0.93	0.04
43	0	-8	4	2	-8	1	0.58	0.03	103	-2	3	4	-5	-5	-2	0.22	0.96
44	-7	5	5	-5	-8	7	0.30	0.94	104	-7	1	1	-5	-8	7	0.22	0.93
45	8	-3	-3	-5	1	5	0.06	0.97	105	-8	6	5	6	2	-6	0.96	0.04
46	2	6	-2	-8	2	7	0.36	1.00	106	-1	-1	2	-4	-5	-5	0.56	0.19
47	6	-7	3	-7	-2	-1	0.06	1.00	107	7	-5	2	5	3	-1	0.04	0.04
48	5	1	-8	1	6	7	0.52	1.00	108	3	-6	1	-3	1	-3	0.07	0.44
49	3	7	6	6	1	-1	0.79	0.00	109	-3	-3	-2	7	1	-2	1.00	0.04
50	-2	3	5	-2	-3	-2	0.88	0.45	110	0	7	1	-3	-5	4	0.04	0.93
51	-6	-6	-4	7	2	-7	0.94	0.06	111	1	1	-5	5	7	5	0.48	0.56
52	-7	5	-5	2	-1	-2	0.97	0.09	112	-8	-3	0	-7	-6	-1	0.89	0.93
53	-1	-4	-8	2	2	-8	0.09	0.00	113	5	0	-3	1	-1	2	0.04	0.96
54	0	-7	6	6	-6	-1	0.70	0.00	114	4	-1	1	-4	8	5	0.81	0.93
55	-6	-6	1	-5	-2	-3	1.00	0.97	115	-5	-4	-4	6	8	4	0.89	0.26
56	-7	-7	1	5	7	7	1.00	1.00	116	6	-5	-1	-2	8	1	0.56	0.93
57	-2	0	3	-4	2	1	1.00	1.00	117	5	-3	1	-7	6	-7	0.19	0.41
58	-4	-5	2	-1	7	-6	0.97	0.03	118	0	-6	-6	-1	-1	3	0.11	1.00
59	2	-6	1	-8	8	2	0.88	0.97	119	2	-4	-1	-5	3	5	0.67	0.93
60	3	7	-2	0	5	1	0.30	0.91	120	-3	3	6	-7	0	0	0.89	0.78

Source. `https://sites.google.com/site/extformpredcomp/competition-results-and`

[34] The text file that contains the data can be downloaded from `https://sites.google.com/site/extformpredcomp/competition-results-and/ext_pred_res.txt?attredirects=0&d=1`.

15 Correlation and Error Matrices

The MATLAB code that was used to conduct the computations is attached in Appendix 16.32.

Table 8
Competition Procedure: Correlation Matrix for First Mover Model Predictions

Model	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
(1) SPE	1							
(2) QRE	.9212	1						
(3) FS-QRE	.9321	.9848	1					
(4) BO-QRE	.9315	.9805	.9951	1				
(5) CR-QRE	.9277	.9816	.9985	.9936	1			
(6) 7S	.9026	.9507	.9496	.9383	.9482	1		
(7) SQRE	.8992	.9684	.9741	.9668	.9762	.9787	1	
(8) SVO-SQRE	.8895	.9627	.9692	.9618	.972	.9779	.9985	1

Correlations of model sets with a lower mean predictive error than the one of the best singular model are underlined.

Table 9
Competition Procedure: Mean Predictive Error Matrix for Binary First Mover Model Sets

Model	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
(1) SPE	.0532							
(2) QRE	.0233	.0141						
(3) FS-QRE	.0247	.0129	.0140					
(4) BO-QRE	.0261	.0142	.0152	.0172				
(5) CR-QRE	.0245	.0128	.0140	.0152	.0143			
(6) 7S	.0182	.0079	.0076	.0084	.0076	.0083		
(7) SQRE	.0197	.0093	.0097	.0108	.0100	.0070	.0094	
(8) SVO-SQRE	.0184	.0087	.0092	.0102	.0095	.0068	.0091	.0090

Model sets with a lower mean predictive error than the one of the best singular model are underlined.

Table 10
Competition Procedure: Correlation Matrix for Second Mover Model Predictions

Model	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
(1) SPE	1							
(2) QRE	.9971	1						
(3) FS-QRE	.9966	.9985	1					
(4) BO-QRE	.9957	.9972	.9997	1				
(5) CR-QRE	.9915	.9901	.9915	.9929	1			
(6) 7S	.9908	.9882	.9883	.9891	.9953	1		
(7) SUM	.9914	.9894	.9914	.9927	.9962	.9973	1	
(8) TTB	.9911	.9895	.9915	.9927	.9962	.9974	.9996	1

Correlations of model sets with a lower predictive error than the one of the best singular model are underlined.

Table 11
Competition Procedure: Mean Predictive Error Matrix for Binary Second Mover Model Sets

Model	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
(1) SPE	.0071							
(2) QRE	.0060	.0057						
(3) FS-QRE	.0060	.0055	.0056					
(4) BO-QRE	.0058	.0053	.0056	.0056				
(5) CR-QRE	.0059	.0050	.0052	.0053	.0067			
(6) 7S	.0044	.0036	.0036	.0036	.0047	.0043		
(7) SUM	.0042	.0035	.0036	.0037	.0045	.0037	.0038	
(8) TTB	.0041	.0035	.0037	.0037	.0045	.0038	.0038	.0038

Model sets with a lower mean predictive error than the one of the best singular model are underlined.

16 MATLAB Codes

16.1 spe.m

The MATLAB program spe.m computes the MSD of the estimation set and the MSD of the prediction set for the second mover subgame perfect equilibrium (SPE) model and the first mover SPE model (see Section 5.3.1). The visual output which is shown in the command window of MATLAB contains for each model the estimation MSD and the prediction MSD. Please ensure that qre_msd.m (the function that computes the MSD, see Appendix 16.2), disptable.m (the function that displays the results, see Appendix 16.3), ext_est_res.txt (a text file that contains the data of the estimation set, see Table 6) and ext_pred_res.txt (a text file that contains the data of the prediction set, see Table 7) are in the same folder. Function qre_msd.m is used as a shortcut for computing the MSD of a particular SPE model because the predictions of the quantal response equilibrium (QRE) are equal to the ones of SPE if the responsiveness to preferences λ_F and λ_S are infinite (see Section 5.3.2) (in the MATLAB code they are equal to 80 which is sufficiently high to approximate infinity).

Second Mover Model: MSD of the Estimation Set

```
clear all;                      % remove items from workspace
model=2;                        % select second mover model
p=80;                           % set lambdaS=80
data=load('ext_est_res.txt');   % load estimation set data
msd_est=qre_msd(p,model,data);  % compute msd
```

Second Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt');  % load prediction set data
msd_pre=qre_msd(p,model,data);  % compute msd
```

Second Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre],-4),...
    'msd_est|msd_pre','SPE_S')
```

```
       msd_est  msd_pre
SPE_S   0.0105   0.0071
```

First Mover Model: MSD of the Estimation Set

```
clear all;                      % remove items from workspace
model=1;                        % select first mover model
p=[80 80];                      % set lambdaS=80 and lambdaF=80
data=load('ext_est_res.txt');   % load estimation set data
msd_est=qre_msd(p,model,data);  % compute msd
```

First Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt');  % load predicition set data
msd_pre=qre_msd(p,model,data);  % compute msd
```

First Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre],-4),...
    'msd_est|msd_pre','SPE_F')

% Author: Wasilios Hariskos, November 2012 %
```

```
       msd_est  msd_pre
SPE_F   0.0545   0.0532
```

16.2 qre_msd.m

Function qre_msd.m computes for a given parameter estimate vector p the MSD of each quantal response equilibrium (QRE) model (see Section 5.3.2). The input arguments specify parameter estimate vector p, the model (first mover=1 or second mover=2) and the data (estimation set or prediction set). The function computes the MSD in three steps: For the 120 games, it stores first the payoff structure and the observed choice probabilities, then it computes the predicted choice probabilities for the specified model (second mover or first mover) and lastly it computes the mean of squared deviations (MSD) between observed and predicted choice probabilities.

```
function msd = qre_msd(p, model, data )
```

Experimental Data and Games

```
xF=data(:,[2 4 6]); % 120x3 payoffs of first mover F for each game
xS=data(:,[3 5 7]); % 120x3 payoffs of second mover S for each game
PI=data(:,8);       % 120x1 observed in probabilities for each game
PR=data(:,9);       % 120x1 observed right probabilities f.e.g.
```

Predictions of Second Mover Model

```
lambdaS=p(1);                % responsiveness to preferences of S
uS=xS;                       % 120x3 utilities for each outcome of each game
pr=exp(lambdaS*uS(:,3))... % 120x1 predicted right probabilities f.e.g.
    ./(exp(lambdaS*uS(:,3))+exp(lambdaS*uS(:,2)));
```

Predictions of First Mover Model

```
if model==1
lambdaF=p(2);                     % responsiveness to preferences of F
uFo=xF(:,1);                      % 120x1 out utilities for each game
euFi=(1-pr).*xF(:,2)+pr.*xF(:,3); % 120x1 expected in utilities f.e.g.
pi=exp(lambdaF*euFi)...           % 120x1 predicted in probabilities f.e.g.
    ./(exp(lambdaS*euFi)+exp(lambdaS*uFo));
end
```

MSD of Second Mover Model

```
if model==2


sdpr=(PR-pr).*(PR-pr); % 120x1 sq. dev.: obs. vs pred. right probabilities
msd=mean(sdpr);        % mean of 120x1 squared deviations
```

MSD of First Mover Model

```
elseif model==1
sdpi=(PI-pi).*(PI-pi); % 120x1 sq. dev.: obs. vs pred. in probabilities
msd=mean(sdpi);        % mean of 120x1 squared deviations
end

% Author: Wasilios Hariskos, November 2012 %
```

16.3 disptable.m

Function disptable.m displays a matrix with labeled columns and labeled rows in the MATLAB command window.

```
function disptable(M, col_strings, row_strings, fmt, spaces)
%DISPTABLE Displays a matrix with per-column or per-row labels.
%   DISPTABLE(M, COL_STRINGS, ROW_STRINGS)
%   Displays matrix or vector M with per-column or per-row labels,
%   specified in COL_STRINGS and ROW_STRINGS, respectively. These can be
%   cell arrays of strings, or strings delimited by the pipe character (|).
%   Either COL_STRINGS or ROW_STRINGS can be ommitted or empty.
%   DISPTABLE(M, COL_STRINGS, ROW_STRINGS, FMT, SPACES)
%   FMT is an optional format string or number of significant digits, as
%   used in NUM2STR. It can also be the string 'int', as a shorthand to
```

```
%   specify that the values should be displayed as integers.
%   SPACES is an optional number of spaces to separate columns, which
%   defaults to 1.
%   Example:
%     disptable(magic(3)*10-30, 'A|B|C', 'a|b|c')
%   Outputs:
%         A    B    C
%     a  50  -20   30
%     b   0   20   40
%     c  10   60  -10
%   Author: João F. Henriques, April 2010
%parse and validate inputs
if nargin < 2, col_strings = []; end
if nargin < 3, row_strings = []; end
if nargin < 4, fmt = 4; end
if nargin < 5, spaces = 2; end
if strcmp(fmt, 'int'),...
    fmt = '%.0f'; end  %shorthand for displaying integer values
assert(ndims(M) <= 2,...
'Can only display a vector or two-dimensional matrix.')
num_rows = size(M,1);
num_cols = size(M,2);
use_col_strings = true;
if ischar(col_strings),  %convert "|"-delimited string to
                     %cell array of strings
col_strings = textscan(col_strings, '%s', 'delimiter','|');
col_strings = col_strings{1};
elseif isempty(col_strings),  %empty input; have one empty string
                              %per column for consistency
col_strings = cell(num_cols,1);
use_col_strings = false;
elseif ~iscellstr(col_strings),
error...
('COL_STRINGS must be a cell array of strings, or a string with "|"');
end
use_row_strings = true;
if ischar(row_strings),  %convert "|"-delimited string to
                         %cell array of strings
row_strings = textscan(row_strings, '%s', 'delimiter','|');
row_strings = row_strings{1};
elseif isempty(row_strings),  %empty input; have one empty string
                              %per row for consistency
row_strings = cell(num_rows,1);
```

```
use_row_strings = false;
elseif ~iscellstr(row_strings),
error...
('ROW_STRINGS must be a cell array of strings, or a string with "|".');
end
assert(~use_col_strings || numel(col_strings) == num_cols, ...
'COL_STRINGS must have one string per column of M, or be empty.')
assert(~use_row_strings || numel(row_strings) == num_rows, ...
'ROW_STRINGS must have one string per column of M, or be empty.')
assert(isscalar(fmt) || (isvector(fmt) && ischar(fmt)), ...
'Format must be a format string or the # of significant digits (NUM2STR).')
%format the table for display
col_text = cell(num_cols,1);  %the text of each column
%spaces to separate columns
if use_col_strings,
blank_column = repmat(' ', num_rows + 1, spaces);
else
blank_column = repmat(' ', num_rows, spaces);
end
for col = 1:num_cols,
%convert this column of the matrix to its string representation
str = num2str(M(:,col), fmt);
%add the column header on top and automatically pad, returning a
%character array
if use_col_strings,
    str = char(col_strings{col}, str);
end
%right-justify and add blanks to separate from previous column
col_text{col} = [blank_column, strjust(str, 'right')];
end
%turn the row labels into a character array, with a blank line on top
if use_col_strings,
left_text = char('', row_strings{:});
else
left_text = char(row_strings{:});
end
%concatenate horizontally the character arrays and display
disp([left_text, col_text{:}])
disp(' ')
end
```

16.4 qre.m

The MATLAB program qre.m estimates the second mover quantal response equilibrium (QRE) model and the first mover QRE model (see Section 5.3.2) and computes the MSD of the estimation set and the MSD of the prediction set for each model. The visual output which is shown in the command window of MATLAB contains for each model the estimation MSD, the prediction MSD and the parameter estimate(s) that minimize(s) the estimation MSD. Please ensure that qre_msd.m (the function that computes the MSD, see Appendix 16.2), disptable.m (the function that displays the results, see Appendix 16.3), ext_est_res.txt (a text file that contains the data of the estimation set, see Table 6) and ext_pred_res.txt (a text file that contains the data of the prediction set, see Table 7) are in the same folder.

