The norm ultimatum

The **ultimatum game** (UG) is a **game** in experimental economics.

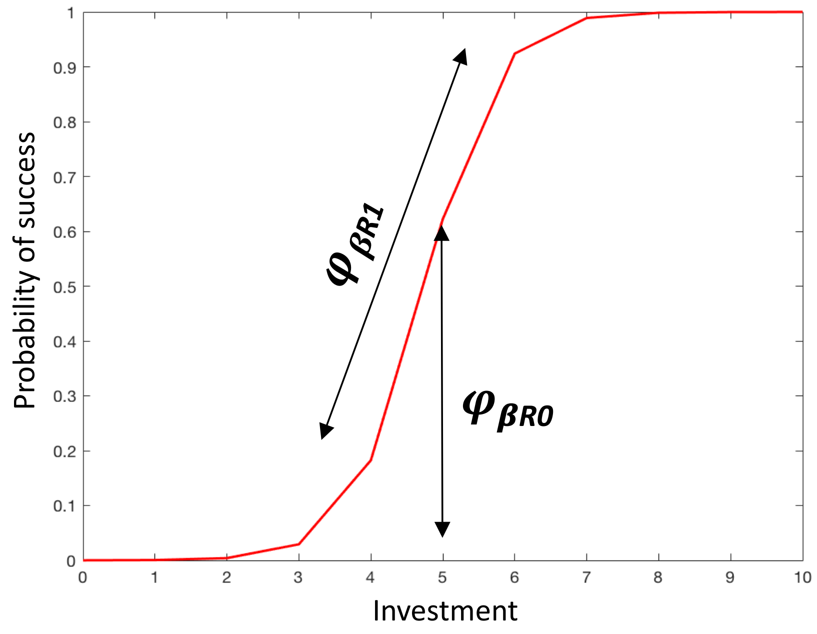
The first player, the proposer ***(P***), conditionally receives a sum of money and proposes how to divide the sum between the him or herself and the other player.

The second player ($\boldsymbol{R}$) make a decision $\boldsymbol{D}$ to either accept ($\boldsymbol{D} = \mathbf{1}$) or reject ($\boldsymbol{D} = \mathbf{0}$) this proposal.

We are interested in how ***P*** learns the acceptance function of $\boldsymbol{R}$.

Specifically, we ask if ***P*** plays the UG against a group of $\boldsymbol{R}$'s, each with a different acceptance function, how quickly will ***P*** be able to learn the lowest offers that $\boldsymbol{R}$ will accept (resistance point), even when these resistance points violate the **fairness norm**, which dictates a 50/50 split of ***P***'s endowment.

We assume that ***P*** learns $\boldsymbol{R}$'s acceptance logistic function $\boldsymbol{\varphi}$, which maps the investment amount into a victory or a defeat. This function is governed by two parameters, an intercept $\boldsymbol{\beta}_{\boldsymbol{R0}}$ and a slope $\boldsymbol{\beta}_{\boldsymbol{R}1}$.



Reinforcement Learning Framework

At each trial $\boldsymbol{t}$, from his/her endowment $\boldsymbol{E_t}$, $\boldsymbol{P}$ has to consider each possible offer $\boldsymbol{A_t}$ and compute the *expected payoff* $\boldsymbol{V_t}$. This expected payoff depends on his/her estimate of how likely it is that $\boldsymbol{A_t}$ will be accepted: $\boldsymbol{P_{R,t}\ (D_t = 1|A_t)}$.

For simplicity, this will be referred to as $\boldsymbol{P_{R,t}(A_t)}$. $\boldsymbol{P}$'s expected payoff $\boldsymbol{V_t}$ for the offer $\boldsymbol{A_t}$ is:

$$\boldsymbol{V_t = [P_{R,t}(A_t) \times (E_t - A_t)] + [(1 - P_{R,t}(A_t)) \times 0]} \tag{1}$$

With the two square brackets representing cases where $\boldsymbol{A_t}$ is accepted or rejected, respectively.

Critically, $\boldsymbol{P}$ estimates $\boldsymbol{P_{R,t}(A_t)}$ by estimating $\boldsymbol{R}$'s acceptance logistic function $\boldsymbol{\varphi}$. Specifically, ***P*** forms a representation ϑ of ***R***'s acceptance function $\boldsymbol{\varphi}$ by forming estimates $\boldsymbol{\beta_{R0}}$ and $\boldsymbol{\beta_{R1}}$ of $\boldsymbol{\varphi}$'s intercept and slope, respectively.

$$\boldsymbol{P_{R,t}(A_t) = 1/(1 + exp(-(\beta_{R0,t} + \beta_{R1,t} \times A_t)))} \tag{2}$$

