Online Appendix for "Optimal Bargaining on eBay Using Reinforcement Learning"

Etan A. Green
etangr@upenn.edu

E. Barry Plunkett
bpiv400@gmail.com

Contents

A Simulation procedure

We simulate a listing until the item sells or until a week elapses.[1] A simulation begins by drawing an arrival time for the first buyer. Whenever a buyer arrives, we draw an experience level and a first offer. If the buyer accepts (i.e., pays the list price) or offers more than the auto-accept price, the offer is immediately accepted, and the listing ends. If the buyer offers less than the auto-reject price, the offer is immediately rejected, and we draw a delay time until the buyer responds. If the first offer is between the automatic thresholds, we draw a delay time until the seller responds and an interarrival time until the next buyer's arrival.

[1] On eBay, a typical listing period is 7 days—e.g., an item listed at noon on Sunday will be removed at noon the following Sunday. However, the data record only the start and end dates of each listing, not the start and end times. As a result, we run each listing from midnight on the start date until 11:59:59p a week later—i.e., 8 days total. We use a common duration for all listings, rather than the listing-specific duration in the data, because durations are only observed for listings that do not sell.

As on eBay, players have 48 hours to respond to an offer; otherwise, the offer is rejected. For delay times under 48 hours, we draw an offer. For buyer offers, we apply the automatic thresholds as above. A buyer rejection ends the thread. An acceptance by either party ends the listing. Otherwise, we draw a delay time for the other player, except in the case of the buyer's take-it-or-leave-it decision, which ends the thread.

As on eBay, players may attach short text messages to eligible offers, which exclude automatic offers, acceptances, and buyer rejections. The data record an indicator for whether a message accompanied an offer but not the contents of the message itself. In our simulations, we draw an indicator for whether a message accompanies an offer, when eligible.

We use $t \in \{1, \ldots, 7\}$ to denote the turn. Buyers make offers on odd turns and sellers on even turns.

B Data processing

We train simulator models using a dataset comprising all 98.3M Best Offer listings on eBay that were listed during the 12 months beginning June 1, 2012, with the full offer history for each listing [Backus et al., 2020a]. We process the data as follows:

1. We discard listings with duplicate titles and fine-grained "product" designations (e.g., iPhones), so as to restrict to unique items. Non-unique items are more likely to have a market price, which makes bargaining uninteresting and also impedes our ability to credibly simulate offer paths for these items, as we do not observe behavior in other markets. Removing listings with duplicate titles also removes items that were manually relisted, as when the seller creates a separate listing for the same item. (Items that are automatically relisted appear as one listing in the data.) As a result, each item is plausibly associated with one listing in the data.
2. We discard listings for which the data indicate that the list price was changed over the course of the listing, as we only observe the final list price.
3. We discard listings in which offers above the automatic accept price are not immediately accepted, offers below the automatic reject price are not immediately rejected, or offers between the automatic thresholds are immediately accepted or rejected. These circumstances indicate that the automatic thresholds have been changed over the course of the listing. As with the list price, we only observe final values for the automatic thresholds.
4. We discard listings with list prices less than \$9.95, for which there is little to bargain over.
5. We discard listings with list prices greater than \$1000, as in [Backus et al., 2020a], which are rare.
6. We discard listings from 4 sparsely populated categories.[2]

[2]These categories are Specialty Services, eBay Motors, Real Estate, and Gift Cards &

7. We discard listings that expire within one week of being listed, the listing duration that we use in our simulations.
8. We discard offers that arrive after the listing has been active for one week.
9. We redefine offers as concessions:

$$\texttt{concession}_t = \frac{\text{change in focal player's offer}}{\text{prior gap between players' offers}} = \frac{\texttt{offer}_{t-2} - \texttt{offer}_t}{\texttt{offer}_{t-2} - \texttt{offer}_{t-1}},$$

   where $\texttt{offer}_0$ is the list price, and $\texttt{offer}_{-1} = 0$. A concession of 0 is a rejection, a concession of 1 is an acceptance, and a concession of $\frac{1}{2}$ splits the difference between the last two offers. Finally, we round concessions to the nearest percentage point, but never rounding down to 0 or up to 1.
10. We discard listings for which at least one concession is either less than 0 or greater than 1, indicating that a player made a less generous offer than his or her previous offer.
11. eBay allows buyers who have rejected a seller counteroffer to rescind that rejection and make a counteroffer (but not accept), though this is rare. If such a counteroffer arrives within 48 hours of the seller's last offer, we ignore the rescinded rejection; if that counteroffer equals the seller's last offer, we code it as an accept and delete any subsequent offers. If the counteroffer arrives more than 48 hours after the seller's last offer, we discard the listing.

After these restrictions, our data comprise 13.4 million listings and 5.7 million threads. We randomize the 772k sellers (and their associated listings) into four partitions:

- `training`: for training the simulated agents (75% of sellers);
- `rl`: for training the discriminator and the agents (10%);
- `validation`: for choosing hyperparameters during training (5%); and
- `test`: a pure holdout sample [e.g., Kleinberg et al., 2018] for reporting results (10%).

In this draft, results are from the `validation` partition. For the published version, results will be reported from the `test` partition.

C Features

We create three sets of features: those specific to the listing, to the thread, and to the turn.

Listing-specific features

These features are of four types:

1. Listing attributes, comprising features that describe the list price, the automatic thresholds, the start date, the number of photos, the condition of the item, and whether fast shipping is offered.

Coupons.