Second Mover Model: MSD of the Estimation Set

```
clear all;                         % remove items from workspace
data=load('ext_est_res.txt');      % load estimation set data
model=2;                           % select second mover model
p0=0;                              % set initial estimate p0 of lambdaS=0
lb=p0;                             % set lower bound lb of lambdaS=0
up=[];                             % set no upper bound up for lambdaS
% find the parameter estimate p=lambdaS that minimizes the msd of the
% estimation set (msd_est) by using the constrained nonliner multivariable
% function fmincon which calls function qre_msd (that computes the msd)
[p,msd_est]=fmincon(@qre_msd,p0,[],[],[],[],lb,up,[],optimset(...
'Display','off', 'LargeScale','off','Algorithm','active-set'),model,data);
```

Second Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt'); % load prediction set data
% compute the msd of the prediction set (msd_pre) based on the parameter
msd_pre=qre_msd(p,model,data); % estimate vector p that minimizes msd_est
```

Second Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre p(1)],-4),...
    'msd_est|msd_pre|lambdaS','QRE_S')
```

```
       msd_est  msd_pre  lambdaS
QRE_S   0.0092   0.0057    2.073
```

First Mover Model: MSD of the Estimation Set

```
clear all;                      % remove items from workspace
data=load('ext_est_res.txt');   % load estimation set data
model=1;                        % select first mover model
p0=[0 0];        % set initial estimate vector p0 = (lambdaS lambdaF) to 0
lb=p0;           % set lower bound vector lb equal to p0
up=[];           % set no upper bound vector up for (lambdaS lambdaF)
% find the parameter estimate vector p = (lambdaS lambdaF) that minimizes
% the msd of the estimation set (msd_est) by using function fmincon which
% calls function qre_msd
[p,msd_est]=fmincon(@qre_msd,p0,[],[],[],[],lb,up,[],optimset(...
'Display','off', 'LargeScale','off','Algorithm','active-set'),model,data);
```

First Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt'); % load prediction set data and compute the
% msd of the prediction set msd_pre (based on the parameter estimate
msd_pre=qre_msd(p,model,data); % vector p that minimizes msd_est)
```

First Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre p(1) p(2)],-4),...
    'msd_est|msd_pre|lambdaS|lambdaF','QRE_F')

% Author: Wasilios Hariskos, November 2012 %
```

```
       msd_est  msd_pre  lambdaS  lambdaF
QRE_F    0.017   0.0141    0.695   0.6823
```

16.5 qre_fs.m

The MATLAB program qre_fs.m estimates the second mover Fehr-Schmidt quantal response equilibrium (QRE) model and the first mover Fehr-Schmidt QRE model (see Section 5.3.3) and computes the MSD of the estimation set and the MSD of the prediction set for each model. The visual output which is shown in the command window of MATLAB contains for each model the estimation MSD, the prediction MSD and the parameter estimates that minimize the estimation MSD. Please ensure that qre_fs_msd.m (the function that computes the MSD, see Appendix 16.6), disptable.m (the function that displays the results, see Appendix 16.3), ext_est_res.txt (a text file that contains the data of the estimation set, see Table 6) and ext_pred_res.txt (a text file that contains the data of the prediction set, see Table 7) are in the same folder.

Second Mover Model: MSD of the Estimation Set

```
clear all;                          % remove items from workspace
data=load('ext_est_res.txt');   % load estimation set data
model=2;                            % select second mover model
% The parameter estimate vector of the second mover model is given by
% p = (lambdaS alphaS=betaS). The value of alphaS is constrained to the
% value of betaS since an unconstrained estimation yields a higher value
% for betaS than for alphaS.
p0=[0 0];                           % set initial values of p equal to 0
lb=p0;                              % set lower bounds of p equal to 0
up=[];                              % set no upper bound up for p
% find the parameter estimate vector p that minimizes the msd of the
% estimation set (msd_est) by using the constrained nonliner multivariable
% function fmincon which calls function qre_fs_msd (that computes the msd)
[p,msd_est]=fmincon(@qre_fs_msd,p0,[],[],[],[],lb,up,[],optimset(...
'Display','off', 'LargeScale','off','Algorithm','active-set'),model,data);
```

Second Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt'); % load prediction set data and compute the
% msd of the prediction set msd_pre (based on the parameter estimate vector
msd_pre=qre_fs_msd(p,model,data); % p that minimizes msd_est)
```

Second Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre p(1) p(2) p(2)],-4),...
    'msd_est|msd_pre|lambdaS|alphaS|betaS','QRE_FS_S')
```

	msd_est	msd_pre	lambdaS	alphaS	betaS
QRE_FS_S	0.0082	0.0056	2.299	0.0353	0.0353

First Mover Model: MSD of the Estimation Set

```
clear all;                        % remove items from workspace
data=load('ext_est_res.txt');     % load estimation set data
model=1;                          % select first mover model
% The parameter estimate vector of the first mover model is given by
% p = (lambdaS alphaS betaS lambdaF alphaF=betaF). The value of
% alphaF is constrained to the value of betaF since an unconstrained
% estimation yields a higher value for betaF than for alphaF.
p0=[0 0 0 0 0];        % set initial values of p equal to 0
lb=p0;          % set lower bounds of p equal to 0
up=[];          % set no upper bound for p
% find the parameter estimate vector p that minimizes the msd of the
% estimation set (msd_est) by using function fmincon and function qre_fs_msd
[p,msd_est]=fmincon(@qre_fs_msd,p0,[],[],[],[],lb,up,[],optimset(...
'Display','off', 'LargeScale','off','Algorithm','active-set'),model,data);
```

First Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt'); % load prediction set data and compute the
% msd of the prediction set msd_pre (based on the parameter estimate vector
msd_pre=qre_fs_msd(p,model,data); % p that minimizes msd_est)
```

First Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre p(1:5) p(5)],-4),...
'msd_est|msd_pre|lambdaS|alphaS|betaS|lambdaF|alphaF|betaF','QRE_FS_F')

% Author: Wasilios Hariskos, November 2012 %
```

	msd_est	msd_pre	lambdaS	alphaS	betaS	lambdaF	alphaF	betaF
QRE_FS_F	0.0118	0.014	0.8295	0.1494	0.1073	0.8142	0.0483	0.0483

16.6 qre_fs_msd.m

Function qre_fs_msd.m computes for a given parameter estimate vector p the MSD of each Fehr-Schmidt quantal response equilibrium (QRE) model (see Section 5.3.3). The input arguments specify parameter estimate vector p, the model (first mover=1 or second mover=2) and the data (estimation set or prediction set). The function computes the MSD in three steps: For the 120 games, it stores first the payoff structure and the observed choice probabilities, then it computes the predicted choice probabilities for the specified model (second mover or first mover) and lastly it computes the mean of squared deviations (MSD) between observed and predicted choice probabilities.

```
function msd = qre_fs_msd(p, model, data )
```

Experimental Data and Games

```
xF=data(:,[2 4 6]); % 120x3 payoffs of first mover F for each game
xS=data(:,[3 5 7]); % 120x3 payoffs of second mover S for each game
PI=data(:,8);       % 120x1 observed in probabilities for each game
PR=data(:,9);       % 120x1 observed right probabilities f.e.g.
```

Predictions for Second Mover

```
lambdaS=p(1);         % responsiveness to preferences of S
alphaS=p(2);          % degree of disadvantageous inequality aversion of S
betaS=p(2);           % degree of advantageous inequality aversion of S
if model==1           % do not constrain the value of beta_S to ...
    betaS=p(3);       % the value of beta_S for the first mover model
end
uS=xS...                  % 120x3 utilities for each outcome of each game
    -alphaS*max(0,xF-xS)...
    -betaS*max(0,xS-xF);
pr=exp(lambdaS*uS(:,3))...% 120x1 predicted right probabilities f.e.g.
    ./(exp(lambdaS*uS(:,3))+exp(lambdaS*uS(:,2)));
```

Predictions for First Mover

```
if model==1
lambdaF=p(4);         % responsiveness to preferences of F
alphaF=p(5);          % degree of disadvantageous inequality aversion of F
betaF=p(5);           % degree of advantageous inequality aversion of F
uF=xF...                          % 120x3 utilities for each game
    -alphaF*max(0,xS-xF)...
    -betaF*max(0,xF-xS);
euFi=(1-pr).*uF(:,2)+pr.*uF(:,3); % 120x1 expected in utilities f.e.g.
pi=exp(lambdaF*euFi)...           % 120x1 predicted in probabilities f.e.g.
    ./(exp(lambdaS*euFi)+exp(lambdaS*uF(:,1)));
end
```

MSD of Second Mover Model

```
if model==2

sdpr=(PR-pr).*(PR-pr); % 120x1 sq. dev.: obs. vs pred. right probabilities
msd=mean(sdpr);        % mean of 120x1 squared deviations
```

MSD of First Mover Model

```
elseif model==1
sdpi=(PI-pi).*(PI-pi); % 120x1 sq. dev.: obs. vs pred. in probabilities
msd=mean(sdpi);        % mean of 120x1 squared deviations
end

% Author: Wasilios Hariskos, November 2012 %
```

16.7 qre_fs_unc.m

The MATLAB program qre_fs_unc.m estimates the second mover Fehr-Schmidt quantal response equilibrium (QRE) model and the first mover Fehr-Schmidt QRE model (see Section 5.3.3) and computes the MSD of the estimation set and the MSD of the prediction set for each model. The visual output which is shown in the command window of MATLAB contains for each model the estimation MSD, the prediction MSD and the unconstrained parameter estimates that minimize the estimation MSD. Please ensure that qre_fs_unc_msd.m (the function that computes the MSD, see Appendix 16.8), disptable.m (the function that displays the results, see Appendix 16.3), ext_est_res.txt (a text file that contains the data of the estimation set, see Table 6) and ext_pred_res.txt (a text file that contains the data of the prediction set, see Table 7) are in the same folder.

Second Mover Model: MSD of the Estimation Set

```
clear all;                      % remove items from workspace
data=load('ext_est_res.txt');   % load estimation set data
model=2;                        % select second mover model
% The parameter estimate vector of the second mover model is given by
% p = (lambdaS alphaS=betaS). The value of alphaS is constrained to the
% value of betaS since an unconstrained estimation yields a higher value
% for betaS than for alphaS.
p0=[0 0];                       % set initial values of p equal to 0
lb=p0;                          % set lower bounds of p equal to 0
up=[];                          % set no upper bound up for p
```

```
% find the parameter estimate vector p that minimizes the msd of the
% estimation set (msd_est) by using the constrained nonliner multivariable
% function fmincon which calls function qre_fs_msd (that computes the msd)
[p,msd_est]=fmincon(@qre_fs_msd,p0,[],[],[],[],lb,up,[],optimset(...
'Display','off', 'LargeScale','off','Algorithm','active-set'),model,data);
```

Second Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt'); % load prediction set data and compute the
% msd of the prediction set msd_pre (based on the parameter estimate vector
msd_pre=qre_fs_msd(p,model,data); % p that minimizes msd_est)
```

Second Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre p(1) p(2) p(2)],-4),...
    'msd_est|msd_pre|lambdaS|alphaS|betaS','QRE_FS_S')
```

```
          msd_est  msd_pre  lambdaS  alphaS   betaS
QRE_FS_S   0.0082   0.0056    2.299  0.0353  0.0353
```

First Mover Model: MSD of the Estimation Set

```
clear all;                      % remove items from workspace
data=load('ext_est_res.txt');   % load estimation set data
model=1;                        % select first mover model
% The parameter estimate vector of the first mover model is given by
% p = (lambdaS alphaS betaS lambdaF alphaF=betaF). The value of
% alphaF is constrained to the value of betaF since an unconstrained
% estimation yields a higher value for betaF than for alphaF.
p0=[0 0 0 0 0];       % set initial values of p equal to 0
lb=p0;          % set lower bounds of p equal to 0
up=[];          % set no upper bound for p
% find the parameter estimate vector p that minimizes the msd of the
% estimation set (msd_est) by using function fmincon and function qre_fs_msd
[p,msd_est]=fmincon(@qre_fs_msd,p0,[],[],[],[],lb,up,[],optimset(...
'Display','off', 'LargeScale','off','Algorithm','active-set'),model,data);
```

First Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt'); % load prediction set data and compute the
% msd of the prediction set msd_pre (based on the parameter estimate vector
msd_pre=qre_fs_msd(p,model,data); % p that minimizes msd_est)
```

First Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre p(1:5) p(5)],-4),...
'msd_est|msd_pre|lambdaS|alphaS|betaS|lambdaF|alphaF|betaF','QRE_FS_F')

% Author: Wasilios Hariskos, November 2012 %
```

	msd_est	msd_pre	lambdaS	alphaS	betaS	lambdaF	alphaF	betaF
QRE_FS_F	0.0118	0.014	0.8295	0.1494	0.1073	0.8142	0.0483	0.0483

16.8 qre_fs_unc_msd.m

Function qre_fs_unc_msd.m computes for a given unconstrained parameter estimate vector p the MSD of each Fehr-Schmidt quantal response equilibrium (QRE) model (see Section 5.3.3). The input arguments specify parameter estimate vector p, the model (first mover=1 or second mover=2) and the data (estimation set or prediction set). The function computes the MSD in three steps: For the 120 games, it stores first the payoff structure and the observed choice probabilities, then it computes the predicted choice probabilities for the specified model (second mover or first mover) and lastly it computes the mean of squared deviations (MSD) between observed and predicted choice probabilities.

```
function msd = qre_fs_unc_msd(p, model, data )
```

Experimental Data and Games

```
xF=data(:,[2 4 6]); % 120x3 payoffs of first mover F for each game
xS=data(:,[3 5 7]); % 120x3 payoffs of second mover S for each game
PI=data(:,8);       % 120x1 observed in probabilities for each game
PR=data(:,9);       % 120x1 observed right probabilities f.e.g.
```

Predictions for Second Mover

```
lambdaS=p(1);          % responsiveness to preferences of S
alphaS=p(2);           % degree of disadvantageous inequality aversion of S
betaS=p(3);% degree of advantageous inequality aversion of S; no constrain!
uS=xS...                   % 120x3 utilities for each outcome of each game
    -alphaS*max(0,xF-xS)...
    -betaS*max(0,xS-xF);
pr=exp(lambdaS*uS(:,3))...% 120x1 predicted right probabilities f.e.g.
    ./(exp(lambdaS*uS(:,3))+exp(lambdaS*uS(:,2)));
```

Predictions for First Mover

```
if model==1
lambdaF=p(4);          % responsiveness to preferences of F
alphaF=p(5);           % degree of disadvantageous inequality aversion of F
betaF=p(6);% degree of advantageous inequality aversion of F; no constrain!
uF=xF...                           % 120x3 utilities for each game
    -alphaF*max(0,xS-xF)...
    -betaF*max(0,xF-xS);
euFi=(1-pr).*uF(:,2)+pr.*uF(:,3); % 120x1 expected in utilities f.e.g.
pi=exp(lambdaF*euFi)...            % 120x1 predicted in probabilities f.e.g.
    ./(exp(lambdaS*euFi)+exp(lambdaS*uF(:,1)));
end
```

MSD of Second Mover Model

```
if model==2

sdpr=(PR-pr).*(PR-pr); % 120x1 sq. dev.: obs. vs pred. right probabilities
msd=mean(sdpr);        % mean of 120x1 squared deviations
```

MSD of First Mover Model

```
elseif model==1
sdpi=(PI-pi).*(PI-pi); % 120x1 sq. dev.: obs. vs pred. in probabilities
msd=mean(sdpi);        % mean of 120x1 squared deviations
end

% Author: Wasilios Hariskos, November 2012 %
```

16.9 qre_bo.m

The MATLAB program qre_bo.m estimates the second mover Bolton-Ockenfels quantal response equilibrium (QRE) model and the first mover Bolton-Ockenfels QRE model (see Section 5.3.4) and computes the MSD of the estimation set and the MSD of the prediction set for each model. The visual output which is shown in the command window of MATLAB contains for each model the estimation MSD, the prediction MSD and the parameter estimates that minimize the estimation MSD. Please ensure that qre_bo_msd.m (the function that computes the MSD, see Appendix 16.10), disptable.m (the function that displays the results, see Appendix 16.3), ext_est_res.txt (a text file that contains the data of the estimation set, see Table 6) and ext_pred_res.txt (a text file that contains the data of the prediction set, see Table 7) are in the same folder.