Reinforcement Learning Framework

$\boldsymbol{P}$ can choose $\boldsymbol{A_t}$ according to a softmax function (multinomial), with a temperature parameter $\boldsymbol{\beta}$

$$\boldsymbol{P}(\boldsymbol{A_t} = \boldsymbol{i}) = \frac{e^{\beta \times V_i}}{\sum_{j=1}^{n} \beta \times V_i} \qquad (3)$$

Learning can be done using a simple delta rule on ϑ's two parameters $\boldsymbol{\beta_{R0,t}}$ and $\boldsymbol{\beta_{R1,t}}$. This means that ***P*** first computes a choice prediction error $\boldsymbol{\delta_{Ct}}$

$$\boldsymbol{\delta_{Ct}} = \boldsymbol{D_t} - \boldsymbol{P_{R,t}}(\boldsymbol{A_t}) \qquad (4)$$

Reinforcement Learning Framework

This choice prediction error can then be used to update the parameters for the next trial using a Rescorla-Wagner update function. For simplicity, we assume that $\boldsymbol{\beta}_{R1}$ will remain constant, while $\boldsymbol{\beta}_{R0}$ will be updated on each trial. We propose four different iterations of an update procedure that ***P*** can conceivably utilize in order to learn $\boldsymbol{\varphi}$.

Case 1: ***P*** updates his/her $\boldsymbol{\beta}_{\mathrm{R0}}$ at time *t* using the choice prediction error δ_{Ct} and a single learning rate α. Specifically, if δ_{Ct} is positive, ***P*** increases $\boldsymbol{\beta}_{\mathrm{R0}}$ (meaning ***P*** lowers the investment value necessary for victory), and if δ_{Ct} is negative ***P*** decreases $\boldsymbol{\beta}_{\mathrm{R0}}$ (meaning ***P*** raises the investment value necessary for victory – recall that increasing the intercept shifts the entire logit function to the right and hence lowers the expected probability of success for each investment):

$$\boldsymbol{\beta}_{R0,t+1}=\boldsymbol{\beta}_{R0,t}+\alpha\times\delta_{Ct} \qquad (5)$$

Reinforcement Learning Framework

Case 2: ***P*** updates his/her $\boldsymbol{\beta_{R0}}$ at time *t* using the choice prediction error δ_{Ct} and two learning rates: α^+ for positive prediction errors, and α^- for a negative prediction errors:

$$\beta_{R0,t+1} = \beta_{R0,t} + \alpha^+ \times \delta_{Ct} \qquad \text{if } \delta_{Ct} > 0 \qquad (6)$$

$$\beta_{R0,t+1} = \beta_{R0,t} + \alpha^- \times \delta_{Ct} \qquad \text{if } \delta_{Ct} < 0$$

Case 3: ***P*** updates his/her $\boldsymbol{\beta_{R0}}$ at time *t* using the *reward* prediction error δ_R and a single learning rate α. The calculation of δ_R is accomplished by subtracting the actual reward from the expected reward and is equivalent to the following expressions:

$$\delta_{Rt} = \delta_{Ct} \times (\boldsymbol{E}_t - \boldsymbol{A}_t) \qquad (7)$$

Reinforcement Learning Framework

$\delta_{\boldsymbol{Rt}}$ is then used by ***P*** to update ϑ in the same way $\delta_{\boldsymbol{Ct}}$ was used in cases 1 and 2:

$$\boldsymbol{\beta}_{\boldsymbol{R0,t+1}}=\boldsymbol{\beta}_{\boldsymbol{R0,t}}+\alpha\times\ \delta_{\boldsymbol{Rt}} \qquad (8)$$

Case 4: ***P*** updates his/her $\boldsymbol{\beta}_{\boldsymbol{R0}}$ at time *t* using the reward prediction error $\delta_{\boldsymbol{Rt}}$ and two learning rates: α^{+} for positive prediction errors, and α^{-} for a negative prediction errors:

$$\beta_{R0,t+1}=\beta_{R0,t}+\alpha^{+}\times\ \delta_{\boldsymbol{Rt}} \qquad \text{if } \delta_{\boldsymbol{Ct}} > 0 \qquad (9)$$

$$\beta_{R0,t+1}=\beta_{R0,t}+\alpha^{-}\times\ \delta_{\boldsymbol{Rt}} \qquad \text{if } \delta_{\boldsymbol{Ct}} < 0$$

Experimental design

We will employ a 2 (within: social vs. nonsocial) X 2 (between: known vs. unknown) design in which each subject will play against both real opponents and computer generated lotteries. Furthermore, subjects will be allocated into a “known” and an “unknown” condition (between subjects).

In the known condition, subjects will be informed that they are playing against responders who have received different starting endowments (social), and that the lotteries against which they are playing were generated from subject data in which responders started with different endowments (non-social).

In the unknown condition, subjects will be told only that the symbols represent different groups of responders (social) and that the symbols represent algorithms programmed to mimic the behavior of said groups (non-social).

Experimental design

In each of these conditions, the resistance points of ***R*** will be identical, and must be learned by ***P*** through trial and error.