2. Seller attributes, comprising features that describe the seller's experience on eBay and with Best Offer listings in particular, the seller's reputation, whether the seller has a store, and whether the seller is located in the US.
3. Context coordinates, which locate listings in a multidimensional space using their categorical assignment. Listings are assigned to one of 17,597 "leaf" categories. We translate each listing's categorical label into coordinates that measure categorical similarity. In particular, we construct a "sentence" from the categories that a given seller participates in, and we use `word2vec` to embed these categories in a 32-dimensional feature space. We then repeat the procedure, constructing each sentence from the categories that a particular buyer participates in. Figure A1 shows a two-dimensional representation of the 64-dimensional space, with categories colored by their corresponding "meta" category (e.g., antiques). Proximate categories are more likely to co-occur in the listings that a seller or buyer participates in.
4. Seller, leaf, and meta attributes, comprising features that summarize activity on listings from the same seller, in the same leaf or meta category—excluding the focal listing. For each grouping, we calculate the number of listings; the average number of threads per listing; the average number of offers per listing, separately for buyer and seller offers; the share of listings that end in a sale; the share of listings that sell for the list price; the rate at which offers expire, separately for buyer and seller offers; and the share of eligible offers that include a message, separately by turn. We also calculate four quantiles of each of the following distributions: the list price; the ratio of the sale price to the list price, among sales; the ratio of the first offer to the list price; the response time between offers, separately for seller and buyer offers; number of prior Best Offer listings that each buyer has participated in; and the rate at which buyers arrive.

Thread-specific features

These features describe three quantities: the time between arrivals, the number of Best Offer listings that a buyer has previously participated in, and the number of prior buyer arrivals.

Turn-specific features

For each turn, we create features that describe the amount of time since the previous turn (set to 0 for $t = 1$), the date and time of the offer, the amount of the offer, and whether it contains a message.

We also create features that describe activity on other threads. These include the number of other threads, both active and inactive; the total number of offers, separately for buyer and seller offers; the number of offers that are currently outstanding, separately for buyer and seller offers; the number of offers that have been made in the past 48 hours, separately for buyer and seller offers; the most generous buyer offer; the most generous seller offer; the most generous

outstanding buyer offer; the most generous outstanding seller offer; the most generous buyer offer in the past 48 hours; and the most generous seller offer in the past 48 hours.

D Simulator

The simulator consists of a set of models that underlie the simulation procedure described above.

Architecture

All simulator models, as well as the discriminator, are implemented as feed-forward neural networks with a common architecture. Groups of 2-3 dozen related features (e.g., offer features for a given turn) are embedded in an $N \times N$ transformation, where N is the number of features in the group. These transformed features are then concatenated with like groups into a family (e.g., offer features for all turns) and embedded in an $M \times M$ transformation, where M is the number of features in the family. These transformed features are then concatenated into a single hidden layer and passed through a 4-layer feed-forward neural networks with 1024 hidden nodes per layer. All layers of both the embedding and feed-forward networks use rectified linear activations and batch normalization. An output layer transforms the final 1024 hidden nodes into a model-specific number of parameters.

Models

We estimate two sets of models: one to predict buyer arrivals and another to predict the components of the offer. Every model uses all of the listing-specific features as inputs. The inclusion of thread- and turn-specific features is model-dependent.

Arrival models

We estimate the arrival of offers using 3 separate models.

- `arrival`:
  - Predicts: arrival time of the first buyer.
  - Output: multinomial distribution over 2305 categories: the 2304 5-minute intervals during the 8-day listing window plus a final category for a non-arrival.
  - Training data: all listings.
  - Loss: cross-entropy.
  - Note: When sampling an arrival time, the exact arrival time is drawn uniformly from the sampled 5-minute interval.
- `interarrival`:
  - Predicts: time between buyer arrivals.

  - Output: multinomial distribution over 2305 categories: the 2304 5-minute intervals during the 8-day listing window plus a final category for a non-arrival.
  - Training data: all interarrival periods, including the censored interarrival time between the last arrival and the end of the listing (except in the event that the last buyer offers the list price or an amount above the auto-accept threshold, in which case the listing ends immediately).
  - Loss: cross-entropy; for censored observations, the cross-entropy loss is calculated for the cumulative probability of censored intervals.
  - Thread features: days between the start of the listing and the most recent arrival; days between the past two arrivals; count of prior arrivals.
  - Note: When sampling an arrival time, the exact arrival time is drawn uniformly from the sampled 5-minute interval.

- `hist`:
  - Predicts: number of prior Best Offer listings that buyer has participated in.
  - Output: parameters of zero-inflated negative binomial distribution.
  - Training data: all arrivals.
  - Loss: negative likelihood for zero-inflated negative binomial distribution.
  - Thread features: days between the start of the listing and buyer's arrival; count of prior arrivals.

Offer models

All offer models are turn-specific and include turn-specific features. Models for buyer turns ($t \in \{1, 3, 5, 7\}$) do not include the turn-specific features that measure activity on other threads. We exclude these features because their inclusion could allow an agent buyer or seller to manipulate buyer behavior on other threads.

All offer models include the following thread-specific features: days between the start of the listing and buyer's arrival; count of prior arrivals; and the buyer's experience with Best Offer listings. All offer models for turn $t > 1$ include all turn-specific features for prior turns (e.g., $\{1, \ldots, t\}$). For the current turn, turn-specific features are limited to those are sampled earlier in the simulation.

- $\texttt{delay}_t$ for $t \in \{2, ..., 7\}$:
  - Predicts: waiting time between offers.
  - Output: multinomial distribution over 577 categories: 576 5-minute intervals in the 48-hour response window plus a final category for an expiration.
  - Training data: all response periods for turn t, including a censored period when the listing ends during the response period.

  - Loss: cross-entropy; for censored observations, the cross-entropy loss is calculated for the cumulative probability of censored intervals.
  - Current turn features: none.

- $\texttt{con}_t$ for $t \in \{1, ..., 7\}$:
  - Predicts: concession.
  - Output:
    - For $t \in \{1, ..., 6\}$, a multinomial distribution over 101 categories—every percentage point concession from 0 to 1.
    - For $t = 7$, a scalar—the probability of an acceptance.
  - Training data: all offers, excluding expirations and automatic seller responses.
  - Loss: cross-entropy.
  - Current turn features: those that describe the timing of the offer; for seller turns, those that summarize activity on other threads.

- $\texttt{message}_t$ for $t \in \{1, ..., 6\}$:
  - Predicts: whether a message accompanies an eligible offer.
  - Output: a scalar—the probability of a message.
  - Training data: all offers for which a message is eligible.
  - Loss: binary cross-entropy.
  - Current turn features: those that describe the timing of the offer and the amount; for seller turns, those that summarize activity on other threads.