Second Mover Model: MSD of the Estimation Set

```
clear all;                        % remove items from workspace
data=load('ext_est_res.txt');   % load estimation set data
model=2;                          % select second mover model
% The parameter estimate vector of the second mover model is given by
% p = (lambdaS bS).
p0=[0 0];                         % set initial values of p equal to 0
lb=p0;                            % set lower bounds of p equal to 0
up=[];                            % set no upper bound up for p
% find the parameter estimate vector p that minimizes the msd of the
% estimation set (msd_est) by using the constrained nonliner multivariable
% function fmincon which calls function qre_bo_msd (that computes the msd)
[p,msd_est]=fmincon(@qre_bo_msd,p0,[],[],[],[],lb,up,[],optimset(...
'Display','off', 'LargeScale','off','Algorithm','active-set'),model,data);
```

Second Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt'); % load prediction set data and compute the
% msd of the prediction set msd_pre (based on the parameter estimate vector
msd_pre=qre_bo_msd(p,model,data); % p that minimizes msd_est)
```

Second Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre p(1) p(2)],-4),...
    'msd_est|msd_pre|lambdaS|bS','QRE_BO_S')
```

```
          msd_est  msd_pre  lambdaS      bS
```

```
QRE_BO_S   0.0073   0.0056    2.361  0.3779
```

First Mover Model: MSD of the Estimation Set

```
clear all;                        % remove items from workspace
data=load('ext_est_res.txt');    % load estimation set data
model=1;                          % select first mover model
% The parameter estimate vector of the first mover model is given by
% p = (lambdaS bS lambdaF bF).
p0=[0 0 0 0];   % set initial values of p equal to 0
lb=p0;          % set lower bounds of p equal to 0
up=[];          % set no upper bound for p
% find the parameter estimate vector p that minimizes the msd of the
% estimation set (msd_est) by using function fmincon and function qre_bo_msd
[p,msd_est]=fmincon(@qre_bo_msd,p0,[],[],[],[],lb,up,[],optimset(...
'Display','off', 'LargeScale','off','Algorithm','active-set'),model,data);
```

First Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt'); % load prediction set data and compute the
% msd of the prediction set msd_pre (based on the parameter estimate vector
msd_pre=qre_bo_msd(p,model,data); % p that minimizes msd_est)
```

First Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre p(1) p(2) p(3) p(4)],-4),...
'msd_est|msd_pre|lambdaS|bS|lambdaF|bF','QRE_BO_F')

% Author: Wasilios Hariskos, November 2012 %
```

```
          msd_est  msd_pre  lambdaS      bS  lambdaF      bF
QRE_BO_F   0.0141   0.0172   0.7783  0.9214    0.773  0.5413
```

16.10 qre_bo_msd.m

Function qre_bo_msd.m computes for a given parameter estimate vector p the MSD of each Bolton-Ockenfels quantal response equilibrium (QRE) model (see Section 5.3.4). The input arguments specify parameter estimate vector p, the model (first mover=1 or second mover=2) and the data (estimation set or prediction set). The function computes the MSD in three steps: For the 120 games, it stores first the payoff structure and the observed choice probabilities, then it computes the predicted choice probabilities for the specified model (second mover or first mover) and lastly it computes the mean of squared deviations (MSD) between observed and predicted choice probabilities.

```
function msd = qre_bo_msd(p, model, data )
```

Experimental Data and Games

```
xF=data(:,[2 4 6]); % 120x3 payoffs of first mover F for each game
xS=data(:,[3 5 7]); % 120x3 payoffs of second mover S for each game
PI=data(:,8);       % 120x1 observed in probabilities for each game
PR=data(:,9);       % 120x1 observed right probabilities f.e.g.
```

Payoff Transformation For each game ...

```
xmin=min([xS xF],[],2);      % store 120x1 minimum payoffs,
xF=xF+repmat(abs(xmin),1,3); % add abs. minimum payoff to each payoff of F,
xS=xS+repmat(abs(xmin),1,3); % add abs. minimum payoff to each payoff of S,
c=xF+xS;                     % compute the cake size c,
epsilon=0.0000000000000001;  % define epsilon as a very small number,
c=c+epsilon;                 % add epsilon to c to avoid division by 0,
sigmaS=xS./c;                % compute proportion of c that S receives and
sigmaF=xF./c;                % compute proportion of c that F receives.
```

Predictions for Second Mover

```
lambdaS=p(1);      % responsiveness to preferences of S
bS=p(2);           % degree of aversion of S to deviations from equal split
for g=1:120        % 120x3 utilities for each outcome of each game
for k=1:3
uS(g,k)=c(g,k)*sigmaS(g,k) - bS*0.5*c(g,k)*((sigmaS(g,k)-0.5)^2);
end
end
pr=exp(lambdaS*uS(:,3))...% 120x1 predicted right probabilities f.e.g.
    ./(exp(lambdaS*uS(:,3))+exp(lambdaS*uS(:,2)));
```

Predictions for First Mover

```
if model==1
lambdaF=p(3);      % responsiveness to preferences of F
bF=p(4);           % degree of aversion of F to deviations from equal split
for g=1:120        % 120x3 utilities for each game
for k=1:3
uF(g,k)=c(g,k)*sigmaF(g,k) - bF*0.5*c(g,k)*((sigmaF(g,k)-0.5)^2);
end
end
euFi=(1-pr).*uF(:,2)+pr.*uF(:,3); % 120x1 expected in utilities f.e.g.
pi=exp(lambdaF*euFi)...           % 120x1 predicted in probabilities f.e.g.
    ./(exp(lambdaS*euFi)+exp(lambdaS*uF(:,1)));
end
```

MSD of Second Mover Model

```
if model==2

sdpr=(PR-pr).*(PR-pr); % 120x1 sq. dev.: obs. vs pred. right probabilities
msd=mean(sdpr);        % mean of 120x1 squared deviations
```

MSD of First Mover Model

```
elseif model==1
sdpi=(PI-pi).*(PI-pi); % 120x1 sq. dev.: obs. vs pred. in probabilities
msd=mean(sdpi);        % mean of 120x1 squared deviations
end

% Author: Wasilios Hariskos, November 2012 %
```

16.11 qre_cr.m

The MATLAB program qre_cr.m estimates the second mover Charness-Rabin quantal response equilibrium (QRE) model and the first mover Charness-Rabin QRE model (see Section 5.3.5) and computes the MSD of the estimation set and the MSD of the prediction set for each model. The visual output which is shown in the command window of MATLAB contains for each model the estimation MSD, the prediction MSD and the parameter estimates that minimize the estimation MSD. Please ensure that qre_cr_msd.m (the function that computes the MSD, see Appendix 16.12), disptable.m (the function that displays the results, see Appendix 16.3), ext_est_res.txt (a text file that contains the data of the estimation set, see Table 6) and ext_pred_res.txt (a text file that contains the data of the prediction set, see Table 7) are in the same folder.

Second Mover Model: MSD of the Estimation Set

```
clear all;                      % remove items from workspace
data=load('ext_est_res.txt');   % load estimation set data
model=2;                        % select second mover model
% The parameter estimate vector of the second mover model is given by
% p = (lambdaS rhoS sigmaS thetaS).
p0=[0 0 0 0];                   % set initial values of p equal to 0
lb=[ 0 -1 -1 0];                % set lower bounds of p
up=[80  1  1 1];                % set no upper bound up for p
% find the parameter estimate vector p that minimizes the msd of the
% estimation set (msd_est) by using the constrained nonliner multivariable
% function fmincon which calls function qre_cr_msd (that computes the msd)
[p,msd_est]=fmincon(@qre_cr_msd,p0,[],[],[],[],lb,up,[],optimset(...
'Display','off', 'LargeScale','off','Algorithm','active-set'),model,data);
```

Second Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt'); % load prediction set data and compute the
% msd of the prediction set msd_pre (based on the parameter estimate vector
msd_pre=qre_cr_msd(p,model,data); % p that minimizes msd_est)
```

Second Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre p(1) p(2) p(3) p(4)],-4),...
    'msd_est|msd_pre|lambdaS|rhoS|sigmaS|thetaS','QRE_CR_S')
```

	msd_est	msd_pre	lambdaS	rhoS	sigmaS	thetaS
QRE_CR_S	0.0042	0.0067	3.437	0.0758	0.0232	0

First Mover Model: MSD of the Estimation Set

```
clear all;                      % remove items from workspace
data=load('ext_est_res.txt');   % load estimation set data
model=1;                        % select first mover model
% The parameter estimate vector of the first mover model is given by
% p = (lambdaS rhoS sigmaS thetaS lambdaF rhoF sigmaF).
p0=[ 0  0  0  0  0  0  0];       % set initial values of p equal to 0
lb=[ 0 -1 -1  0  0 -1 -1];       % set lower bounds of p equal to 0
```

```
up=[80  1  1  1 80  1  1];       % set no upper bound for p
% find the parameter estimate vector p that minimizes the msd of the
% estimation set (msd_est) by using function fmincon and function qre_cr_msd
[p,msd_est]=fmincon(@qre_cr_msd,p0,[],[],[],[],lb,up,[],optimset(...
'Display','off', 'LargeScale','off','Algorithm','active-set'),model,data);
```

First Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt'); % load prediction set data and compute the
% msd of the prediction set msd_pre (based on the parameter estimate vector
msd_pre=qre_cr_msd(p,model,data); % p that minimizes msd_est)
```

First Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre p(1) p(2) p(3) p(4) p(5) p(6) p(7)],-4),...
'msd_est|msd_pre|lambdaS|rhoS|sigmaS|thetaS|lambda_F|rhoF|sigmaF','QRE_CR_F')

% Author: Wasilios Hariskos, November 2012 %
```

	msd_est	msd_pre	lambdaS	rhoS	sigmaS	thetaS	lambda_F	rhoF	sigmaF
QRE_CR_F	0.0112	0.0143	0.8746	0.0819	-0.134	0.0016	0.8585	0.0815	-0.0143

16.12 qre_cr_msd.m

Function qre_cr_msd.m computes for a given parameter estimate vector p the MSD of each Charness-Rabin quantal response equilibrium (QRE) model (see Section 5.3.5). The input arguments specify parameter estimate vector p, the model (first mover=1 or second mover=2) and the data (estimation set or prediction set). The function computes the MSD in three steps: For the 120 games, it stores first the payoff structure and the observed choice probabilities, then it computes the predicted choice probabilities for the specified model (second mover or first mover) and lastly it computes the mean of squared deviations (MSD) between observed and predicted choice probabilities.

```
function msd = qre_cr_msd(p, model, data )
```

Experimental Data and Games

```
xF=data(:,[2 4 6]); % 120x3 payoffs of first mover F for each game
xS=data(:,[3 5 7]); % 120x3 payoffs of second mover S for each game
PI=data(:,8);       % 120x1 observed in probabilities for each game
PR=data(:,9);       % 120x1 observed right probabilities f.e.g.
```

Predictions for Second Mover

```
lambdaS=p(1);          % responsiveness to preferences of S
rhoS=p(2);             % weight of other payoff if S is better off
sigmaS=p(3);           % weight of other payoff if S is worse off
thetaS=p(4);           % weight of other payoff if F misbehaved
q=zeros(120,1);        % indicates misbehavior of F
r=zeros(120,3);        % indivates for each outcome if S is better off
s=zeros(120,3);        % indicates for each outcome if S is worse off
for g=1:120            % check for each game if F misbehaved
if xS(g,1)>max(xS(g,2),xS(g,3)) && ...
xF(g,1)+xS(g,1)>max(xF(g,2)+xS(g,2),xF(g,3)+xS(g,3))
q(g,1)=-1;             % if true set value of q to -1
end
for k=1:3
if xS(g,k)>xF(g,k)     % check for each outcome of a game if S is better off
r(g,k)=1;              % if true set value of r to 1
end
if xS(g,k)<xF(g,k)     % check for each outcome of a game if S is worse off
s(g,k)=1;              % if true set value of s to 1
end
uS(g,k)=...            % compute for each outcome of game the utiliy of S
        xS(g,k)*(1-rhoS*r(g,k)-sigmaS*s(g,k)-thetaS*q(g,1)) + ...
        xF(g,k)*(rhoS*r(g,k)+sigmaS*s(g,k)+thetaS*q(g,1));
end
end
pr=exp(lambdaS*uS(:,3))...% 120x1 predicted right probabilities f.e.g.
    ./(exp(lambdaS*uS(:,3))+exp(lambdaS*uS(:,2)));
```

Predictions for First Mover

```
if model==1
lambdaF=p(5);         % responsiveness to preferences of F
rhoF=p(6);            % weight of other payoff if S is worse off
sigmaF=p(7);          % weight of other payoff if S is better off
uF=xF.*(1-rhoF*s-sigmaF*r)...       % 120x3 utilities for each game
    + xS.*(rhoF*s+sigmaF*r);
euFi=(1-pr).*uF(:,2)+pr.*uF(:,3); % 120x1 expected in utilities f.e.g.
pi=exp(lambdaF*euFi)...           % 120x1 predicted in probabilities f.e.g.
    ./(exp(lambdaS*euFi)+exp(lambdaS*uF(:,1)));
end
```

MSD of Second Mover Model

```
if model==2

sdpr=(PR-pr).*(PR-pr); % 120x1 sq. dev.: obs. vs pred. right probabilities
msd=mean(sdpr);        % mean of 120x1 squared deviations
```

MSD of First Mover Model

```
elseif model==1
sdpi=(PI-pi).*(PI-pi); % 120x1 sq. dev.: obs. vs pred. in probabilities
msd=mean(sdpi);        % mean of 120x1 squared deviations
end

% Author: Wasilios Hariskos, November 2012 %
```

16.13 seven_strategies.m

The MATLAB program seven_strategies.m estimates the second mover seven strategies (7S) model and the first mover 7S model (see Section 5.3.6) and computes the MSD of the estimation set and the MSD of the prediction set for each model. The visual output which is shown in the command window of MATLAB contains for each model the estimation MSD, the prediction MSD and the parameter estimates that minimize the estimation MSD. Please ensure that disptable.m (the function that displays the results, see Appendix 16.3), ext_est_res.txt (a text file that contains the data of the estimation set, see Table 6), ext_pred_res.txt (a text file that contains the data of the prediction set, see Table 7), pred_s.m (the function that computes for each game the predictions of the second mover model, see Appendix 16.14) and pred_f.m (the function that computes for each game the predictions of first mover model, see Appendix 16.15) are in the same folder.