However, in the social-known condition (where subjects know the reason for the different thresholds), subjects will be aware that the fairness norm is not violated by low offers, while in the social-unknown condition this fact is never made explicit.

Therefore, in the social-known condition, subjects could explore the full acceptance range in an attempt either to (i) learn what behavior is fair *or* (ii) to maximize profit.

Experiment phase 1: generate different environments, where fairness norm differs.

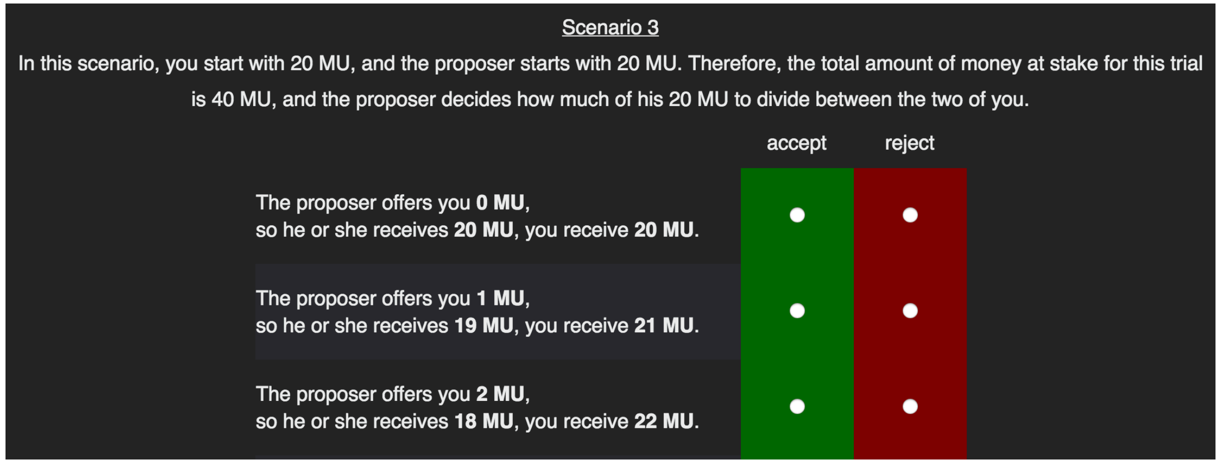
Strategy method, *without feedback*. Interested in the Responder's behavior

P received an endowment of 20€
You received an endowment of X€

5 Scenarios/Environments:

1: X = 0€ ~ fair 50/50 is 10€
2: X = 5€ ~ fair 50/50 is 7.5€
3: X = 10€ ~ fair 50/50 is 5€
4: X = 15€ ~ fair 50/50 is 2.5€
5: X = 20€ ~ fair 50/50 is 0€

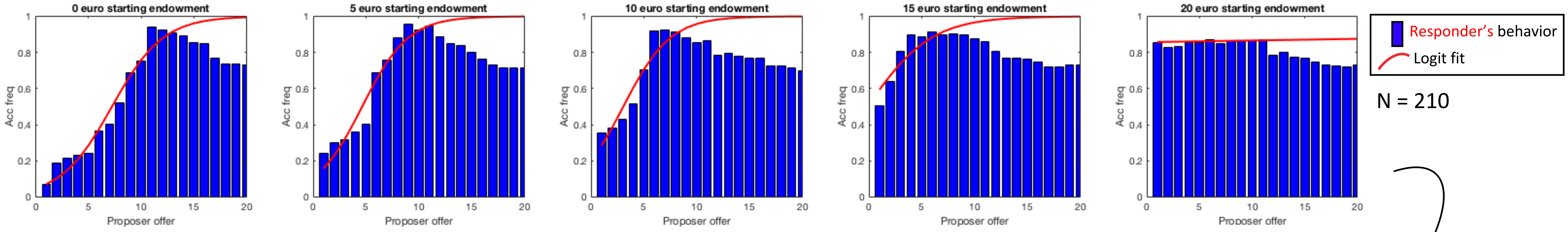
What participants saw



Scenario 3

In this scenario, you start with 20 MU, and the proposer starts with 20 MU. Therefore, the total amount of money at stake for this trial is 40 MU, and the proposer decides how much of his 20 MU to divide between the two of you.