Discriminator

We train a separate neural network to discriminate between simulated threads and those observed in the data. The discriminator observes every feature described in the previous section, along with the final turn of the thread. For threads that do not go the full 7 turns, features for unobserved turns are inputted as 0. Otherwise, the discriminator shares the same architecture with the models in the simulator.

We also train a separate "placebo" discriminator for which the simulated threads come from simulations in which the $\lambda = 1 + \epsilon$ heuristic buyer (see Figure 1b) replaces a randomly selected buyer.

Both discriminator models are trained using the `rl` partition.

Training

We use the Adam algorithm [Kingma and Ba, 2015] with a minibatch size of 128, $\beta_1 = .9$, $\beta_2 = .999$, and an initial learning rate of 10^{-3}. We step the learning rate down by a factor of 10 when the training loss does not improve by at least 1% from the previous epoch. Training ends when the learning rate reaches 10^{-7}.

We train using two dropout rates: one for the second layer of the embedding network, and one for all 4 layers of the fully-connected network. We tune the

dropout rates by grid search, choosing the pair of dropout rate that minimizes the loss in the `validation` set.

E Agent seller

The agent seller is a neural network that maps a set of features to a distribution over offers.

Timing of offers

We do not allow the agent to choose when it responds to a buyer offer. Instead, it responds when a human seller would. That is, we draw the seller's response time from the $\texttt{delay}_t$ model (for $t \in \{2, 4, 6\}$)—with one exception: we redraw in the case of an expiration delay, so as to allow the agent to actively choose to let a buyer offer expire. Figure A2 shows seller response times.

Action space

The agent chooses from among the 6 most common seller concessions observed in the data—$\frac{1}{4}$, $\frac{1}{3}$, $\frac{2}{5}$, $\frac{1}{2}$, $\frac{3}{5}$, $\frac{2}{3}$—along with `accept`, `reject`, and `expire`. We do not allow the agent to include a message with its offer. Recent evidence from eBay in Germany suggests that this imposes a handicap on the agent [Backus et al., 2020b].

Input features

The agent observes all of the features that we create for the simulator excepting the seller, category, and meta-category attributes, which are not directly observable to sellers on eBay. We also provide the current turn. Features that have not been determined before the agent acts (i.e., the offer in the current turn or attributes of future turns) are inputted as 0.

Architecture

The agent comprises two separate neural networks: a policy network, which outputs a multinomial distribution over the agent's 9 possible offers; and a value network, which outputs an estimated value of the current state (as represented by the features that the agent observes). The policy network is used to sample agent actions, while the value network provides a baseline for comparing realized payoffs when calculating the loss (see below). We implement both networks as feed-forward neural networks with the same architecture as the discriminator, except we use 512 hidden nodes, rather than 1024, owing to the smaller number of input features. We initialize both networks with random weights.

Training

A training epoch begins by drawing a listing randomly from the `rl` partition and simulating buyer arrivals and offer paths as in the simulator, except with the agent acting in place of the offer models for seller turns. When the listing ends, the agent receives a payoff (see below), and we draw another listing. After collecting 4096 agent offers and associated payoffs, we calculate the loss using the advantage action critic (A2C) algorithm [Mnih et al., 2016].

As when training the simulator, we use the Adam optimizer to update the parameters of both networks, with $\beta_1 = .9$, $\beta_2 = .999$. Unlike simulator training, however, we conduct the update in a single batch. For the policy network, we use a learning rate of 10^{-4}. For the value network, we use a learning rate of 10^{-3}, which allows the value estimate to catch up quickly when the policy changes. We do not train either network with dropout, as even modest dropout rates hurt performance in the `validation` partition.

Training initially runs for 1500 epochs with an entropy bonus of 10^{-3}. We then linearly decay the entropy bonus to 0 over the next 1500 epochs. Training continues until the agent converges to a deterministic policy,[3] or until the average entropy reaches 10^{-2}. In total, training lasts about 10^4 epochs. On a workstation with a high-end gaming GPU (Nvidia RTX 2080 Ti) and a CPU with 32 double-threaded physical cores, and using the parallelization framework in the `rlpyt` python package [Stooke and Abbeel, 2019], this takes less than one day.

Payoffs

What is the payoff to the agent at the end of the listing? If the item sells, the payoff is the sale price, x. If the listing does not sell, the agent gets to keep the item. Hence, the payoff is value of the item, which we denote v and define momentarily—discounted by $\delta \in [0, 1]$, which captures the decay in an item's value if it does not sell and which we treat as a hyperparameter. The agent's expected payoff is:

$$\begin{aligned} \mathbb{E}[\texttt{payoff}] &= \mathbb{E}\big[\mathbb{1}\{\texttt{sale}\} \cdot \texttt{sale\_price} + \mathbb{1}\{\neg\texttt{sale}\} \cdot \delta \cdot v\big] \\ &= P(\texttt{sale}) \cdot \mathbb{E}[\texttt{price}|\texttt{sale}] + \big(1 - P(\texttt{sale})\big) \cdot \delta \cdot v \end{aligned} \tag{1}$$

A fundamental limitation of empirical research on bargaining is that valuations are private and thus cannot be observed by the analyst. This is one reason why the vast majority of studies in this literature are conducted in the laboratory, where valuations can be assigned. Instead, we estimate a continuation value for each item, consistent with a seller who relists the item until it sells. To do so, we set the value of each item, v, equal to the expected value in

[3] Although the simulated buyers play stochastically, the optimal seller policy is deterministic because the buyer's response depends only on the seller's offer, not on the rates at which it mixes over potential offers.

(1). Solving for v,

$$v = \frac{P(\texttt{sale}) \cdot \mathbb{E}[\texttt{price}|\texttt{sale}]}{1 - \delta\big(1 - P(\texttt{sale})\big)} \tag{2}$$

For a perfectly patient seller ($\delta = 1$), or for an item that always sells, the value of an item is $\mathbb{E}[\texttt{price}|\texttt{sale}]$, its (eventual) sale price.