Second Mover Model: MSD of the Estimation Set

```
clear all;                        % remove items from workspace
data=load('ext_est_res.txt');    % load estimation set data
PR=data(:,9);       % 120x1 observed right probabilities f.e.g.
% compute predictions of each strategy and game and store them in a 120x5
% matrix. Use function pred_s.m that uses the data as input and returns for
% each game the predictions of each strategy
p=pred_s(data); % matrix with 5 columns = pR pN pJ pW pD & 120 rows = games
beta=regress(PR,p); % find optimal probability distribution over strategies
% beta=regress(PR,p) returns a 5-by-1 beta vector of coefficient estimates
% for a multilinear regression of the responses in PR on the predictors in
% p; p is an 120-by-5 matrix of 5 predictors (the predictions of the five
```

```
% strategies Ratio, NiceR, JointMx, MxWeak, MnDiff)) at each of 120
% observations (the games); PR is an 120-by-1 vector of observed responses
% (the observed right choice probabilities).
beta=beta'/sum(beta); % beta = (betaR betaN betaJ betaW betaD) is normalized
% so that the sum of the coefficient estimates = 1
pr=sum(p.*repmat(beta,120,1),2); % 120x1 predicted overall right choice
% probabilities for each game
sdpr=(PR-pr).*(PR-pr); % 120x1 sq. dev.: obs. vs pred. right probabilities
msd_est=mean(sdpr);  % mean of 120x1 squared deviations
```

Second Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt'); % load prediction set data
PR=data(:,9);       % 120x1 observed right probabilities f.e.g.
%compute predictions of each strategy and game and store them in a 120x5
p=pred_s(data); % matrix with 5 columns = pR pN pJ pW pD & 120 rows = games
% compute predicted probability to observe a right choice for each game
pr=sum(p.*repmat(beta,120,1),2);
sdpr=(PR-pr).*(PR-pr); % 120x1 sq. dev.: obs. vs pred. right probabilities
msd_pre=mean(sdpr);  % mean of 120x1 squared deviations
```

Second Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre beta],-4),...
    'msd_est|msd_pre|betaR|betaN|betaJ|betaW|betaD','7S_S')
```

	msd_est	msd_pre	betaR	betaN	betaJ	betaW	betaD
7S_S	0.0029	0.0043	0.5038	0.3565	0.0581	0.0445	0.0371

First Mover Model: MSD of the Estimation Set

```
clear all;                      % remove items from workspace
data=load('ext_est_res.txt');   % load estimation set data
PI=data(:,8);       % 120x1 observed in probabilities for each game
%compute predictions of each strategy and game and store them in a 120x6
p=pred_f(data); % matrix with 6 cols = pR pL pM pJ pW pD & 120 rows = games
alpha=regress(PI,p); % find optimal probability distribution over strategies
alpha=alpha'/sum(alpha); %alpha=(alphaR alphaL alphaM alphaJ alphaW alphaD)
% compute predicted probability to observe an in choice for each game
```

```
pi=sum(p.*repmat(alpha,120,1),2); % 120x1 in choice probabilities
sdpi=(PI-pi).*(PI-pi); % 120x1 sq. dev.: obs. vs pred. in probabilities
msd_est=mean(sdpi);  % mean of 120x1 squared deviations
```

First Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt'); % load prediction set data
PI=data(:,8);      % 120x1 observed in probabilities for each game
%compute predictions of each strategy and game and store them in a 120x6
p=pred_f(data); % matrix with 6 cols = pR pL pM pJ pW pD & 120 rows = games
% compute predicted probability to observe an in choice for each game
pi=sum(p.*repmat(alpha,120,1),2); % 120x1 in choice probabilities
sdpi=(PI-pi).*(PI-pi); % 120x1 sq. dev.: obs. vs pred. in probabilities
msd_pre=mean(sdpi);  % mean of 120x1 squared deviations
```

First Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre alpha],-4),...
    'msd_est|msd_pre|alphaR|alphaL|alphaM|alphaJ|alphaW|alphaD','7S_F')

% Author: Wasilios Hariskos, November 2012 %
```

	msd_est	msd_pre	alphaR	alphaL	alphaM	alphaJ	alphaW	alphaD
7S_F	0.0119	0.0083	0.4348	0.195	0.2005	0.0665	0.0619	0.0414

16.14 pred_s.m

Function pred_s.m reads the data of the estimation set or the prediction set as input and computes for each of the five strategies of the second mover seven strategies model[35] the predicted right choice probability for each of the 120 games. As an output it returns a 120-by-5 vector p. Notice that the model that was implemented in the code that was posted on the competition homepage[36] deviates from the description of the model in the introductory paper of Ert et al. (2011). However, this code is based only on the model as it is described by Ert et al. (2011). In comparison to this code, the one that was published on the competition homepage specifies that an individual that applies $MnDiff$ chooses $left$ if she is indifferent between $left$ and $right$ instead of choosing randomly. The part where this code deviates from the one on the prediction homepage is marked with an * below. Notice moreover that

[35] Three of the seven strategies yield the same predictions, therefore the seven strategies reduce to five strategies.
[36] `https://sites.google.com/site/extformpredcomp/baseline-models/seven-strategies-matlab`.

the optimal distribution over strategies that is estimated via regression analysis by program seven_strategies.m (see Appendix 16.13) differs as a consequence too.

```
function p=pred_s(data)
xF=data(:,[2 4 6]); % 120x3 payoffs of first mover F for each game
xS=data(:,[3 5 7]); % 120x3 payoffs of second mover S for each game
% Right choice probabality vector for stategy ...
pR=zeros(120,1); % Ratio,
pN=zeros(120,1); % NiceR,
pJ=zeros(120,1); % JointMx,
pW=zeros(120,1); % MxWeak,
pD=zeros(120,1); % MnDiff are filled with 120 zeros, one for each game,
                 % i.e., the default option is to choose left
left=2;  % second column of xS contains for each game the payoff of left
right=3; % third column of xS contains for each game the payoff of right
% Check for each game and strategy if right choice probability is 0.5 or 1
for game=1:120
    % Ratio
    if xS(game,left) < xS(game,right)
            pR(game)=1; % right choice
    end
    if xS(game,left) == xS(game,right)
            pR(game)=0.5; % random choice
    end
    % JointMx
    if xS(game,left)+xF(game,left) < xS(game,right)+xF(game,right)
            pJ(game)=1; % right choice
    end
    if xS(game,left)+xF(game,left)==xS(game,right)+xF(game,right)
            pJ(game)=0.5; % random choice
    end
    % NiceR
    if xS(game,left) < xS(game,right)
            pN(game)=1; % right choice
    end
    if xS(game,left) == xS(game,right)
            pN(game)=pJ(game); % JointMx choice
    end
    % MxWeak
    if min(xS(game,left),xF(game,left)) < min(xS(game,right),xF(game,right))
            pW(game)=1; % right choice
    end
    if min(xS(game,left),xF(game,left))==min(xS(game,right),xF(game,right))
```

```
            pW(game)=0.5; % random choice
    end
    % MnDiff
    if abs(xS(game,left)-xF(game,left)) > abs(xS(game,right)-xF(game,right))
            pD(game)=1; % right choice
    end
    if abs(xS(game,left)-xF(game,left))==abs(xS(game,right)-xF(game,right))
            pD(game)=0.5; % random choice (* not 0, see comment above)
    end
end
p=[pR pN pJ pW pD]; % save prediction of each strategy in a 120x5 matrix
end

% Author: Wasilios Hariskos, November 2012 %
```

16.15 pred_f.m

Function pred_f.m reads the data of the estimation set or the prediction set as input and computes for each of the six strategies of the first mover seven strategies model[37] the predicted in choice probability for each of the 120 games. As an output it returns a 120-by-6 vector p. Notice that the model that was implemented in the code that was posted on the competition homepage[38] deviates from the description of the model in the introductory paper of Ert et al. (2011). However, this code is based only on the model as it is described by Ert et al. (2011). In comparison to this code, the one that was published on the competition homepage specifies that for (1) $MnDiff$ the individual chooses *out* if she is indifferent between *in* and *out* instead of choosing randomly and that she compares to the maximum and not to the minimum of the two absolute differences implied by the *in* choice to the absolute difference implied by the *out* choice; (2) for $JointMx$ and $Level-1$ that she chooses *out* if she is indifferent between *in* and *out* instead of choosing randomly; and (3) for $MaxMin$ that she chooses *in* if she is indifferent between *in* an *out* instead of choosing randomly. The parts where this code deviates from the one on the prediction homepage are marked with an * below. Notice moreover that the optimal distribution over strategies that is estimated via regression analysis by program seven_strategies.m (see Appendix 16.13) differs as a consequence too.

```
function p=pred_f(data)
xF=data(:,[2 4 6]); % 120x3 payoffs of first mover F for each game
xS=data(:,[3 5 7]); % 120x3 payoffs of second mover S for each game
% Right choice probabality vector for stategy ...
pRS=zeros(120,1); % Ratio Second Mover (necessary for Ratio First Mover),
pR=zeros(120,1); % Ratio First Mover,
```

[37] One of the seven strategies cannot be applied to the first mover choice problem, therefore the seven strategies reduce to six strategies.

[38] https://sites.google.com/site/extformpredcomp/baseline-models/seven-strategies-matlab.

```
pL=zeros(120,1); % Level-1,
pM=zeros(120,1); % MaxMin,
pJ=zeros(120,1); % JointMx,
pW=zeros(120,1); % MxWeak,
pD=zeros(120,1); % MnDiff are filled with 120 zeros, one for each game,
                 % i.e., the default option is to choose out
out=1; % first column of xS or xF contains for each game the payoff of left
left=2;% second column of xS or xF contains for each game the payoff of left
right=3;% third column of xS or xF contains for each game the payoff of right
% Check for each game and strategy if right choice probability is 0.5 or 1
for game=1:120
    % Ratio Second Mover
    if xS(game,left) < xS(game,right)
            pRS(game)=1; % in choice
    end
    if xS(game,left) == xS(game,right)
            pRS(game)=0.5; % random choice
    end
    % Ratio First Mover
    if xF(game,out) < (1-pRS(game))*xF(game,left)+pRS(game)*xF(game,right)
            pR(game)=1; % in choice
    end
    if xF(game,out) == (1-pRS(game))*xF(game,left)+pRS(game)*xF(game,right)
            pR(game)=0.5; % random choice
    end
    % Level-1
    if xF(game,out) < 1/2*xF(game,left)+1/2*xF(game,right)
            pL(game)=1; % in choice
    end
    if xF(game,out) == 1/2*xF(game,left)+1/2*xF(game,right)
            pL(game)=.5; % random choice (* not 0, see comment above)
    end
    % MaxMin
    if xF(game,out) < min(xF(game,left),xF(game,right))
            pM(game)=1; % in choice
    end
    if xF(game,out) == min(xF(game,left),xF(game,right))
            pM(game)=.5; % random choice (* not 1, see comment above)
    end
    % JointMx
    if xF(game,out)+xS(game,out) <...
       max(xF(game,left)+xS(game,left),xF(game,right)+xS(game,right))
            pJ(game)=1; % in choice
```

```
    end
    if xF(game,out)+xS(game,out) ==...
       max(xF(game,left)+xS(game,left),xF(game,right)+xS(game,right))
               pJ(game)=0.5; % random choice (* not 0, see comment above)
    end
    % MxWeak
    if min(xF(game,out),xS(game,out)) <...
    max(min(xF(game,left),xS(game,left)),min(xF(game,right),xS(game,right)))
            pW(game)=1; % in choice
    end
    if min(xF(game,out),xS(game,out)) ==...
    max(min(xF(game,left),xS(game,left)),min(xF(game,right),xS(game,right)))
            pW(game)=0.5; % random choice
    end
    % MnDiff
    if abs(xF(game,out)-xS(game,out)) >...
    min(abs(xF(game,left)-xS(game,left)),abs(xF(game,right)-xS(game,right)))
    % (* not max, see comment above)
            pD(game)=1; % in choice
    end
    if abs(xF(game,out)-xS(game,out)) ==...
    min(abs(xF(game,left)-xS(game,left)),abs(xF(game,right)-xS(game,right)))
    % (* not max, see comment above)
            pD(game)=0.5; % random choice (* not 0, see comment above)
    end
end
p=[pR pL pM pJ pW pD]; % save prediction of each strategy in a 120x6 matrix
end

% Author: Wasilios Hariskos, November 2012 %
```

16.16 sum_ose.m

The MATLAB program sum_ose.m estimates the second mover stochastic utility maximizer (SUM) model with own welfare, social welfare, and equality (OSE) preferences (see Section 5.4.1) and computes the MSD of the estimation set and the MSD of the prediction set. The visual output which is shown in the command window of MATLAB contains the estimation MSD, the prediction MSD and the parameter estimates that minimize the estimation MSD. Please ensure that sum_ose_msd.m (the function that computes the MSD, see Appendix 16.17), disptable.m (the function that displays the results, see Appendix 16.3), ext_est_res.txt (a text file that contains the data of the estimation set, see Table 6) and ext_pred_res.txt (a text file that contains the data of the prediction set, see Table 7) are in the same folder.

Second Mover Model: MSD of the Estimation Set

```
clear all;                        % remove items from workspace
data=load('ext_est_res.txt');   % load estimation set data
% The parameter estimate vector of the second mover model is given by
% p = (lambdaS alpha beta).
p0=[0 0 0];                       % set initial values of p equal to 0
lb=p0;                            % set lower bounds of p equal to 0
up=[];                            % set no upper bound up for p
% find the parameter estimate vector p that minimizes the msd of the
% estimation set (msd_est) by using the constrained nonliner multivariable
% function fmincon which calls function sum_ose_msd (that computes the msd)
[p,msd_est]=fmincon(@sum_ose_msd,p0,[],[],[],[],lb,up,[],optimset(...
'Display','off', 'LargeScale','off','Algorithm','active-set'),data);
```

Second Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt'); % load prediction set data and compute the
% msd of the prediction set msd_pre (based on the parameter estimate vector
msd_pre=sum_ose_msd(p,data); % p that minimizes msd_est)
```

Second Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre p(1) p(2) p(3)],-4),...
    'msd_est|msd_pre|lambdaS|alpha|beta','SUM_OSE_S')

% Author: Wasilios Hariskos, November 2012 %
```

	msd_est	msd_pre	lambdaS	alpha	beta
SUM_OSE_S	0.0016	0.0038	3.888	0.1551	0.1068

16.17 sum_ose_msd.m

Function sum_ose_msd.m computes for a given parameter estimate vector p the MSD of the second mover stochastic utility maximizer (SUM) model with own welfare, social welfare, and equality (OSE) preferences (see Section 5.4.1). The input arguments specify parameter estimate vector p and the data (estimation set or prediction set). The function computes the MSD in three steps: For the 120 games, it stores first the payoff structure and the observed choice probabilities, then it computes the predicted choice probabilities and lastly it computes the mean of squared deviations (MSD) between observed and predicted choice probabilities.

```
function msd = sum_ose_msd( p, data )
```

Experimental Data and Games

```
xF=data(:,[2 4 6]); % 120x3 payoffs of first mover F for each game
xS=data(:,[3 5 7]); % 120x3 payoffs of second mover S for each game
PR=data(:,9);       % 120x1 observed right probabilities f.e.g.
```

Predictions for Second Mover

```
n=120;  % number of games
left=2; % second column of xF or xS contains the left payoffs of the games
right=3;% third column of xF or xS contains the right payoffs of the games
% OWN WELFARE is maximized by ...
o(1:120,1)=1;                         % right choice (default)
for game=1:n
if xS(game,left)>xS(game,right)       % left choice
o(game,1)=0;
end
if xS(game,left)==xS(game,right)
o(game,1)=0.5;                        % left and right
end
end
% SOCIAL WELFARE is maximized by ...
s(1:120,1)=1;                         % right choice (default)
for game=1:n
if xS(game,left)+xF(game,left)>xS(game,right)+xF(game,right)% left choice
s(game,1)=0;
end
if xS(game,left)+xF(game,left)==xS(game,right)+xF(game,right)% left & right
s(game,1)=0.5;
end
end
% EQUALITY (self-centered) is maximized by ...
e(1:120,1)=1;                         % right choice (default)
for game=1:n
if xS(game,left)- abs(xS(game,left)- xF(game,left))>...
   xS(game,right)-abs(xS(game,right)-xF(game,right))% left choice
e(game,1)=0;
end
if xS(game,left)- abs(xS(game,left)- xF(game,left))==...
   xS(game,right)-abs(xS(game,right)-xF(game,right))% left and right
```

```
e(game,1)=0.5;
end
end
% Parameters
lambdaS=p(1);         % responsiveness to preferences
alpha=p(2);           % standard weight of component s and e
beta=p(3);            % additional weight of s if o is not discriminating
% Weighting vector if o is discriminating
w(2)=alpha;          % weight of social welfare component
w(3)=alpha;          % weight of equality (self-centered) component
w(1)=1-w(2)-w(3);    % weight of own welfare component
% Utility of option right if o is discriminating (default)
uSr=w(1)*o+w(2)*s+w(3)*e; % 120x3 utilities for each outcome of each game
% Weighting vector if o is not discriminating
w(2)=alpha+beta;   % higher weight for social welfare component
w(3)=alpha;        % equal weight for equality (self-centered) component
w(1)=1-w(2)-w(3);  % lower weight for own welfare component
for game=1:n
if o(game,1)==0.5  % utility of option right if o in not discriminating
uSr(game,1)=w(1)*o(game,1)+w(2)*s(game,1)+w(3)*e(game,1);
end
end
uSl=1-uSr;         % utility of option left
pr=exp(lambdaS*uSr)...% 120x1 predicted right probabilities for each game
    ./(exp(lambdaS*uSr)+exp(lambdaS*uSl));
```

MSD of Second Mover Model

```
sdpr=(PR-pr).*(PR-pr); % 120x1 sq. dev.: obs. vs pred. right probabilities
msd=mean(sdpr);        % mean of 120x1 squared deviations

end

% Author: Wasilios Hariskos, November 2012 %
```

16.18 sum_ose_unc.m

The MATLAB program sum_ose_unc.m estimates the second mover stochastic utility maximizer (SUM) model with own welfare, social welfare, and equality (OSE) preferences (see Section 5.4.1) and computes the MSD of the estimation set and the MSD of the prediction set. The visual output which is shown in the command window of MATLAB contains the estimation MSD, the prediction MSD and the parameter estimates that minimize the estimation MSD. Please ensure that sum_ose_unc_msd.m (the function that computes the MSD, see Appendix 16.19), disptable.m (the function that displays the results, see Appendix 16.3), ext_est_res.txt (a text file that contains the data of the estimation set, see Table 6) and ext_pred_res.txt (a text file that contains the data of the prediction set, see Table 7) are in the same folder.