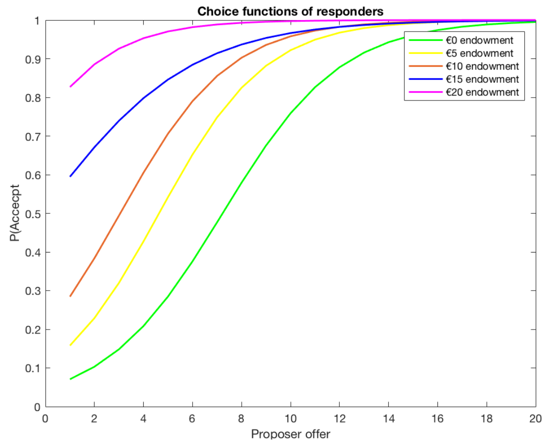
	accept	reject
The proposer offers you **0 MU**, so he or she receives **20 MU**, you receive **20 MU**.	●	●
The proposer offers you **1 MU**, so he or she receives **19 MU**, you receive **21 MU**.	●	●
The proposer offers you **2 MU**, so he or she receives **18 MU**, you receive **22 MU**.	●	●



N = 210

This generated different acceptance functions of the responders, with higher starting endowments creating lower resistance points.

We used these acceptance functions to simulate behavioral data in order to determine how many trials would be required for optimal model recovery/identifiability



Experiment phase 2: Investigating norm learning

These tasks involve feedback. We focus on Proposer behavior

Phase 2a/b: social norms VS maximizing reward.

You received an endowment of 20€.

Phase 2a (Known):

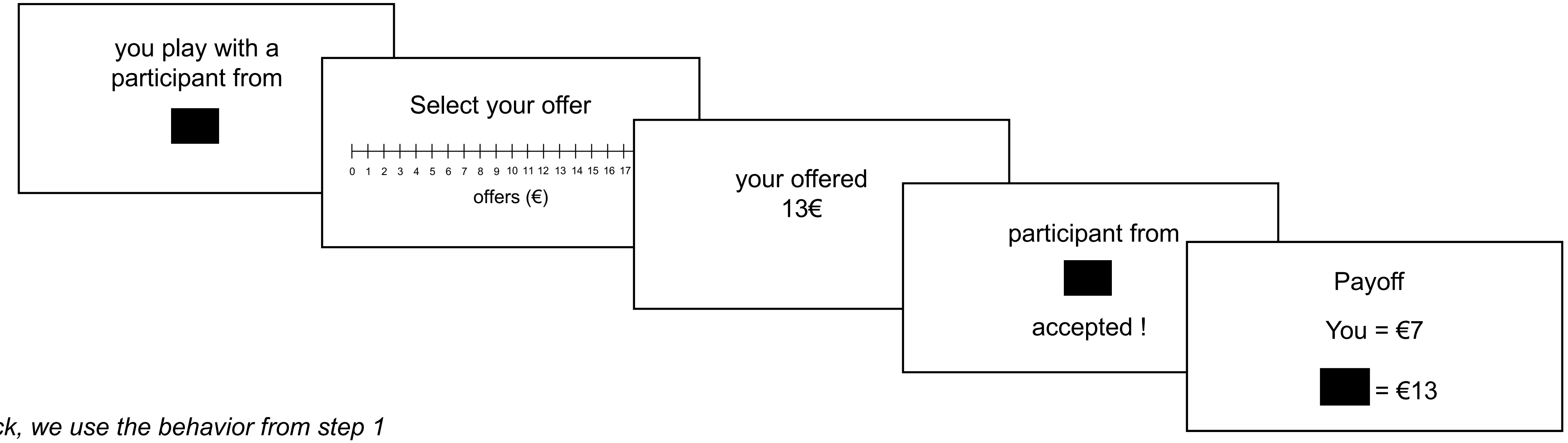
R will be repeatedly sampled from one of 5 experiments which received a fixed (but unknown to you) endowment of [0-20€].

Phase 2b (Unknown):

R is represented by different symbols

Each experiment will be indicated by an abstract symbol (■, ●, ♦, ◣, ★)

We pair the conditions/environments with the abstract symbols, e.g. 1: ■ / 2: ● / 3: ♦ / 4: ◣ / 5: ★



As feedback, we use the behavior from step 1

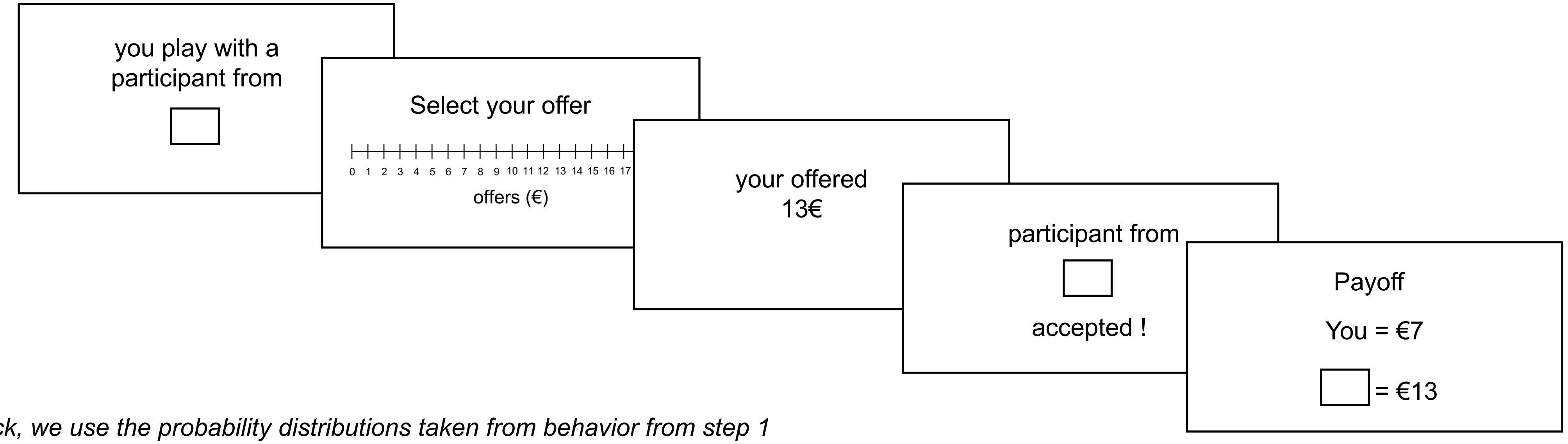
Experiment phase 2: Investigating norm learning
These tasks involve feedback. We focus on Proposer behavior