We estimate $P(\texttt{sale})$ and $\mathbb{E}[\texttt{price}|\texttt{sale}]$ by simulating each listing 1000 times. Figure A5 shows distributions of v for different values of δ. A strong predictor of an item's value is the seller's experience. As shown in Figure A6, items listed by more experienced sellers sell less often and for a lower share of the list price, suggesting that more experienced sellers more greatly inflate list prices. The agent does not observe v directly. Rather, it implicitly estimates v from the features that it observes.

Evaluation

We evaluate the agent seller by simulating each listing in the holdout partition 10 times, with the agent acting as the seller. The analyses in the paper compare the agent's behavior in these simulations to that of human sellers in the same partition.

F Agent buyer

The agent buyer is a neural network that maps a set of features to a distribution over offers.

Arrival process

For a given listing, the simulator begins by simulating an arrival time for every potential buyer. If no buyers are scheduled to arrive, we draw a new listing. Otherwise, one of the arrivals is chosen uniformly at random to be the agent. Note that if the item sells before the selected arrival time, the agent buyer will not have an opportunity to make a first offer.

Timing of later offers

We do not allow the agent to choose when it responds to a seller offer. Instead, it responds when a human buyer would. That is, we draw the seller's response time from the $\texttt{delay}_t$ model (for $t \in \{3, 5, 7\}$)—with one exception: we redraw in the case of an expiration walk, so as to allow the agent to actively choose to walk. Figure A3 shows buyer response times.

Action space

The agent chooses from among the 6 most common, turn-specific buyer concessions observed in the data. For turn 1, the agent chooses from among the

following fractions of the list price: $\frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{5}{6}$, as well as `walk` and `accept` (i.e., pay the list price). In turns 3 and 5, the agent chooses from among the following fractional concessions: $\frac{1}{6}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}$, along with `walk`, and `accept`. In turn 7, the agent chooses between `walk` and `accept`. As with the agent seller, we do not allow the agent buyer to include a message with its offer.

Input features

The agent observes all of the features that the seller observes except for those that cannot be observed by the buyer. These excluded features comprise 1) the features that describe activity between the seller and other buyers (i.e., on other threads), 2) the thread count, 3) the features that pertain to the automatic thresholds chosen by the seller, and 4) the count of the seller's prior listings and Best Offer listings. (The buyer observes the seller's feedback score.) Features that have not been determined before the agent acts (i.e., the offer in the current turn or attributes of future turns) are inputted as 0.

Architecture

The architecture of the agent buyer is identical to that of the agent seller.

Training

The training process is identical to that described above for the agent seller, except that the agent replaces the offer models for one randomly chosen buyer (as described above). Training one buyer agent takes less than one day.

Payoffs

If the agent buyer purchases the item, it receives the value of the item less the sale price; otherwise it receives 0. We train a range of agents, each of which values items at a given multiple of the list price, denoted λ.

Evaluation

We evaluate the agent buyer by simulating each listing in the holdout partition 10 times, with the agent acting as a randomly selected buyer, using the arrival process described above. The analyses in the paper compare the agent's behavior in these simulations to that of randomly selected human sellers in the same partition.

To create this comparison set of human buyers, we draw one buyer uniformly at random from each listing in the data. If the listing expires, this is straightforward. However, if the item sells before the expiration time, we need to account for the possibility that a randomly selected buyer from the set of *potential* arrivals may not have arrived before the listing ended. To do so, we use the arrival models to simulate buyer arrivals in the period between the end of the listing and the unreached expiration. We sample a buyer, uniformly

at random, from the combined set of observed and simulated arrivals. If an observed thread is chosen, that thread enters into the comparison set; otherwise, the comparison set does not include a thread from the focal listing.

Figure A4 compares distributions of arrival times and buyer number between the agent simulations and the comparison set. In both cases, the distributions are essentially identical.

References

Backus, Matthew, Thomas Blake, Brad Larsen, and Steven Tadelis (2020a) "Sequential bargaining in the field: Evidence from millions of online bargaining interactions," *The Quarterly Journal of Economics*, Vol. 135, pp. 1319–1361.

Backus, Matthew, Thomas Blake, Jett Pettus, and Steven Tadelis (2020b) "Communication and bargaining breakdown: An empirical analysis," *NBER*.

Kingma, Diederik P. and Jimmy Ba (2015) "Adam: A Method for Stochastic Optimization," in Yoshua Bengio and Yann LeCun eds. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kleinberg, Jon, Himabindu Lakkaraju, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan (2018) "Human decisions and machine predictions," *The Quarterly Journal of Economics*, Vol. 133, pp. 237–293.

Mnih, Volodymyr, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu (2016) "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, PMLR.

Stooke, Adam and Pieter Abbeel (2019) "rlpyt: A research code base for deep reinforcement learning in pytorch," *arXiv preprint arXiv:1909.01500*.

G Figures

Figure A1: Two-dimensional representation of the categorical space. Dots are categories, sized by the number of listings. Colors and labels correspond to meta-categories.

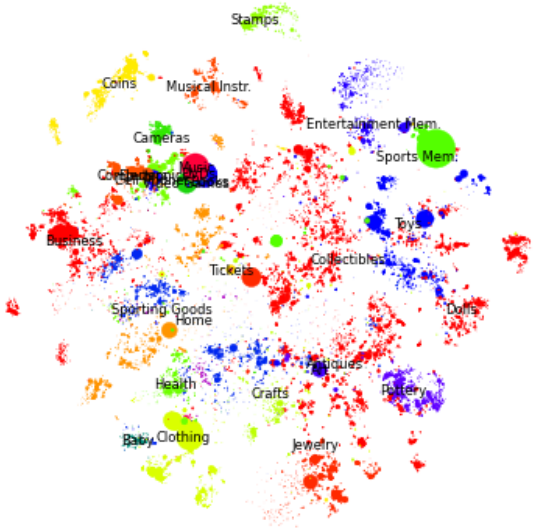


Figure A2: Distributions of response times, for seller turns.