Second Mover Model: MSD of the Estimation Set

```
clear all;                          % remove items from workspace
data=load('ext_est_res.txt');       % load estimation set data
% The parameter estimate vector of the second mover model is given by
% p = (lambdaS alpha beta gamma delta).
p0=[0 0 0 0 0];                     % set initial values of p equal to 0
lb=p0;                              % set lower bounds of p equal to 0
up=[];                              % set no upper bound up for p
% find the parameter estimate vector p that minimizes the msd of the
% estimation set (msd_est) by using the constrained nonliner multivariable
% function fmincon which calls function sum_ose_msd (that computes the msd)
[p,msd_est]=fmincon(@sum_ose_unc_msd,p0,[],[],[],[],lb,up,[],optimset(...
'Display','off', 'LargeScale','off','Algorithm','active-set'),data);
```

Second Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt'); % load prediction set data and compute the
% msd of the prediction set msd_pre (based on the parameter estimate vector
msd_pre=sum_ose_unc_msd(p,data); % p that minimizes msd_est)
```

Second Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre p(1) p(2) p(3) p(4) p(5)],-4),...
    'msd_est|msd_pre|lambdaS|alpha|beta|gamma|delta','SUM_OSE_S')

% Author: Wasilios Hariskos, November 2012 %
```

```
           msd_est  msd_pre  lambdaS   alpha    beta   gamma   delta
SUM_OSE_S   0.0016   0.0038    3.904  0.1628  0.1474  0.2631  0.1567
```

16.19 sum_ose_unc_msd.m

Function sum_ose_unc_msd.m computes for a given parameter estimate vector p the MSD of the second mover stochastic utility maximizer (SUM) model with own welfare, social welfare, and equality (OSE) preferences (see Section 5.4.1). The input arguments specify parameter estimate vector p and the data (estimation set or prediction set). The function computes the MSD in three steps: For the 120 games, it stores first the payoff structure and the observed choice probabilities, then it computes the predicted choice probabilities and lastly it computes the mean of squared deviations (MSD) between observed and predicted choice probabilities.

```
function msd = sum_ose_unc_msd( p, data )
```

Experimental Data and Games

```
xF=data(:,[2 4 6]); % 120x3 payoffs of first mover F for each game
xS=data(:,[3 5 7]); % 120x3 payoffs of second mover S for each game
PR=data(:,9);       % 120x1 observed right probabilities f.e.g.
```

Predictions for Second Mover

```
n=120;  % number of games
left=2; % second column of xF or xS contains the left payoffs of the games
right=3;% third column of xF or xS contains the right payoffs of the games
% OWN WELFARE is maximized by ...
o(1:120,1)=1;                        % right choice (default)
for game=1:n
if xS(game,left)>xS(game,right)      % left choice
o(game,1)=0;
end
if xS(game,left)==xS(game,right)
o(game,1)=0.5;                       % left and right
end
end
% SOCIAL WELFARE is maximized by ...
s(1:120,1)=1;                        % right choice (default)
for game=1:n
if xS(game,left)+xF(game,left)>xS(game,right)+xF(game,right)% left choice
s(game,1)=0;
end
if xS(game,left)+xF(game,left)==xS(game,right)+xF(game,right)% left & right
s(game,1)=0.5;
end
```

```
end
% EQUALITY (self-centered) is maximized by ...
e(1:120,1)=1;                          % right choice (default)
for game=1:n
if xS(game,left)- abs(xS(game,left)- xF(game,left))>...
   xS(game,right)-abs(xS(game,right)-xF(game,right))% left choice
e(game,1)=0;
end
if xS(game,left)- abs(xS(game,left)- xF(game,left))==...
   xS(game,right)-abs(xS(game,right)-xF(game,right))% left and right
e(game,1)=0.5;
end
end
% Parameters
lambdaS=p(1);         % responsiveness to preferences
alpha=p(2);           % standard weight of s and e
beta=p(3);            % additional weight of s if o is not discriminating
gamma=p(4);
delta=p(5);
% Weighting vector if o is discriminating
w(2)=alpha;          % weight of social welfare component
w(3)=beta;          % weight of equality (self-centered) component
w(1)=1-w(2)-w(3);    % weight of own welfare component
% Utility of option right if o is discriminating (default)
uSr=w(1)*o+w(2)*s+w(3)*e; % 120x3 utilities for each outcome of each game
% Weighting vector if o is not discriminating
w(2)=gamma;        % weight for social welfare component
w(3)=delta;        % weight for equality (self-centered) component
w(1)=1-w(2)-w(3);  % lower weight for own welfare component
for game=1:n
if o(game,1)==0.5  % utility of option right if o in not discriminating
uSr(game,1)=w(1)*o(game,1)+w(2)*s(game,1)+w(3)*e(game,1);
end
end
uSl=1-uSr;         % utility of option left
pr=exp(lambdaS*uSr)...% 120x1 predicted right probabilities for each game
    ./(exp(lambdaS*uSr)+exp(lambdaS*uSl));
```

MSD of Second Mover Model

```
sdpr=(PR-pr).*(PR-pr); % 120x1 sq. dev.: obs. vs pred. right probabilities
msd=mean(sdpr);        % mean of 120x1 squared deviations
```

```
end

% Author: Wasilios Hariskos, November 2012 %
```

16.20 ttb.m

The MATLAB program ttb.m estimates the second mover take-the-best (TTB) model (see Section 5.4.2) and computes the MSD of the estimation set and the MSD of the prediction set. The visual output which is shown in the command window of MATLAB contains the estimation MSD, the prediction MSD and the parameter estimates that minimize the estimation MSD. Please ensure that ttb_msd.m (the function that computes the MSD, see Appendix 16.21), disptable.m (the function that displays the results, see Appendix 16.3), ext_est_res.txt (a text file that contains the data of the estimation set, see Table 6) and ext_pred_res.txt (a text file that contains the data of the prediction set, see Table 7) are in the same folder.

Second Mover Model: MSD of the Estimation Set

```
clear all;                        % remove items from workspace
data=load('ext_est_res.txt');    % load estimation set data
% The parameter estimate vector of the second mover model is given by
% p = (sA sB sC sD epsilon).
p0=[0 0 0 0 0];                   % set initial values of p equal to 0
lb=p0;                            % set lower bounds of p equal to 0
up=[];                            % set no upper bound up for p
% find the parameter estimate vector p that minimizes the msd of the
% estimation set (msd_est) by using the constrained nonliner multivariable
% function fmincon which calls function ttb_msd (that computes the msd)
[p,msd_est]=fmincon(@ttb_msd,p0,[],[],[],[],lb,up,[],optimset(...
'Display','off', 'LargeScale','off','Algorithm','sqp'),data);
```

Second Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt'); % load prediction set data and compute the
% msd of the prediction set msd_pre (based on the parameter estimate vector
msd_pre=ttb_msd(p,data); % p that minimizes msd_est)
```

Second Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre p(1:4) 1-sum(p(1:4)) p(5)],-4),...
    'msd_est|msd_pre|sA|sB|sC|sD|sE|epsilon','TTB_S')

% Author: Wasilios Hariskos, November 2012 %
```

```
      msd_est  msd_pre      sA      sB      sC      sD      sE  epsilon
TTB_S   0.0018   0.0038  0.2986  0.0707  0.0637  0.3845  0.1825   0.0167
```

16.21 ttb_msd.m

Function ttb_msd.m computes for a given parameter estimate vector p the MSD of the second mover take-the-best (TTB) model (see Section 5.4.2). The input arguments specify parameter estimate vector p and the data (estimation set or prediction set). The function computes the MSD in three steps: For the 120 games, it stores first the payoff structure and the observed choice probabilities, then it computes the predicted choice probabilities and lastly it computes the mean of squared deviations (MSD) between observed and predicted choice probabilities.

```
function msd = ttb_msd( p, data )
```

Experimental Data and Games

```
xF=data(:,[4 6]); % 120x3 payoffs of first mover F for each game
xS=data(:,[5 7]); % 120x3 payoffs of second mover S for each game
PR=data(:,9);     % 120x1 observed right probabilities f.e.g.
```

Predictions for Second Mover

```
n=120;  % number of games
left=1; % first column of xF or xS contains the left payoffs of the games
right=2;% second column of xF or xS contains the right payoffs of the games
% Attribute 1 = highest own payoff
ca1(1:120,2)=0;             % cue 1 values of alternatives are 0 (default)
for game=1:n
if xS(game,left)>xS(game,right)
ca1(game,:)=[1 0];          % cue 1 discriminates --> choose left
end
if xS(game,left)<xS(game,right)
ca1(game,:)=[0 1];          % cue 1 discriminates --> choose right
end
end
% Attribute 2 = highest joint payoff
ca2(1:120,2)=0;             % cue 2 values of alternatives are 0 (default)
for game=1:n
if xS(game,left)+xF(game,left)>xS(game,right)+xF(game,right)
ca2(game,:)=[1 0];          % cue 2 discriminates --> choose left
```

```
end
if xS(game,left)+xF(game,left)<xS(game,right)+xF(game,right)
ca2(game,:)=[0 1];            % cue 2 discriminates --> choose right
end
end
% Attribute 3 = highest own payoff - absolute payoff difference
ca3(1:120,2)=0;               % cue 3 values of alternatives are 0 (default)
for game=1:n
if xS(game,left)- abs(xS(game,left)- xF(game,left))>...
   xS(game,right)-abs(xS(game,right)-xF(game,right))
ca3(game,:)=[1 0];            % cue 3 discriminates --> choose left
end
if xS(game,left)- abs(xS(game,left)- xF(game,left))<...
   xS(game,right)-abs(xS(game,right)-xF(game,right))
ca3(game,:)=[0 1];            % cue 3 discriminates --> choose right
end
end
% Parameters
sA=p(1);            % share of type A second movers
sB=p(2);            % share of type B second movers
sC=p(3);            % share of type C second movers
sD=p(4);            % share of type D second movers
sE=1-sA-sB-sC-sD;   % share of type E second movers
epsilon=p(5);       % choice error (has the same value for all types & cues)
% Type A: Predicted choice probability right
prA(1:120,1)=0.5;           % random choice (default)
for game=1:n
if ca1(game,left)==1 && ca1(game,right)==0
    prA(game,1)=0+epsilon; % cue 1 discriminates --> left
end
if ca1(game,left)==0 && ca1(game,right)==1
    prA(game,1)=1-epsilon; % cue 1 discriminates --> right
end
end
% Type B: Predicted choice probability rightprA(1:120,1)=0.5;
prB(1:120,1)=0.5;
for game=1:n
if ca2(game,left)==1 && ca2(game,right)==0
    prB(game,1)=0+epsilon; % cue 2 discriminates --> left
end
if ca2(game,left)==0 && ca2(game,right)==1
    prB(game,1)=1-epsilon; % cue 2 discriminates --> right
end
```

```
end
% Type C: Predicted choice probability right
prC(1:120,1)=0.5;
for game=1:n
if ca3(game,left)==1 && ca3(game,right)==0
    prC(game,1)=0+epsilon; % cue 3 discriminates --> left
end
if ca3(game,left)==0 && ca3(game,right)==1
    prC(game,1)=1-epsilon; % cue 3 discriminates --> right
end
end
% Type D: Predicted choice probability right
prD(1:120,1)=prA;
for game=1:n
if prA(game,1)==0.5
    prD(game,1)=prB(game,1);
end
end
% Type E: Predicted choice probability right
prE(1:120,1)=prA;
for game=1:n
if prA(game,1)==0.5
    prE(game,1)=prC(game,1);
end
end% Predicted probability to observe a right choice
pr=sA*prA+sB*prB+sC*prC+sD*prD+sE*prE;
%[prA prB prC prD prE]
```

MSD of Second Mover Model

```
sdpr=(PR-pr).*(PR-pr); % 120x1 sq. dev.: obs. vs pred. right probabilities
msd=mean(sdpr);        % mean of 120x1 squared deviations

end

% Author: Wasilios Hariskos, November 2012 %
```

16.22 sqre_sub.m

The MATLAB program sqre_sub.m estimates the first mover subjective quantal response equilibrium (SQRE) model (see Section 5.4.3) and computes the MSD of the estimation set and the MSD of the prediction set. The visual output which is shown in the command window of MATLAB contains the estimation MSD, the prediction MSD and the parameter estimates that minimize the estimation MSD. Please ensure that sqre_sub_msd.m (the function that computes the MSD, see Appendix 16.23), disptable.m (the function that displays the results, see Appendix 16.3), ext_est_res.txt (a text file that contains the data of the estimation set, see Table 6) and ext_pred_res.txt (a text file that contains the data of the prediction set, see Table 7) are in the same folder.