Phase 2c/d: social norms VS maximizing reward.

You received an endowment of 20€.
Phase 2c (Known):
R will be generated by algorithms mimicking the behavior of participants sampled from one of 5 experiments which received a fixed (but unknown to you) endowment of [0-20€].
Phase 2d (Unknown):
R is represented by different symbols

Each machine will be indicated by an abstract symbol (□, ○, ◊, ◺,☆)

We pair the conditions/environments with the abstract symbols, e.g. 1: □ / 2: ○ / 3: ◊/ 4:◺ / 5:☆

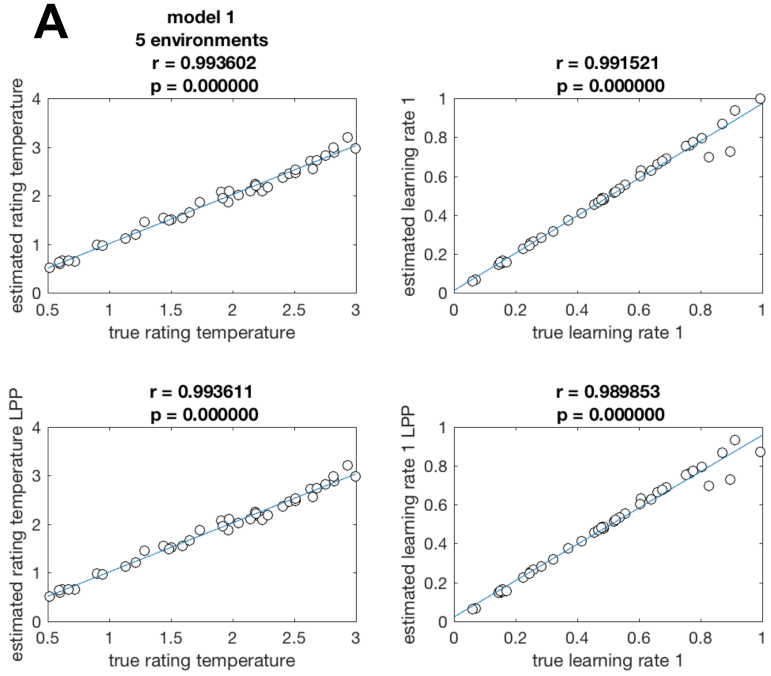


As feedback, we use the probability distributions taken from behavior from step 1