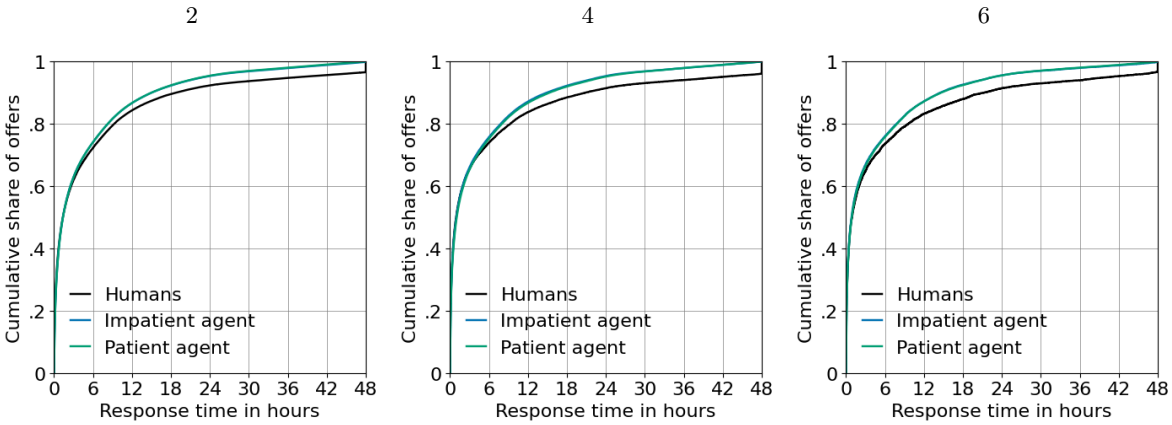


Figure A3: Distributions of response times, for buyer turns.

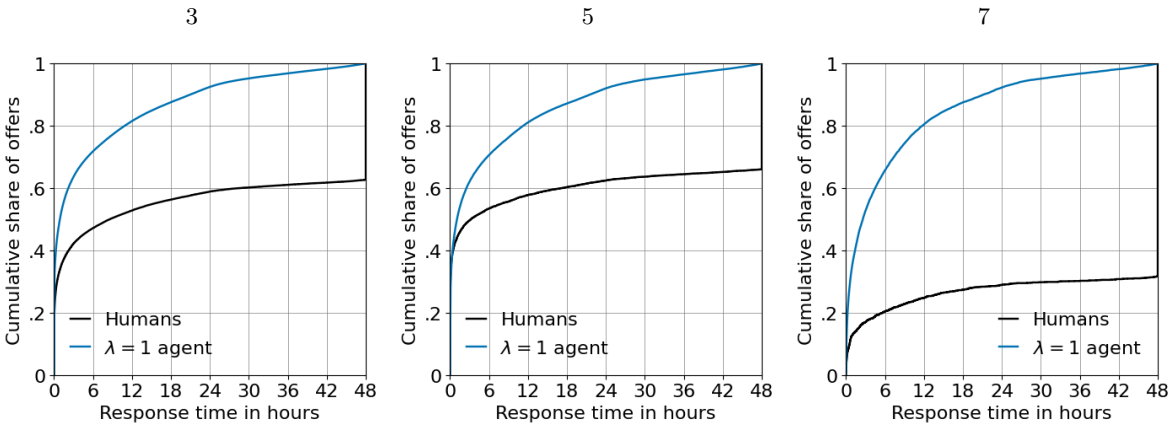


Figure A4: Distributions of buyer number (left) and arrival time (right), in the agent's simulations and in the comparison set of human buyers.

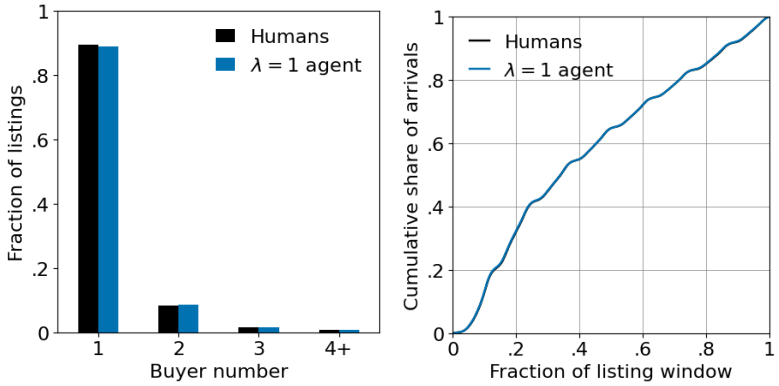


Figure A5: Distributions of item values, normalized by the list price and prior to discounting.

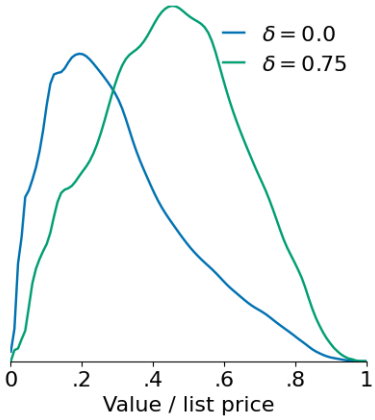


Figure A6: Probability of a sale (left) and average sale price (right), by seller experience.

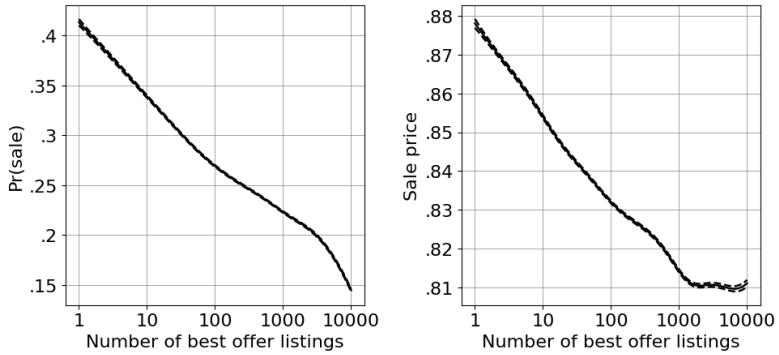


Figure A7: Distribution of the first buyer's arrival time (left) and time between subsequent arrivals (right).