First Mover Model: MSD of the Estimation Set

```
clear all;                      % remove items from workspace
data=load('ext_est_res.txt');   % load estimation set data
% The parameter estimate vector of the first mover model is given by
% p = (gamma delta s1 s2 s3).
p0=[0 0 0 0 0];         % set initial values of p equal to 0
lb=p0;                  % set lower bounds of p equal to 0
up=[];                  % set no upper bounds for p
% find the parameter estimate vector p that minimizes the msd of the
% estimation set (msd_est) by using function fmincon and function qre_fs_msd
[p,msd_est]=fmincon(@sqre_sub_msd,p0,[],[],[],[],lb,up,[],optimset(...
'Display','off', 'LargeScale','off','Algorithm','active-set'),data);
```

First Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt'); % load prediction set data and compute the
% msd of the prediction set msd_pre (based on the parameter estimate vector
msd_pre=sqre_sub_msd(p,data); % p that minimizes msd_est)
```

First Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre p],-4),...
    'msd_est|msd_pre|gamma|delta|s1|s2|s3','SQRE')

% Author: Wasilios Hariskos, November 2012 %
```

```
      msd_est  msd_pre  gamma   delta      s1      s2      s3
SQRE    0.005   0.0094  1.319  0.2157  0.5115  0.7029  0.8801
```

16.23 sqre_sub_msd.m

Function sqre_sub_msd.m computes for a given parameter estimate vector p the MSD of the first mover subjective quantal response equilibrium (SQRE) model (see Section 5.4.3. The input arguments specify parameter estimate vector p and the data (estimation set or prediction set). The function computes the MSD in three steps: For the 120 games, it stores first the payoff structure and the observed choice probabilities, then it computes the predicted choice probabilities and lastly it computes the mean of squared deviations (MSD) between observed and predicted choice probabilities.

```
function msd=sqre_sub_msd(p,data)
```

Experimental Data and Games

```
xF=data(:,[2 4 6]); % 120x3 payoffs of first mover F for each game
xS=data(:,[3 5 7]); % 120x3 payoffs of second mover S for each game
PI=data(:,8);       % 120x1 observed in probabilities for each game
```

Predictions for First Mover

```
gamma=p(1);                % determines responsiveness to preferences
alphaF=0;    alphaS=p(2);% degree to disadvantageous inequality aversion
betaF=p(2);  betaS=p(2);% degree of advantageous inequality aversion
s1=p(3);                   % share of selfish first movers
s2=p(4);                   % share of highly responsive first movers
s3=p(5);                   % share of first movers with self-centered beliefs

% TYPE 1
t=1;             % selfish + high responsiveness + self-centered belief
lambdaF=2*gamma; % own responsiveness
lambdaS=gamma;   % truncated belief about responsiveness of S
uS=xS(:,2:3);    % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=xF(:,1);                             % utility out
uF(:,2)=xF(:,2).*(1-pr(:,t))+xF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...          % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 2
t=2;             % selfish + high responsiveness + pessimistic belief
lambdaF=2*gamma; % own responsiveness
lambdaS=gamma;   % truncated belief about responsiveness of S
```

```
uS=-xF(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=xF(:,1);                              % utility out
uF(:,2)=xF(:,2).*(1-pr(:,t))+xF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...           % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 3
t=3;             % selfish + low responsiveness + self-centered belief
lambdaF=gamma;   % own responsiveness
lambdaS=0;       % truncated belief about responsiveness of S
uS=xS(:,2:3);    % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=xF(:,1);                              % utility out
uF(:,2)=xF(:,2).*(1-pr(:,t))+xF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...           % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 4
t=4;             % selfish + low responsiveness + pessimistic belief
lambdaF=gamma;   % own responsiveness
lambdaS=0;       % truncated belief about responsiveness of S
uS=-xF(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=xF(:,1);                              % utility out
uF(:,2)=xF(:,2).*(1-pr(:,t))+xF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...           % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TYPE 5
t=5;             % inequality averse + high lambda + self-centered belief
lambdaF=2*gamma; % own responsiveness
lambdaS=gamma;   % truncated belief about responsiveness of S
iaF=xF-alphaF*max(0,xS-xF)-betaF*max(0,xF-xS); % utils of F out, left, right
iaS=xS-alphaS*max(0,xF-xS)-betaS*max(0,xS-xF); % utils of S out, left, right
uS=iaS(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=iaF(:,1);                              % utility out
uF(:,2)=iaF(:,2).*(1-pr(:,t))+iaF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...           % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
```

```
% TYPE 6
t=6;             % inequality averse + high lambda + pessimistic belief
lambdaF=2*gamma; % own responsiveness
lambdaS=gamma;   % truncated belief about responsiveness of S
iaF=xF-alphaF*max(0,xS-xF)-betaF*max(0,xF-xS); % utilities out, left, right
uS=-iaF(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=iaF(:,1);                              % utility out
uF(:,2)=iaF(:,2).*(1-pr(:,t))+iaF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...            % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 7
t=7;             % inequality averse + low lambda + self-centered belief
lambdaF=gamma;   % own responsiveness
lambdaS=0;       % truncated belief about responsiveness of S
iaF=xF-alphaF*max(0,xS-xF)-betaF*max(0,xF-xS); % utilities out, left, right
iaS=xS-alphaS*max(0,xF-xS)-betaS*max(0,xS-xF); % utils of S out, left, right
uS=iaS(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=iaF(:,1);                              % utility out
uF(:,2)=iaF(:,2).*(1-pr(:,t))+iaF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...            % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 8
t=8;             % inequality averse + low lambda + pessimistic belief
lambdaF=gamma;   % own responsiveness
lambdaS=0;       % truncated belief about responsiveness of S
iaF=xF-alphaF*max(0,xS-xF)-betaF*max(0,xF-xS); % utilities out, left, right
uS=-iaF(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=iaF(:,1);                              % utility out
uF(:,2)=iaF(:,2).*(1-pr(:,t))+iaF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...            % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
pi  = s1*s2*s3*p_in(:,1)... % weighted choice probability in
    + s1*s2*(1-s3)*p_in(:,2)...
    + s1*(1-s2)*s3*p_in(:,3)...
    + s1*(1-s2)*(1-s3)*p_in(:,4)...
    + (1-s1)*s2*s3*p_in(:,5)...
    + (1-s1)*s2*(1-s3)*p_in(:,6)...
```

```
    + (1-s1)*(1-s2)*s3*p_in(:,7)...
    + (1-s1)*(1-s2)*(1-s3)*p_in(:,8);
```

MSD of First Mover Model

```
sdpi=(PI-pi).*(PI-pi); % 120x1 sq. dev.: obs. vs pred. in probabilities
msd=mean(sdpi);        % mean of 120x1 squared deviations

end

% Author: Wasilios Hariskos, November 2012 %
```

16.24 sqre_unc.m

The MATLAB program sqre_unc.m estimates the first mover subjective quantal response equilibrium (SQRE) model (see Section 5.4.3) and computes the MSD of the estimation set and the MSD of the prediction set. The visual output which is shown in the command window of MATLAB contains the estimation MSD, the prediction MSD and the parameter estimates that minimize the estimation MSD. Please ensure that sqre_unc_msd.m (the function that computes the MSD, see Appendix 16.25), disptable.m (the function that displays the results, see Appendix 16.3), ext_est_res.txt (a text file that contains the data of the estimation set, see Table 6) and ext_pred_res.txt (a text file that contains the data of the prediction set, see Table 7) are in the same folder.

First Mover Model: MSD of the Estimation Set

```
clear all;                       % remove items from workspace
data=load('ext_est_res.txt');    % load estimation set data
% The parameter estimate vector of the first mover model is given by
% p = (gamma alphaF betaF alphaS betaS s1 s2 s3).
p0=[0 0 0 0 0 0 0 0];   % set initial values of p equal to 0
lb=p0;                  % set lower bounds of p equal to 0
up=[];                  % set no upper bounds for p
% find the parameter estimate vector p that minimizes the msd of the
% estimation set (msd_est) by using function fmincon and function qre_fs_msd
[p,msd_est]=fmincon(@sqre_unc_msd,p0,[],[],[],[],lb,up,[],optimset(...
'Display','off', 'LargeScale','off','Algorithm','active-set'),data);
```

First Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt'); % load prediction set data and compute the
```

```
% msd of the prediction set msd_pre (based on the parameter estimate vector
msd_pre=sqre_unc_msd(p,data); % p that minimizes msd_est)
```

First Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre p],-4),...
    'msd_est|msd_pre|gamma|alphaF|betaF|alphaS|betaS|s1|s2|s3','SQRE')

% Author: Wasilios Hariskos, November 2012 %
```

	msd_est	msd_pre	gamma	alphaF	betaF	alphaS	betaS	s1	s2	s3
SQRE	0.0049	0.0093	1.283	0.019	0.1969	0.2268	0.2497	0.4852	0.7093	0.8761

16.25 sqre_unc_msd.m

Function sqre_unc_msd.m computes for a given parameter estimate vector p the MSD of the first mover subjective quantal response equilibrium (SQRE) model (see Section 5.4.3. The input arguments specify parameter estimate vector p and the data (estimation set or prediction set). The function computes the MSD in three steps: For the 120 games, it stores first the payoff structure and the observed choice probabilities, then it computes the predicted choice probabilities and lastly it computes the mean of squared deviations (MSD) between observed and predicted choice probabilities.

```
function msd=sqre_unc_msd(p,data)
```

Experimental Data and Games

```
xF=data(:,[2 4 6]); % 120x3 payoffs of first mover F for each game
xS=data(:,[3 5 7]); % 120x3 payoffs of second mover S for each game
PI=data(:,8);       % 120x1 observed in probabilities for each game
```

Predictions for First Mover

```
gamma=p(1);                 % determines responsiveness to preferences
alphaF=p(2); alphaS=p(4);% degree to disadvantageous inequality aversion
betaF=p(3);  betaS=p(5);% degree of advantageous inequality aversion
s1=p(6);                    % share of selfish first movers
s2=p(7);                    % share of highly responsive first movers
s3=p(8);                    % share of first movers with self-centered beliefs
```

```
% TYPE 1
t=1;            % selfish + high responsiveness + self-centered belief
lambdaF=2*gamma; % own responsiveness
lambdaS=gamma;   % truncated belief about responsiveness of S
uS=xS(:,2:3);    % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=xF(:,1);                              % utility out
uF(:,2)=xF(:,2).*(1-pr(:,t))+xF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...           % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 2
t=2;            % selfish + high responsiveness + pessimistic belief
lambdaF=2*gamma; % own responsiveness
lambdaS=gamma;   % truncated belief about responsiveness of S
uS=-xF(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=xF(:,1);                              % utility out
uF(:,2)=xF(:,2).*(1-pr(:,t))+xF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...           % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 3
t=3;            % selfish + low responsiveness + self-centered belief
lambdaF=gamma;   % own responsiveness
lambdaS=0;       % truncated belief about responsiveness of S
uS=xS(:,2:3);    % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=xF(:,1);                              % utility out
uF(:,2)=xF(:,2).*(1-pr(:,t))+xF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...           % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 4
t=4;            % selfish + low responsiveness + pessimistic belief
lambdaF=gamma;   % own responsiveness
lambdaS=0;       % truncated belief about responsiveness of S
uS=-xF(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=xF(:,1);                              % utility out
uF(:,2)=xF(:,2).*(1-pr(:,t))+xF(:,3).*pr(:,t); % expected utility in
```

```
p_in(:,t)=exp(lambdaF*uF(:,2))./...            % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TYPE 5
t=5;             % inequality averse + high lambda + self-centered belief
lambdaF=2*gamma; % own responsiveness
lambdaS=gamma;   % truncated belief about responsiveness of S
iaF=xF-alphaF*max(0,xS-xF)-betaF*max(0,xF-xS); % utils of F out, left, right
iaS=xS-alphaS*max(0,xF-xS)-betaS*max(0,xS-xF); % utils of S out, left, right
uS=iaS(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=iaF(:,1);                               % utility out
uF(:,2)=iaF(:,2).*(1-pr(:,t))+iaF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...            % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 6
t=6;             % inequality averse + high lambda + pessimistic belief
lambdaF=2*gamma; % own responsiveness
lambdaS=gamma;   % truncated belief about responsiveness of S
iaF=xF-alphaF*max(0,xS-xF)-betaF*max(0,xF-xS); % utilities out, left, right
uS=-iaF(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=iaF(:,1);                               % utility out
uF(:,2)=iaF(:,2).*(1-pr(:,t))+iaF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...            % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 7
t=7;             % inequality averse + low lambda + self-centered belief
lambdaF=gamma;   % own responsiveness
lambdaS=0;       % truncated belief about responsiveness of S
iaF=xF-alphaF*max(0,xS-xF)-betaF*max(0,xF-xS); % utilities out, left, right
iaS=xS-alphaS*max(0,xF-xS)-betaS*max(0,xS-xF); % utils of S out, left, right
uS=iaS(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=iaF(:,1);                               % utility out
uF(:,2)=iaF(:,2).*(1-pr(:,t))+iaF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...            % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 8
t=8;             % inequality averse + low lambda + pessimistic belief
```

```
lambdaF=gamma;   % own responsiveness
lambdaS=0;       % truncated belief about responsiveness of S
iaF=xF-alphaF*max(0,xS-xF)-betaF*max(0,xF-xS); % utilities out, left, right
uS=-iaF(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=iaF(:,1);                                 % utility out
uF(:,2)=iaF(:,2).*(1-pr(:,t))+iaF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...             % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
pi  = s1*s2*s3*p_in(:,1)... % weighted choice probability in
    + s1*s2*(1-s3)*p_in(:,2)...
    + s1*(1-s2)*s3*p_in(:,3)...
    + s1*(1-s2)*(1-s3)*p_in(:,4)...
    + (1-s1)*s2*s3*p_in(:,5)...
    + (1-s1)*s2*(1-s3)*p_in(:,6)...
    + (1-s1)*(1-s2)*s3*p_in(:,7)...
    + (1-s1)*(1-s2)*(1-s3)*p_in(:,8);
```

MSD of First Mover Model

```
sdpi=(PI-pi).*(PI-pi); % 120x1 sq. dev.: obs. vs pred. in probabilities
msd=mean(sdpi);        % mean of 120x1 squared deviations

end

% Author: Wasilios Hariskos, November 2012 %
```

16.26 sqre_svo_sub.m

The MATLAB program sqre_svo_sub.m estimates the first mover subjective quantal response equilibrium (SQRE-SVO) model (see Section 5.4.4) and computes the MSD of the estimation set and the MSD of the prediction set. The visual output which is shown in the command window of MATLAB contains the estimation MSD, the prediction MSD and the parameter estimates that minimize the estimation MSD. Please ensure that sqre_svo_sub_msd.m (the function that computes the MSD, see Appendix 16.27), disptable.m (the function that displays the results, see Appendix 16.3), ext_est_res.txt (a text file that contains the data of the estimation set, see Table 6) and ext_pred_res.txt (a text file that contains the data of the prediction set, see Table 7) are in the same folder.