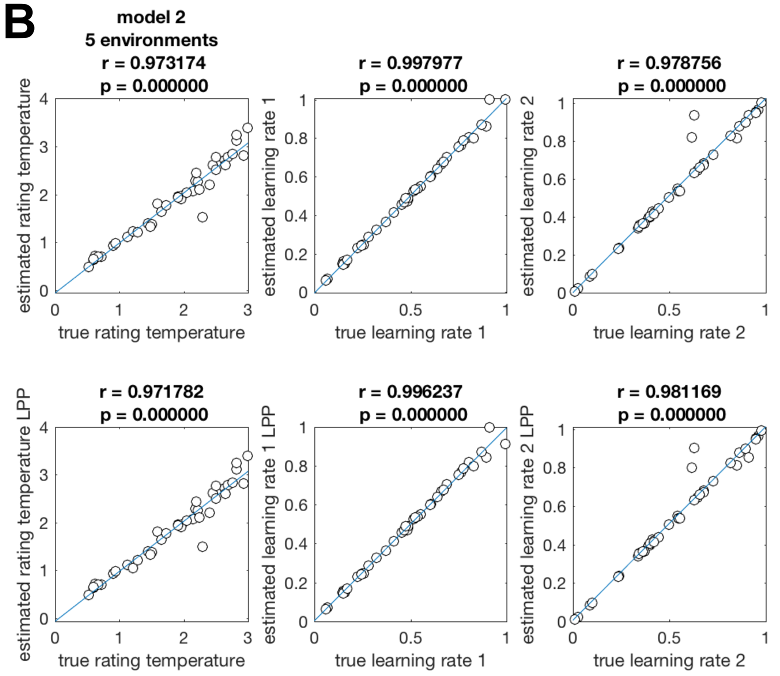
Simulations

40 simulations with 120 trials over 5 environments, 24 trials per environment. **A**, **B**, **D**, and **E** depict parameter recovery, with the top row of each using uniform parameter bounds and the bottom rows using Laplace approximation. In all cases Laplace approximation results in better recovery. **C** is a confusion matrix indicating how well each model is identified relative to the other models.