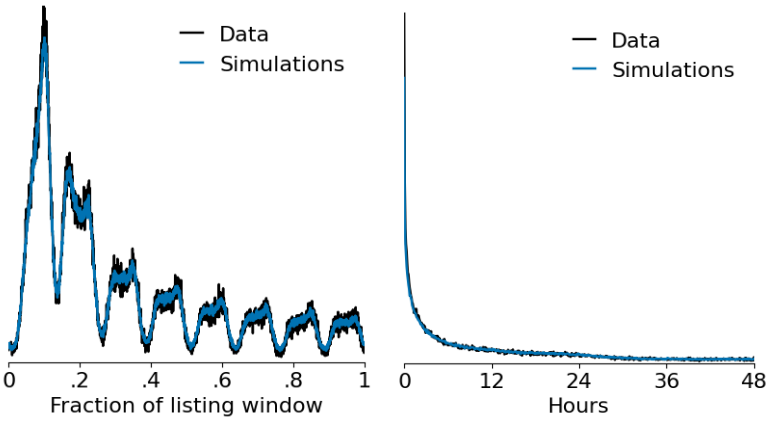


Figure A8: Distributions of buyer experience.

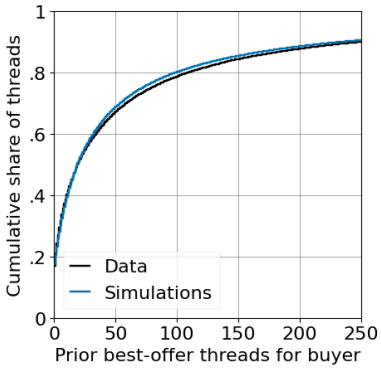


Figure A9: Distributions of concessions, by turn.

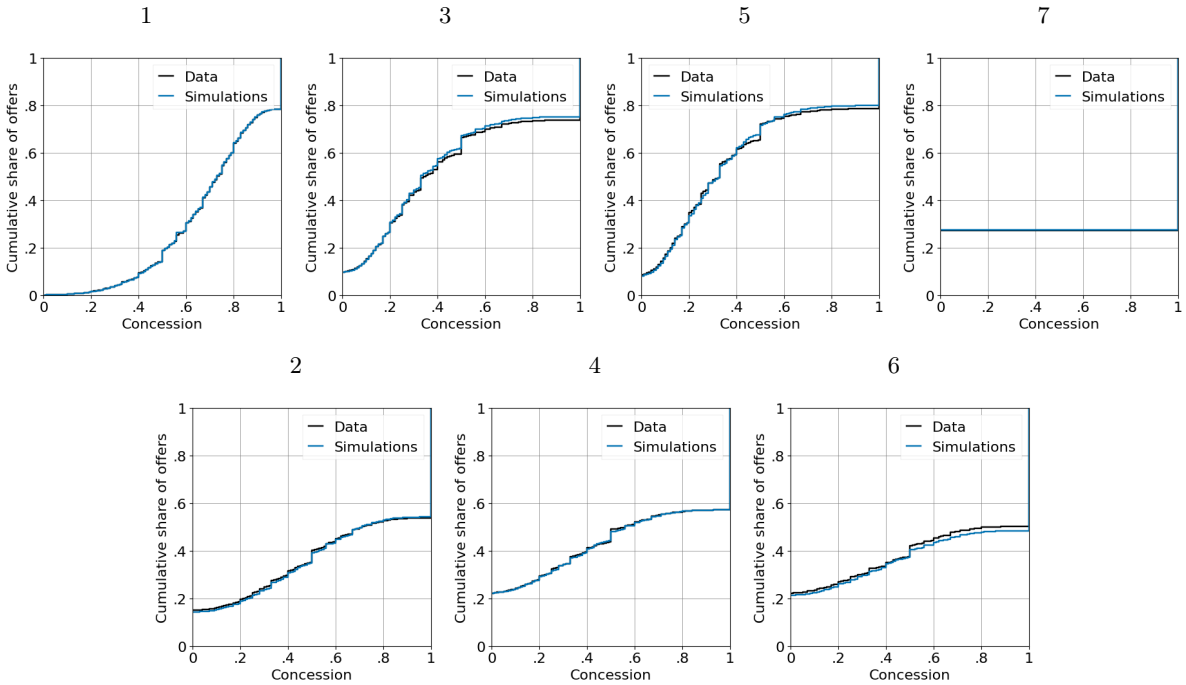


Figure A10: Distributions of response times, by turn.