First Mover Model: MSD of the Estimation Set

```
clear all;                       % remove items from workspace
data=load('ext_est_res.txt');    % load estimation set data
% The parameter estimate vector of the first mover model is given by
% p = (gamma delta epsilon s1 s2 s3).
p0=[0 0 0 0 .55 0];       % initial values of p equal
lb=p0;                    % set lower bounds of p equal to 0
up=[];                    % set no upper bounds for p
% find the parameter estimate vector p that minimizes the msd of the
% estimation set (msd_est) by using function fmincon and function qre_fs_msd
[p,msd_est]=fmincon(@sqre_svo_sub_msd,p0,[],[],[],[],lb,up,[],optimset(...
'Display','off', 'LargeScale','off','Algorithm','sqp'),data);
```

First Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt'); % load prediction set data and compute the
% msd of the prediction set msd_pre (based on the parameter estimate vector
msd_pre=sqre_svo_sub_msd(p,data); % p that minimizes msd_est)
```

First Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre p],-4),...
'msd_est|msd_pre|gamma|delta|epsilon|s1|s2|s3'...
,'SVO-SQRE-SUB')

% Author: Wasilios Hariskos, November 2012 %
```

	msd_est	msd_pre	gamma	delta	epsilon	s1	s2	s3
SVO-SQRE-SUB	0.0051	0.009	1.194	0.1859	0.3442	0.5703	0.7036	0.8704

16.27 sqre_svo_sub_msd.m

Function sqre_svo_sub_msd.m computes for a given parameter estimate vector p the MSD of the first mover subjective quantal response equilibrium (SQRE-SVO) model (see Section 5.4.4. The input arguments specify parameter estimate vector p and the data (estimation set or prediction set). The function computes the MSD in three steps: For the 120 games, it stores first the payoff structure and the observed choice probabilities, then it computes the predicted choice probabilities and lastly it computes the mean of squared deviations (MSD) between observed and predicted choice probabilities.

```
function msd=sqre_svo_sub_msd(p,data)
```

Experimental Data and Games

```
xF=data(:,[2 4 6]); % 120x3 payoffs of first mover F for each game
xS=data(:,[3 5 7]); % 120x3 payoffs of second mover S for each game
PI=data(:,8);       % 120x1 observed in probabilities for each game
```

Predictions for First Mover

```
gamma=p(1);                % determines responsiveness to preferences
alphaF=p(2);alphaS=p(2)+p(3);% degree to disadvantageous inequality aversion
betaF=p(2);  betaS=p(2);% degree of advantageous inequality aversion
omegaF=p(2); omegaS=p(2);% degree of social welfare orientation
s1=p(4);                   % share of selfish first movers
s2=p(5);                   % share of highly responsive first movers
s3=p(6);                   % share of first movers with self-centered beliefs

% TYPE 1
t=1;             % selfish + high responsiveness + self-centered belief
lambdaF=2*gamma; % own responsiveness
lambdaS=gamma;   % truncated belief about responsiveness of S
uS=xS(:,2:3);    % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=xF(:,1);                              % utility out
uF(:,2)=xF(:,2).*(1-pr(:,t))+xF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...           % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 2
t=2;             % selfish + high responsiveness + pessimistic belief
lambdaF=2*gamma; % own responsiveness
lambdaS=gamma;   % truncated belief about responsiveness of S
uS=-xF(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=xF(:,1);                              % utility out
uF(:,2)=xF(:,2).*(1-pr(:,t))+xF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...           % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 3
t=3;             % selfish + low responsiveness + self-centered belief
lambdaF=gamma;   % own responsiveness
lambdaS=0;       % truncated belief about responsiveness of S
uS=xS(:,2:3);    % belief about preferences of S
```

```
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=xF(:,1);                                 % utility out
uF(:,2)=xF(:,2).*(1-pr(:,t))+xF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...             % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 4
t=4;              % selfish + low responsiveness + pessimistic belief
lambdaF=gamma;    % own responsiveness
lambdaS=0;        % truncated belief about responsiveness of S
uS=-xF(:,2:3);    % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=xF(:,1);                                 % utility out
uF(:,2)=xF(:,2).*(1-pr(:,t))+xF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...             % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TYPE 5
t=5;              % prosocial + high lambda + self-centered belief
lambdaF=2*gamma; % own responsiveness
lambdaS=gamma;    % truncated belief about responsiveness of S
svoF=xF-alphaF*max(0,xS-xF)-betaF*max(0,xF-xS)...
    +omegaF*(xF+xS); % utils of F out, left, right
svoS=xS-alphaS*max(0,xF-xS)-betaS*max(0,xS-xF)...
    +omegaS*(xF+xS); % utils of S out, left, right
uS=svoS(:,2:3);    % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=svoF(:,1);                                 % utility out
uF(:,2)=svoF(:,2).*(1-pr(:,t))+svoF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...             % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 6
t=6;              % prosocial + high lambda + pessimistic belief
lambdaF=2*gamma; % own responsiveness
lambdaS=gamma;    % truncated belief about responsiveness of S
svoF=xF-alphaF*max(0,xS-xF)-betaF*max(0,xF-xS)...
    +omegaF*(xF+xS); % utils of F out, left, right
uS=-svoF(:,2:3);    % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=svoF(:,1);                                 % utility out
```

```
uF(:,2)=svoF(:,2).*(1-pr(:,t))+svoF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...           % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 7
t=7;             % prosocial + low lambda + self-centered belief
lambdaF=gamma;   % own responsiveness
lambdaS=0;       % truncated belief about responsiveness of S
svoF=xF-alphaF*max(0,xS-xF)-betaF*max(0,xF-xS)...
    +omegaF*(xF+xS); % utils of F out, left, right
svoS=xS-alphaS*max(0,xF-xS)-betaS*max(0,xS-xF)...
    +omegaS*(xF+xS); % utils of S out, left, right
uS=svoS(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=svoF(:,1);                              % utility out
uF(:,2)=svoF(:,2).*(1-pr(:,t))+svoF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...           % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 8
t=8;             % prosocial + low lambda + pessimistic belief
lambdaF=gamma;   % own responsiveness
lambdaS=0;       % truncated belief about responsiveness of S
svoF=xF-alphaF*max(0,xS-xF)-betaF*max(0,xF-xS)...
    +omegaF*(xF+xS); % utils of F out, left, right
uS=-svoF(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=svoF(:,1);                              % utility out
uF(:,2)=svoF(:,2).*(1-pr(:,t))+svoF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...           % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
pi  = s1*s2*s3*p_in(:,1)... % weighted choice probability in
    + s1*s2*(1-s3)*p_in(:,2)...
    + s1*(1-s2)*s3*p_in(:,3)...
    + s1*(1-s2)*(1-s3)*p_in(:,4)...
    + (1-s1)*s2*s3*p_in(:,5)...
    + (1-s1)*s2*(1-s3)*p_in(:,6)...
    + (1-s1)*(1-s2)*s3*p_in(:,7)...
    + (1-s1)*(1-s2)*(1-s3)*p_in(:,8);
```

MSD of First Mover Model

```
sdpi=(PI-pi).*(PI-pi); % 120x1 sq. dev.: obs. vs pred. in probabilities
```

```
msd=mean(sdpi);         % mean of 120x1 squared deviations

end

% Author: Wasilios Hariskos, November 2012 %
```

16.28 sqre_svo_unc.m

The MATLAB program sqre_svo_unc.m estimates the first mover subjective quantal response equilibrium (SQRE-SVO) model (see Section 5.4.4) and computes the MSD of the estimation set and the MSD of the prediction set. The visual output which is shown in the command window of MATLAB contains the estimation MSD, the prediction MSD and the parameter estimates that minimize the estimation MSD. Please ensure that sqre_svo_unc_msd.m (the function that computes the MSD, see Appendix 16.29), disptable.m (the function that displays the results, see Appendix 16.3), ext_est_res.txt (a text file that contains the data of the estimation set, see Table 6) and ext_pred_res.txt (a text file that contains the data of the prediction set, see Table 7) are in the same folder.

First Mover Model: MSD of the Estimation Set

```
clear all;                      % remove items from workspace
data=load('ext_est_res.txt');   % load estimation set data
% The parameter estimate vector of the first mover model is given by
% p = (gamma alphaF betaF omegaF alphaS betaS omegaS s1 s2 s3).
p0=[0 0 0 0 0 0 0 0 0 0];% set initial values of p equal to 0
lb=p0;                   % set lower bounds of p equal to 0
up=[];                   % set no upper bounds for p
% find the parameter estimate vector p that minimizes the msd of the
% estimation set (msd_est) by using function fmincon and function qre_fs_msd
[p,msd_est]=fmincon(@sqre_svo_unc_msd,p0,[],[],[],[],lb,up,[],optimset(...
'Display','off', 'LargeScale','off','Algorithm','sqp'),data);
```

First Mover Model: MSD of the Prediction Set

```
data=load('ext_pred_res.txt'); % load prediction set data and compute the
% msd of the prediction set msd_pre (based on the parameter estimate vector
msd_pre=sqre_svo_unc_msd(p,data); % p that minimizes msd_est)
```

First Mover Model: Visual Output

```
disptable(roundn([msd_est msd_pre p],-4),...
```

```
'msd_est|msd_pre|gamma|alphaF|betaF|omegaF|alphaS|betaS|omegaS|s1|s2|s3'...
,'SVO-SQRE')

% Author: Wasilios Hariskos, November 2012 %
```

	msd_est	msd_pre	gamma	alphaF	betaF	omegaF	alphaS	betaS	omegaS	s1	s2	s3
SVO-SQRE	0.0048	0.0093	1.187	0.0207	0.2004	0	0.4447	0.1838	0.134	0.491	0.7047	0.8806

16.29 sqre_svo_unc_msd.m

Function sqre_svo_unc_msd.m computes for a given parameter estimate vector p the MSD of the first mover subjective quantal response equilibrium (SQRE-SVO) model (see Section 5.4.4. The input arguments specify parameter estimate vector p and the data (estimation set or prediction set). The function computes the MSD in three steps: For the 120 games, it stores first the payoff structure and the observed choice probabilities, then it computes the predicted choice probabilities and lastly it computes the mean of squared deviations (MSD) between observed and predicted choice probabilities.

```
function msd=sqre_svo_unc_msd(p,data)
```

Experimental Data and Games

```
xF=data(:,[2 4 6]); % 120x3 payoffs of first mover F for each game
xS=data(:,[3 5 7]); % 120x3 payoffs of second mover S for each game
PI=data(:,8);       % 120x1 observed in probabilities for each game
```

Predictions for First Mover

```
gamma=p(1);               % determines responsiveness to preferences
alphaF=p(2); alphaS=p(5);% degree to disadvantageous inequality aversion
betaF=p(3);  betaS=p(6);% degree of advantageous inequality aversion
omegaF=p(4); omegaS=p(7);% degree of social welfare orientation
s1=p(8);                  % share of selfish first movers
s2=p(9);                  % share of highly responsive first movers
s3=p(10);                  % share of first movers with self-centered beliefs

% TYPE 1
t=1;             % selfish + high responsiveness + self-centered belief
lambdaF=2*gamma; % own responsiveness
lambdaS=gamma;   % truncated belief about responsiveness of S
```

```
uS=xS(:,2:3);    % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=xF(:,1);                              % utility out
uF(:,2)=xF(:,2).*(1-pr(:,t))+xF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...           % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 2
t=2;             % selfish + high responsiveness + pessimistic belief
lambdaF=2*gamma; % own responsiveness
lambdaS=gamma;   % truncated belief about responsiveness of S
uS=-xF(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=xF(:,1);                              % utility out
uF(:,2)=xF(:,2).*(1-pr(:,t))+xF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...           % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 3
t=3;             % selfish + low responsiveness + self-centered belief
lambdaF=gamma;   % own responsiveness
lambdaS=0;       % truncated belief about responsiveness of S
uS=xS(:,2:3);    % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=xF(:,1);                              % utility out
uF(:,2)=xF(:,2).*(1-pr(:,t))+xF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...           % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 4
t=4;             % selfish + low responsiveness + pessimistic belief
lambdaF=gamma;   % own responsiveness
lambdaS=0;       % truncated belief about responsiveness of S
uS=-xF(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=xF(:,1);                              % utility out
uF(:,2)=xF(:,2).*(1-pr(:,t))+xF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...           % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TYPE 5
t=5;             % prosocial + high lambda + self-centered belief
```

```
lambdaF=2*gamma; % own responsiveness
lambdaS=gamma;   % truncated belief about responsiveness of S
svoF=xF-alphaF*max(0,xS-xF)-betaF*max(0,xF-xS)...
    +omegaF*(xF+xS); % utils of F out, left, right
svoS=xS-alphaS*max(0,xF-xS)-betaS*max(0,xS-xF)...
    +omegaS*(xF+xS); % utils of S out, left, right
uS=svoS(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=svoF(:,1);                                  % utility out
uF(:,2)=svoF(:,2).*(1-pr(:,t))+svoF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...               % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 6
t=6;             % prosocial + high lambda + pessimistic belief
lambdaF=2*gamma; % own responsiveness
lambdaS=gamma;   % truncated belief about responsiveness of S
svoF=xF-alphaF*max(0,xS-xF)-betaF*max(0,xF-xS)...
    +omegaF*(xF+xS); % utils of F out, left, right
uS=-svoF(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=svoF(:,1);                                  % utility out
uF(:,2)=svoF(:,2).*(1-pr(:,t))+svoF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...               % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 7
t=7;             % prosocial + low lambda + self-centered belief
lambdaF=gamma;   % own responsiveness
lambdaS=0;       % truncated belief about responsiveness of S
svoF=xF-alphaF*max(0,xS-xF)-betaF*max(0,xF-xS)...
    +omegaF*(xF+xS); % utils of F out, left, right
svoS=xS-alphaS*max(0,xF-xS)-betaS*max(0,xS-xF)...
    +omegaS*(xF+xS); % utils of S out, left, right
uS=svoS(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=svoF(:,1);                                  % utility out
uF(:,2)=svoF(:,2).*(1-pr(:,t))+svoF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...               % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
% TYPE 8
t=8;             % prosocial + low lambda + pessimistic belief
```

```
lambdaF=gamma;   % own responsiveness
lambdaS=0;       % truncated belief about responsiveness of S
svoF=xF-alphaF*max(0,xS-xF)-betaF*max(0,xF-xS)...
    +omegaF*(xF+xS); % utils of F out, left, right
uS=-svoF(:,2:3);   % belief about preferences of S
pr(:,t)=exp(lambdaS*uS(:,2))./... % belief about choice probability
          (exp(lambdaS*uS(:,1))+exp(lambdaS*uS(:,2)));
uF(:,1)=svoF(:,1);                               % utility out
uF(:,2)=svoF(:,2).*(1-pr(:,t))+svoF(:,3).*pr(:,t); % expected utility in
p_in(:,t)=exp(lambdaF*uF(:,2))./...            % choice probability in
         (exp(lambdaF*uF(:,1))+exp(lambdaF*uF(:,2)));
pi  = s1*s2*s3*p_in(:,1)... % weighted choice probability in
    + s1*s2*(1-s3)*p_in(:,2)...
    + s1*(1-s2)*s3*p_in(:,3)...
    + s1*(1-s2)*(1-s3)*p_in(:,4)...
    + (1-s1)*s2*s3*p_in(:,5)...
    + (1-s1)*s2*(1-s3)*p_in(:,6)...
    + (1-s1)*(1-s2)*s3*p_in(:,7)...
    + (1-s1)*(1-s2)*(1-s3)*p_in(:,8);
```

MSD of First Mover Model

```
sdpi=(PI-pi).*(PI-pi); % 120x1 sq. dev.: obs. vs pred. in probabilities
msd=mean(sdpi);        % mean of 120x1 squared deviations

end

% Author: Wasilios Hariskos, November 2012 %
```

16.30 reliability.m

```
clear all;
est=load('ext_est_res.txt'); % estimation set of competition
pre=load('ext_pred_res.txt');% prediction set of competition
```

Trainings Sets and Validation Set

```
for i=1:119
training1(:,:,i)=[est; pre(1:i-1,:); pre(i+1:120,:)];   % LOOCV I
training2(:,:,i)=[pre(1:i-1,:); pre(i+1:120,:)];        % LOOCV II
end
```

```
i=120;
training1(:,:,i)=[est; pre(1:119,:)];                  % LOOCV I
training2(:,:,i)=pre(1:119,:);                         % LOOCV II
validation=pre;                                        % LOOCV I and II
```