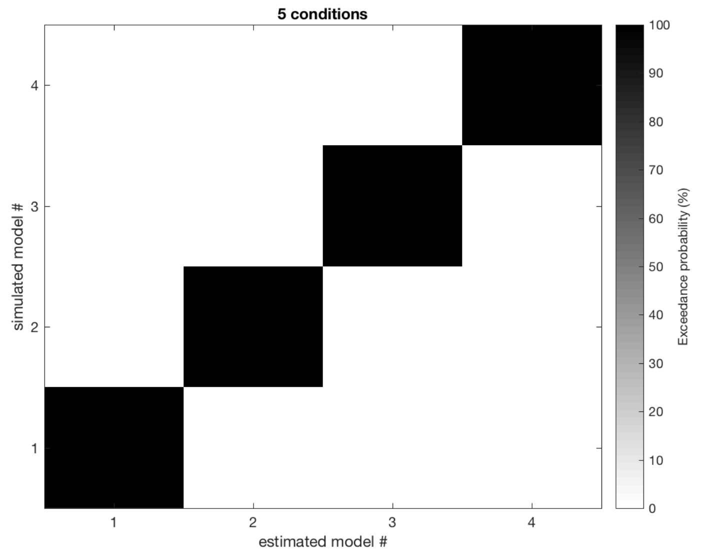
A



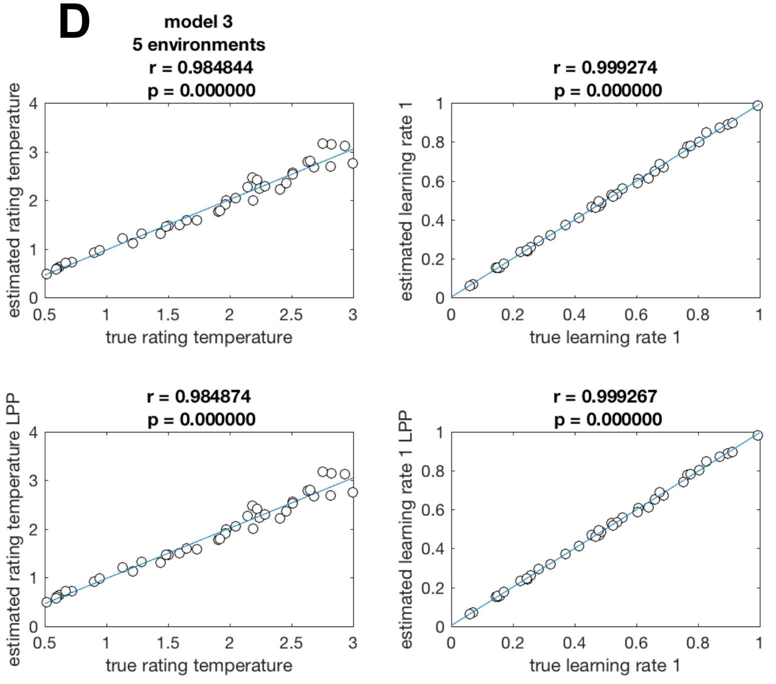
B



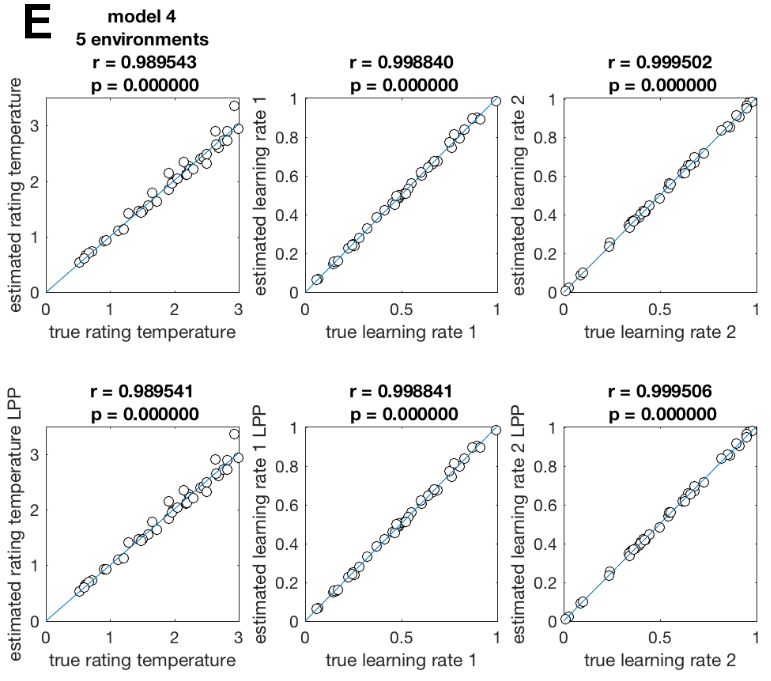
C



D



E



Design specifications

Based on simulations, we decided to use 5 scenarios/conditions, with 24 trials per scenario. If we conservatively estimate 1 trial to take 10 seconds (6 seconds for decision, 4 seconds for inter-stimulus interval), that results in 40 minutes of UG playing time, 20 minutes in the social, and 20 minutes in the non-social conditions

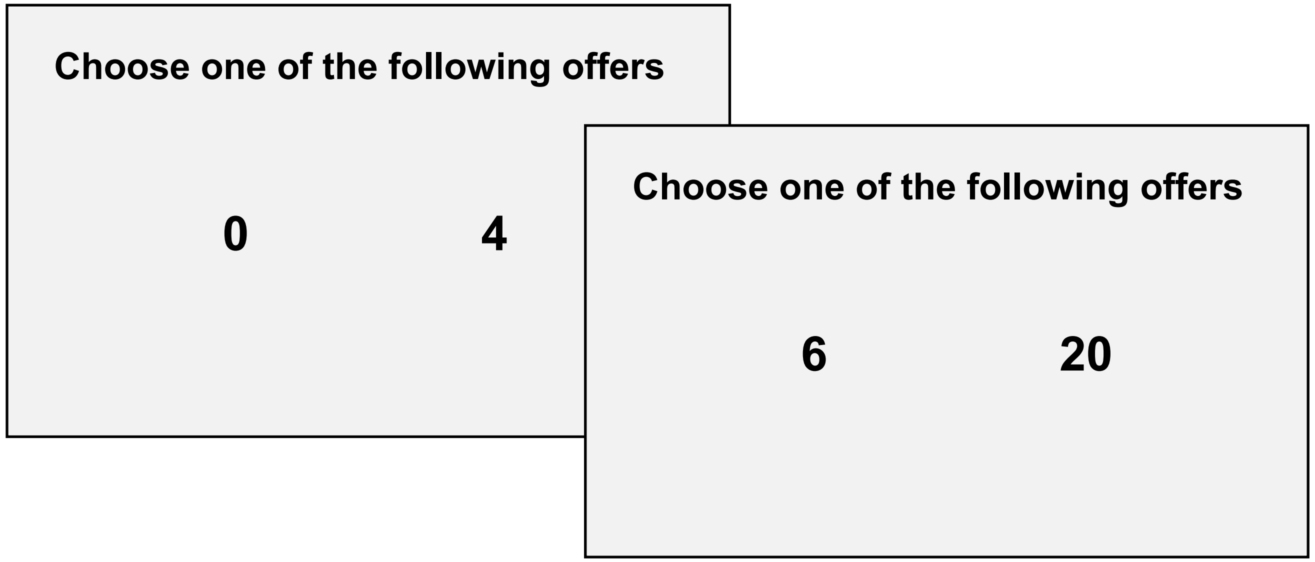
Additional tasks

- Fitting priors
  - To avoid fitting additional free parameters, the model requires initial estimates of $\boldsymbol{\beta}_{R0}$ and $\boldsymbol{\beta}_{R1}$.
- Dictator game
  - To get estimates of subjects' collective utility functions
- Risk assessment
  - Use quick and simple Eckel-Grossman lottery to assess risk preferences (Eckel & Grossman, 2002)

Fitting priors

These tasks involve no feedback. We focus on Proposer a priori beliefs about what offers will be accepted