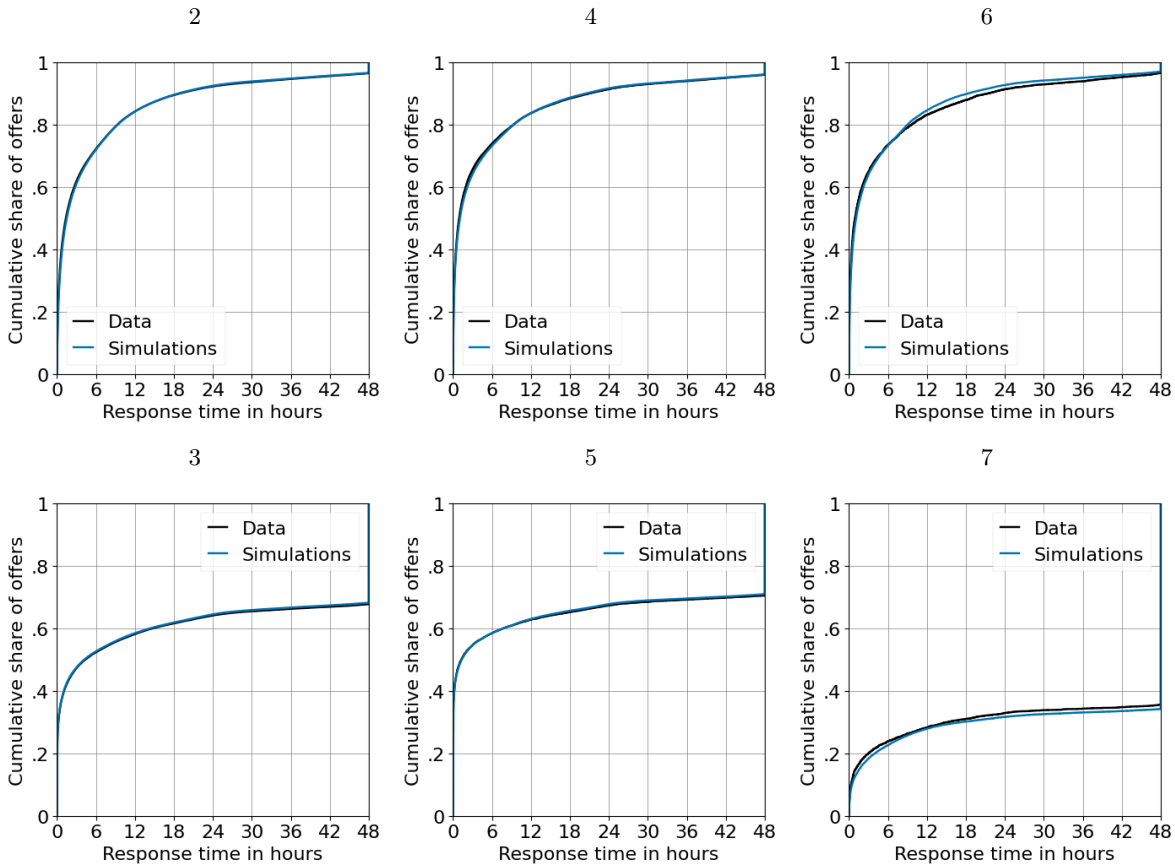
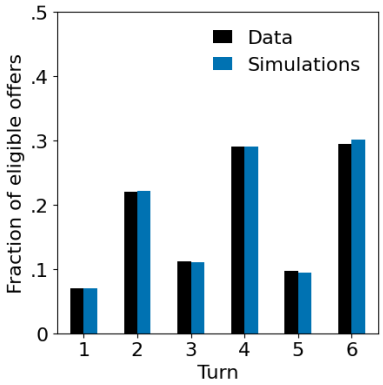


Figure A11: Message rate for eligible offers, by turn.



Note: Acceptances, buyer rejections, and automatic seller offers are not eligible for messages.

Figure A12: Distributions of threads per listing (left) and turns per thread (right).

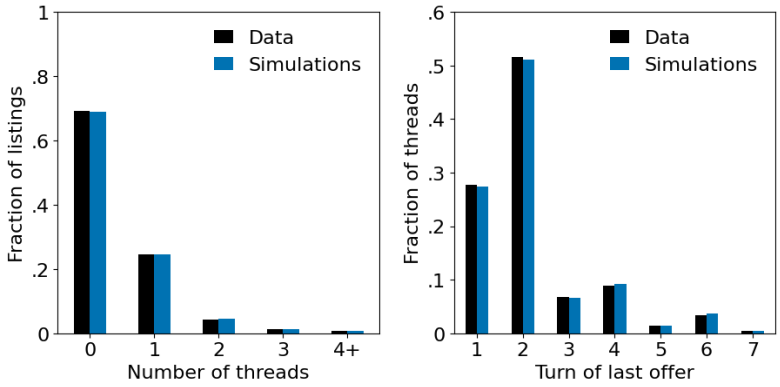


Figure A13: Distributions of days to sale (left), sale price (middle), and normalized sale price (right).

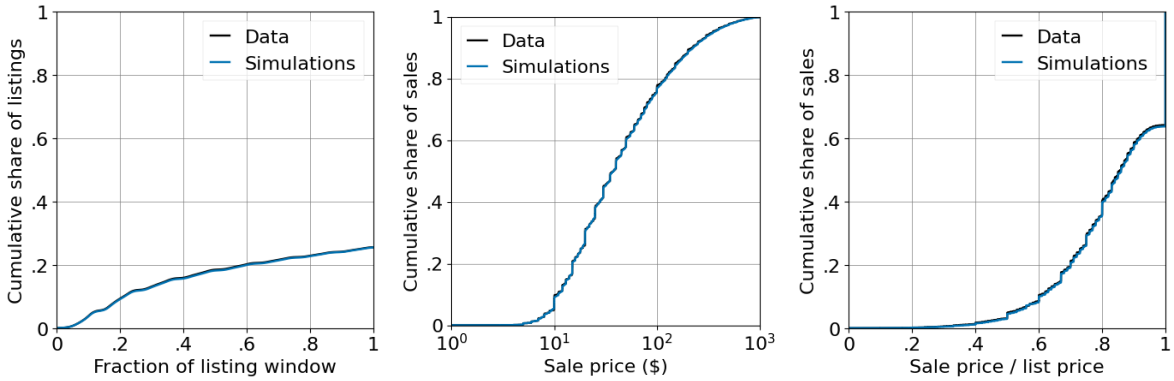


Figure A14: Comparison between collectibles (top row) and other listings (bottom row). Collectibles attract fewer buyers, are associated with longer threads, are less likely to sell, and sell for the list price less often when they do sell. The simulator recreates these patterns.