Out-of-Sample Fit LOOCV

```
for I=1:2   % first mover (I=1) or second mover (I=2)
for i=1:120 % 120 cross validation results
for cv=1:2  % LOOCV I (cv=1) or LOOCV II (cv=2)
if cv==1
training=training1; % training sets of LOOCV I
else
training=training2; % training sets of LOOCV II
end
M=1; % SPE
[sd_pre(i,M,I,cv),]=qre_msd([80 80],I,validation(i,:));
M=2; %QRE
p0=[0 0 0 0 0 0 0]; lb=p0; up=[];
[p,]=fmincon(@qre_msd,p0,[],[],[],[],lb,up,[],optimset('Display','off',...
'LargeScale','off','Algorithm','sqp'),I,training(:,:,i));
[sd_pre(i,M,I,cv),]=qre_msd(p,I,validation(i,:));
M=3; %FS-QRE
[p,]=fmincon(@qre_fs_msd,p0,[],[],[],[],lb,up,[],optimset('Display',...
'off','LargeScale','off','Algorithm','sqp'),I,training(:,:,i));
[sd_pre(i,M,I,cv),]=qre_fs_msd(p,I,validation(i,:));
M=4; %BO-QRE
[p,]=fmincon(@qre_bo_msd,p0,[],[],[],[],lb,up,[],optimset('Display',...
'off','LargeScale','off','Algorithm','sqp'),I,training(:,:,i));
[sd_pre(i,M,I,cv),]=qre_bo_msd(p,I,validation(i,:));
M=5; %CR-QRE
lb=[0 -1 -1 0 0 -1 -1];
[p,]=fmincon(@qre_cr_msd,p0,[],[],[],[],lb,up,[],optimset('Display',...
'off','LargeScale','off','Algorithm','sqp'),I,training(:,:,i));
[sd_pre(i,M,I,cv),]=qre_cr_msd(p,I,validation(i,:));
lb=p0;

if I==1
M=6; %7S
p=pred_f(training(:,:,i));
PIT=training(:,8,i);
alpha=regress(PIT,p);
alpha=alpha'/sum(alpha);
```

```
p=pred_f(validation(i,:)); pi=sum(p.*repmat(alpha,1,1),2);
sd_pre(i,M,I,cv)=(validation(i,8)-pi).^2;
M=7; %SQRE
[p,]=fmincon(@sqre_sub_msd,p0,[],[],[],[],lb,up,[],optimset('Display',...
'off','LargeScale','off','Algorithm','sqp'),training(:,:,i));
[sd_pre(i,M,I,cv),]=sqre_sub_msd(p,validation(i,:));
M=8; %SVO-SQRE
[p,]=fmincon(@sqre_svo_sub_msd,p0,[],[],[],[],lb,up,[],optimset('Display',...
'off','LargeScale','off','Algorithm','sqp'),training(:,:,i));
[sd_pre(i,M,I,cv),]=sqre_svo_sub_msd(p,validation(i,:));

else
M=6; %7S
p=pred_s(training(:,:,i));
PRT=training(:,9,i);
beta=regress(PRT,p);
beta=beta'/sum(beta);
p=pred_s(validation(i,:)); pr=sum(p.*repmat(beta,1,1),2);
sd_pre(i,M,I,cv)=(validation(i,9)-pr).^2;
M=7; %SUM
[p,]=fmincon(@sum_ose_msd,p0,[],[],[],[],lb,up,[],optimset('Display',...
'off','LargeScale','off','Algorithm','sqp'),training(:,:,i));
[sd_pre(i,M,I,cv),]=sum_ose_msd(p,validation(i,:));
M=8; %TTB
[p,]=fmincon(@ttb_msd,p0,[],[],[],[],lb,up,[],optimset('Display',...
'off','LargeScale','off','Algorithm','sqp'),training(:,:,i));
[sd_pre(i,M,I,cv),]=ttb_msd(p,validation(i,:));
end
end
end
end
```

Display Tables

```
disptable(roundn([mean(sd_pre(:,:,1,1))' mean(sd_pre(:,:,1,2))'...
mean(sd_pre(:,:,2,1))' mean(sd_pre(:,:,2,2))'],-4),...
'LOOCV I-F|LOOCV II-F|LOOCV-I-S|LOOCV-II-S',...
'SPE|QRE|FS-QRE|BO-QRE|CR-QRE|7S|OWN-I|OWN-II')
```

	LOOCV I-F	LOOCV II-F	LOOCV-I-S	LOOCV-II-S
SPE	0.0532	0.0532	0.0071	0.0071
QRE	0.0137	0.0139	0.0057	0.0058

```
FS-QRE     0.0121      0.0123     0.0056      0.0058
BO-QRE     0.0146      0.0144     0.0055      0.0056
CR-QRE     0.0121      0.0118     0.0051      0.0056
7S         0.0085       0.009     0.0042      0.0044
OWN-I      0.0082      0.0086     0.0038       0.004
OWN-II     0.0088      0.0079      0.004      0.0047
```

16.31 averaging.m

Please run reliability.m and matrices.m first to generate the data that are necessary for the computations of averaging.m; the output of averaging.m contains the tables in Section 5.5.3 and Appendix 15.

averaging.m

```
clear all
load reliability.mat sd_pre p_pre
p=p_pre;
load matrices.mat PI PR pi_p pr_p
```

Model Sets

```
C = cell(8,1);
for k=1:8 % generates the 255 model sets
C{k,1}=combntns(1:8,k);
end
```

Predictions for Each Validation Procedure

```
p_pre(:,:,1,1)=pi_p;
p_pre(:,:,2,1)=p(:,:,1,1);
p_pre(:,:,3,1)=p(:,:,1,2);
p_pre(:,:,1,2)=pr_p;
p_pre(:,:,2,2)=p(:,:,2,1);
p_pre(:,:,3,2)=p(:,:,2,2);
P=[PI PR];
```

Results

```
c=0;
set=zeros(255,8);
for k=1:8
    M=C{k,1}';          % set of model sets M={M_1,...,M_j,...,M_m}
    for Mj=1:size(M,2)  % for each model set M_j
    c=c+1;              % (counts number of model sets)
    set(c,1:size(M(:,Mj),1))=M(:,Mj)'; % save model set
for I=1:2
    for i=1:3           % for each splitting procedure
    mp(:,c,i,I)=mean(p_pre(:,M(:,Mj),i,I),2); % compute model set predictions
    msd(c,i,I)=mean((mp(:,c,i,I)-P(:,I)).^2); % compute model set msd
    vsd(c,i,I)=var((mp(:,c,i,I)-P(:,I)).^2);  % compute model set vsd
    end
end
end
end
for i=1:3 % optimal within procedures (owp)
    for I=1:2
    index(i,I)=sum((msd(:,i,I)==min(msd(:,i,I))).*(1:255)'); % find set index
    owp(i,:,I)=set(index(i,I),:);  % save set
    msd_owp(i,I)=msd(index(i,I),i,I);% save msd
    vsd_owp(i,I)=vsd(index(i,I),i,I);% save vsd
    end
end
% optimal between procedures (obp)
mmsd=mean(msd,2);
for I=1:2
oindex(I)=sum((mmsd(:,:,I)==min(mmsd(:,:,I))).*(1:255)'); % find set index
obp(1,:,I)=set(oindex(I),:);                % save set
end
for i=1:3
    for I=1:2
    msd_obp(i,I)=msd(oindex(I),i,I);             % save msd
    vsd_obp(i,I)=vsd(oindex(I),i,I);             % save vsd
    end
end
% best three models (btm)
for i=1:3
    for I=1:2
    mp_btm(:,i,I)=mean(p_pre(:,[6 7 8],i,I),2);   % calculate predictions
    msd_btm(i,I)=mean((mp_btm(:,i,I)-P(:,I)).^2); % calculate msd
```

```
    vsd_btm(i,I)=var((mp_btm(:,i,I)-P(:,I)).^2);  % calculate vsd
    end
end
% single best model (sbm)
I=1; %first mover
msd_sbm(:,:,I)=[  mean((mean(p_pre(:,6,1,I),2)-P(:,I)).^2)... % msd comp
        mean((mean(p_pre(:,7,2,I),2)-P(:,I)).^2)... % msd LOOCV-I
        mean((mean(p_pre(:,8,3,I),2)-P(:,I)).^2)];  % msd LOOCV-II
vsd_sbm(:,:,I)=[  var((mean(p_pre(:,6,1,I),2)-P(:,I)).^2)...  % vsd comp
        var((mean(p_pre(:,7,2,I),2)-P(:,I)).^2)...  % vsd LOOCV-I
        var((mean(p_pre(:,8,3,I),2)-P(:,I)).^2)];   % vsd LOOCV-II
I=2; % second mover
msd_sbm(:,:,I)=[  mean((mean(p_pre(:,7,1,I),2)-P(:,I)).^2)... % msd comp
        mean((mean(p_pre(:,7,2,I),2)-P(:,I)).^2)... % msd LOOCV-I
        mean((mean(p_pre(:,7,3,I),2)-P(:,I)).^2)];  % msd LOOCV-II
vsd_sbm(:,:,I)=[  var((mean(p_pre(:,7,1,I),2)-P(:,I)).^2)...  % vsd comp
        var((mean(p_pre(:,7,2,I),2)-P(:,I)).^2)...  % vsd LOOCV-I
        var((mean(p_pre(:,7,3,I),2)-P(:,I)).^2)];   % vsd LOOCV-II
```

Error and Variance

```
msd_f=[msd_sbm(:,:,1); msd_btm(:,1)'; msd_owp(:,1)'; msd_obp(:,1)']
vsd_f=[vsd_sbm(:,:,1); vsd_btm(:,1)'; vsd_owp(:,1)'; vsd_obp(:,1)']
msd_s=[msd_sbm(:,:,2); msd_btm(:,2)'; msd_owp(:,2)'; msd_obp(:,2)']
vsd_s=[vsd_sbm(:,:,2); vsd_btm(:,2)'; vsd_owp(:,2)'; vsd_obp(:,2)']
```

```
msd_f =

    0.0083    0.0082    0.0079
    0.0072    0.0071    0.0074
    0.0068    0.0067    0.0063
    0.0071    0.0069    0.0063


vsd_f =

  1.0e-003 *

    0.2069    0.2299    0.1979
    0.1644    0.1569    0.1747
    0.1456    0.1504    0.1474
    0.1530    0.1565    0.1474
```

```
msd_s =

    0.0038    0.0038    0.0040
    0.0037    0.0038    0.0041
    0.0033    0.0034    0.0036
    0.0033    0.0034    0.0037


vsd_s =

  1.0e-003 *

    0.1112    0.1112    0.1276
    0.1007    0.1005    0.1231
    0.0754    0.0814    0.1001
    0.0734    0.0790    0.1064
```

Optimal Model Sets

```
opt_m_f=[owp(:,1:3,1); obp(:,1:3,1)]
opt_m_s=[owp(:,1:4,2); obp(:,1:4,2)]
variance_ttb=var((mean(p_pre(:,8,1,2),2)-P(:,2)).^2)
```

```
opt_m_f =

     6     8     0
     3     6     7
     5     6     8
     5     6     8


opt_m_s =

     2     6     7     8
     2     6     7     8
     5     6     7     0
     2     6     7     0
```

```
variance_ttb =

  1.1349e-004
```

16.32 matrices.m

```
clear all
```

Model Predictions for Competition Procedure

```
spe;                m=1; pr_p(:,m)=pr_pre; pi_p(:,m)=pi_pre;
qre;                m=2; pr_p(:,m)=pr_pre; pi_p(:,m)=pi_pre;
qre_fs;             m=3; pr_p(:,m)=pr_pre; pi_p(:,m)=pi_pre;
qre_bo;             m=4; pr_p(:,m)=pr_pre; pi_p(:,m)=pi_pre;
qre_cr;             m=5; pr_p(:,m)=pr_pre; pi_p(:,m)=pi_pre;
seven_strategies;   m=6; pr_p(:,m)=pr_pre; pi_p(:,m)=pi_pre;
sqre_sub;           m=7; pi_p(:,m)=pi_pre;
sqre_svo_sub;       m=8; pi_p(:,m)=pi_pre;
sum_ose;            m=7; pr_p(:,m)=pr_pre;
ttb;                m=8; pr_p(:,m)=pr_pre;
```

Correlation Matrix Predictions First Mover (Prediction Set)

```
corr_p=corr(pi_p)
```

```
corr_p =

    1.0000    0.9212    0.9321    0.9315    0.9277    0.9026    0.8992    0.8895
    0.9212    1.0000    0.9848    0.9805    0.9816    0.9507    0.9684    0.9627
    0.9321    0.9848    1.0000    0.9951    0.9985    0.9496    0.9741    0.9692
    0.9315    0.9805    0.9951    1.0000    0.9936    0.9383    0.9668    0.9618
    0.9277    0.9816    0.9985    0.9936    1.0000    0.9482    0.9762    0.9720
    0.9026    0.9507    0.9496    0.9383    0.9482    1.0000    0.9787    0.9779
    0.8992    0.9684    0.9741    0.9668    0.9762    0.9787    1.0000    0.9985
    0.8895    0.9627    0.9692    0.9618    0.9720    0.9779    0.9985    1.0000
```

MSD Matrix Prediction Set First Mover

```
data=load('ext_pred_res.txt'); % load prediction set data
PI=data(:,8);
msd_c=zeros(8,8);
for r=1:8
for c=1:8
pi_c(:,r,c)=mean(pi_p(:,[r c]),2);
sdpi_c(:,r,c)=(PI-pi_c(:,r,c)).*(PI-pi_c(:,r,c));
msd_c(r,c)=mean(sdpi_c(:,r,c));
end
end
msd_c_p=msd_c
```

```
msd_c_p =

    0.0532    0.0233    0.0247    0.0261    0.0245    0.0182    0.0197    0.0184
    0.0233    0.0141    0.0129    0.0142    0.0128    0.0079    0.0093    0.0087
    0.0247    0.0129    0.0140    0.0152    0.0140    0.0076    0.0097    0.0092
    0.0261    0.0142    0.0152    0.0172    0.0152    0.0084    0.0108    0.0102
    0.0245    0.0128    0.0140    0.0152    0.0143    0.0076    0.0100    0.0095
    0.0182    0.0079    0.0076    0.0084    0.0076    0.0083    0.0070    0.0068
    0.0197    0.0093    0.0097    0.0108    0.0100    0.0070    0.0094    0.0091
    0.0184    0.0087    0.0092    0.0102    0.0095    0.0068    0.0091    0.0090
```

Correlation Matrix Predictions First Mover (Prediction Set)

```
corr_p_S=corr(pr_p)
```

```
corr_p_S =

    1.0000    0.9971    0.9966    0.9957    0.9915    0.9908    0.9914    0.9911
    0.9971    1.0000    0.9985    0.9972    0.9901    0.9882    0.9894    0.9895
    0.9966    0.9985    1.0000    0.9997    0.9915    0.9883    0.9914    0.9915
    0.9957    0.9972    0.9997    1.0000    0.9929    0.9891    0.9927    0.9927
    0.9915    0.9901    0.9915    0.9929    1.0000    0.9953    0.9962    0.9962
    0.9908    0.9882    0.9883    0.9891    0.9953    1.0000    0.9973    0.9974
    0.9914    0.9894    0.9914    0.9927    0.9962    0.9973    1.0000    0.9996
    0.9911    0.9895    0.9915    0.9927    0.9962    0.9974    0.9996    1.0000
```

MSD Matrix Prediction Set First Mover

```
data=load('ext_pred_res.txt'); % load prediction set data
PR=data(:,9);
msd_c=zeros(8,8);
for r=1:8
for c=1:8
pr_c(:,r,c)=mean(pr_p(:,[r c]),2);
sdpr_c(:,r,c)=(PR-pr_c(:,r,c)).*(PR-pr_c(:,r,c));
msd_c(r,c)=mean(sdpr_c(:,r,c));
end
end
msd_c_p_S=msd_c
save matrices.mat
```

```
msd_c_p_S =

    0.0071    0.0060    0.0060    0.0058    0.0059    0.0044    0.0042    0.0041
    0.0060    0.0057    0.0055    0.0053    0.0050    0.0036    0.0035    0.0035
    0.0060    0.0055    0.0056    0.0056    0.0052    0.0036    0.0036    0.0037
    0.0058    0.0053    0.0056    0.0056    0.0053    0.0036    0.0037    0.0037
    0.0059    0.0050    0.0052    0.0053    0.0067    0.0047    0.0045    0.0045
    0.0044    0.0036    0.0036    0.0036    0.0047    0.0043    0.0037    0.0038
    0.0042    0.0035    0.0036    0.0037    0.0045    0.0037    0.0038    0.0038
    0.0041    0.0035    0.0037    0.0037    0.0045    0.0038    0.0038    0.0038
```