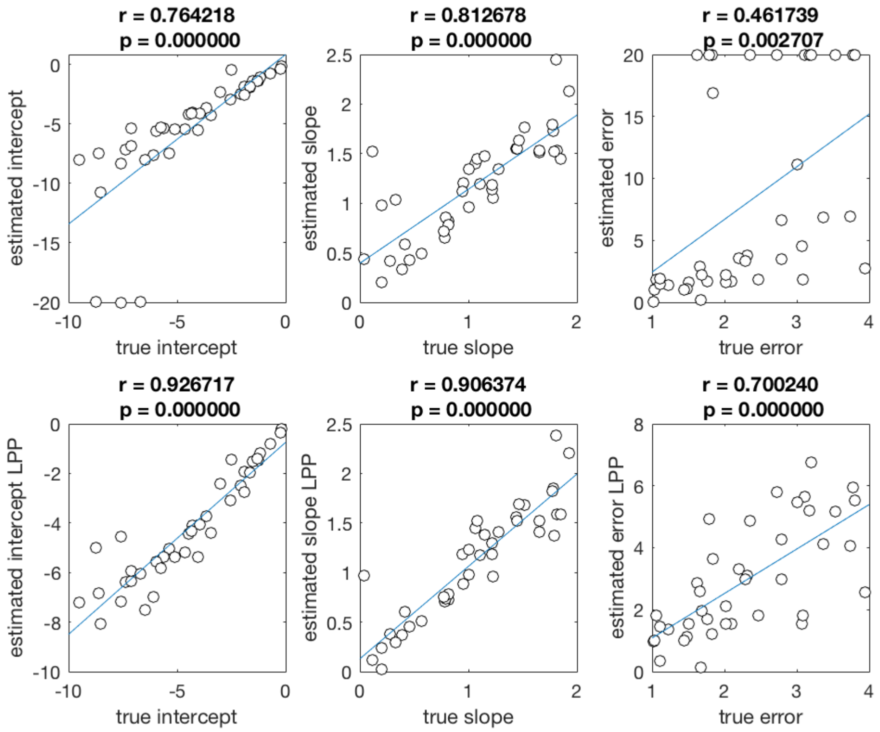
Ask proposers to make choices between two possible offers


Choose one of the following offers

0 4

Choose one of the following offers

6 20



These plots depict 40 simulated subjects making 100 different binary choices. The top row depicts parameters estimated from uniform bounds and the bottom rows depicts parameters estimated using Laplace approximation.

If we estimate each trial taking 3.5 seconds, with no ISI, then prior estimation should take 5.8 minutes.

These same specifications apply to the dictator game!

Design specifications

The entire experiment will take roughly 1 hour.

Prior estimation:	~7 minutes
Dictator game:	~7 minutes
Risk task:	~3 minutes
UG social:	~20 minutes
UG non-social:	~20 minutes
Total:	~57 minutes

If we estimate that each trial in the UG will take ~6 seconds, with a 4 second inter-stimulus interval, then we have 120 trials for the UG.

In order to help decide how many scenarios/environment to include, we simulated 40 subjects playing 120 trials of the UG with 3 (40 trials per environment), 4 (30 trials per environment), and 5 (24 trials per environment) in order to test for differences in parameter recovery and model identifiability.

In every case, parameter recovery was highly significant, however 5 environments slightly outperformed simulations with 3 and 4 conditions.

There was no difference in model identifiability.

Hypotheses

1: **Learning occurs:**

- In all conditions, proposers start with a (distributional) prior about the norm, but gradually integrate feedback to update the norm and improve payoff.

2: **Learning can be captured in an RL framework**

- Proposer learns the intercept of the acceptance function by trial-and-error

3: **Learning differs between social and non-social conditions**:

- Subjects learn faster in the non-social condition due to reluctance to explore acceptance space out of adherence to fairness norm
- More asymmetric learning in social (learn more from rejection than acceptance, because of social "acceptability") than in non-social, leading to sub-optimal behavior, (or perhaps the opposite - more responsive to victories in social condition?)

4: **Learning differs between the social-unknown and both social-unknown and non-social conditions, the latter two of which do not differ**:

- Faster learning in social-known than in social-unknown due to subjects using the opacity of what constitutes fair as an excuse to eschew fairness and maximize profit, and explore the acceptance space in the same way as in the non-social condition

6: **Priors differ between social and non-social**

- Uniform in the non-social condition, while driven by a norm expectation (50/50) in the social condition.

7: **Decision functions differ between social and non-social**

- More maximizer in non-social, and more exploratory (i.e. "soft") in social.

Supplementary material

If we estimate that each trial in the UG will take ~6 seconds, with a 4 second inter-stimulus interval, then we have 120 trials for the UG.

In order to help decide how many scenarios/environment to include, we simulated 40 subjects playing 120 trials of the UG with 3 (40 trials per environment), 4 (30 trials per environment), and 5 (24 trials per environment) in order to test for differences in parameter recovery and model identifiability.

In every case, parameter recovery was highly significant, however 5 environments slightly outperformed simulations with 3 and 4 conditions.

There was no difference in model identifiability.

We also ran simulations for prior estimation and found little difference between 8, 10, and 12 repetitions of each division possibility.

Ideas/Questions

Could starting in the non-social condition lead to more exploratory behavior in the social condition? If it just puts people into that mode… I would think that if there is an order effect, non-social -> social affects social, but social -> non-social doesn’t effect non-social.

Next…

- Pilot the study with interleafed and block design with ~12 participants in each, and see which one fits with our predictions closer.
- Pilot with 3 and 5 different environments, see if it matters empirically.
- Pre-registration with *As Predicted*?