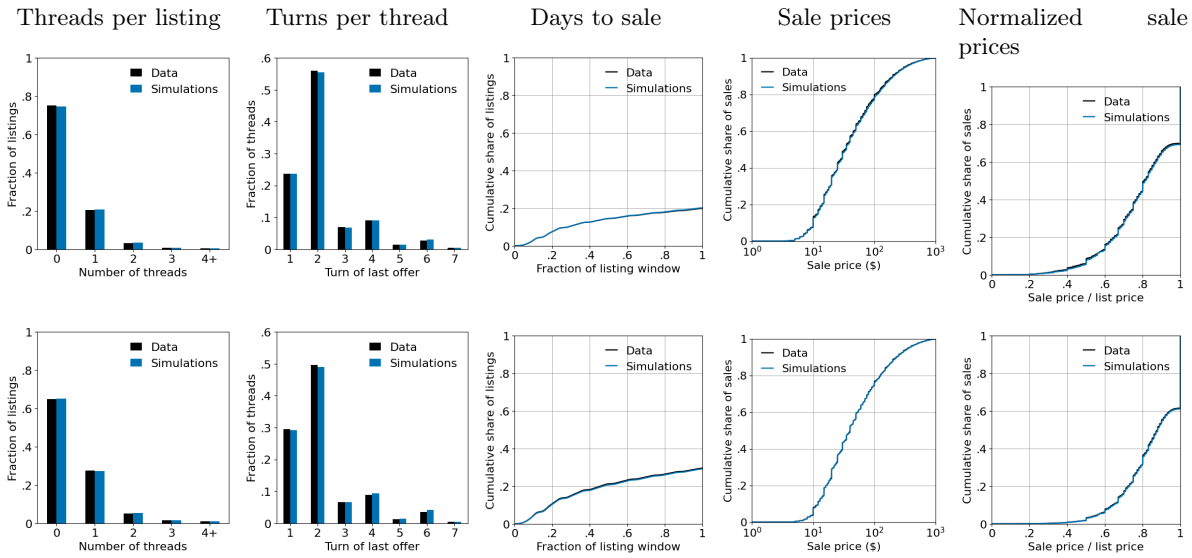


Figure A15: Discriminator ROC curve.

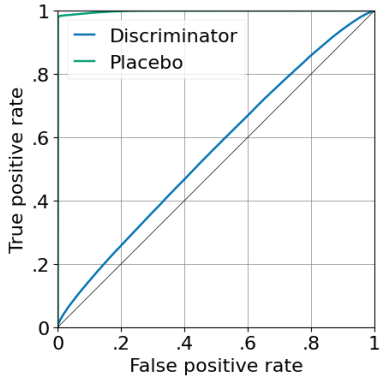
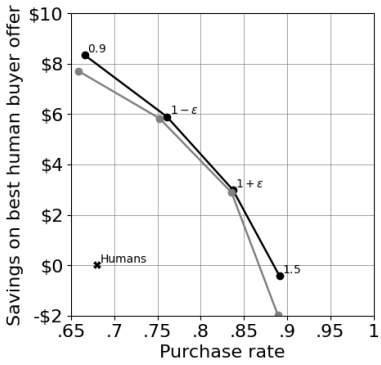


Figure A16: Savings on the best buyer offer in the data.



Note: Restricted to listings in the data for which at least one buyer arrives.

Figure A17: Distributions of market values.

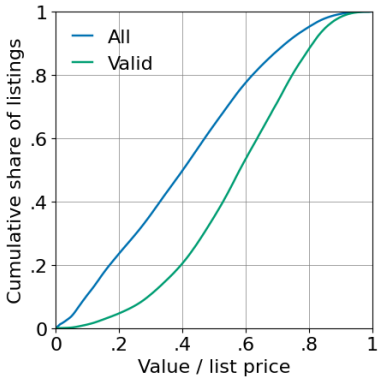


Figure A18: Distributions of market values for unsold listings.

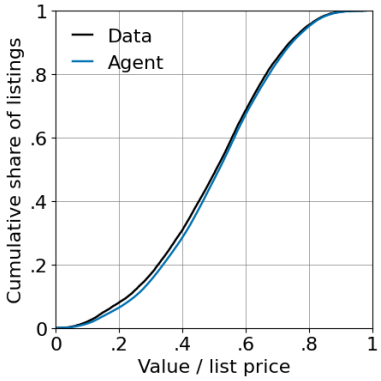


Figure A19: Cumulative distribution of normalized sale prices.

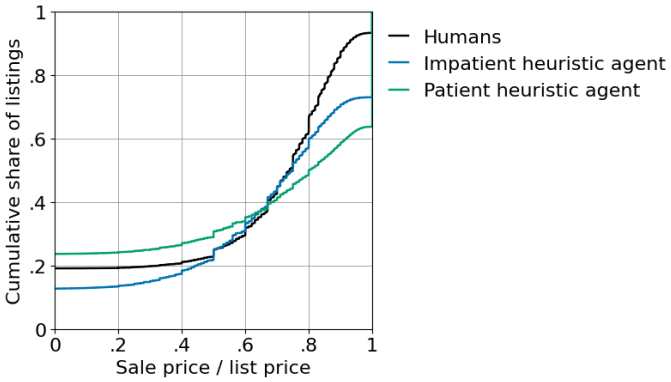


Figure A20: Distributions of buyer offers, by turn.

(a) Buyer's first offer **(b)** Buyer's second offer

Figure A21: Rates at which sellers reject first offers.

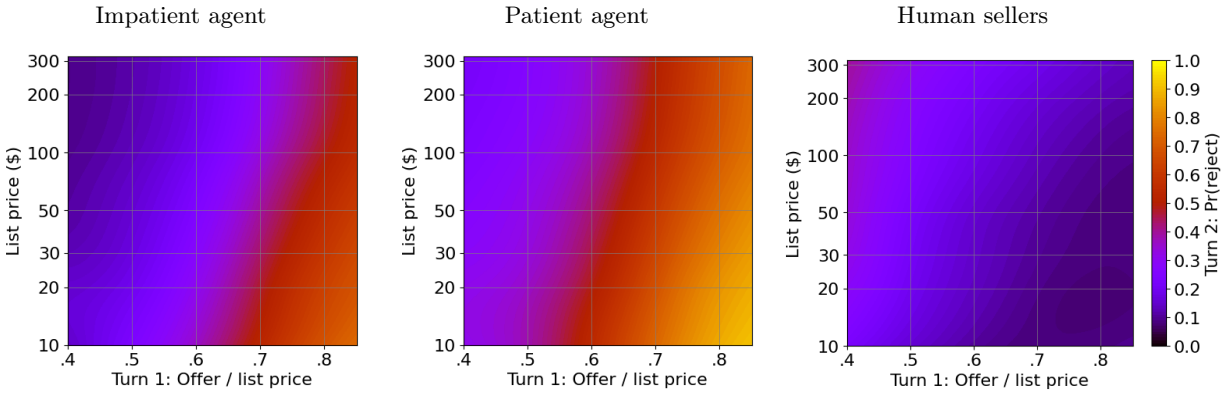


Figure A22: Rate at which buyers pay full price after the seller rejects the first offer.

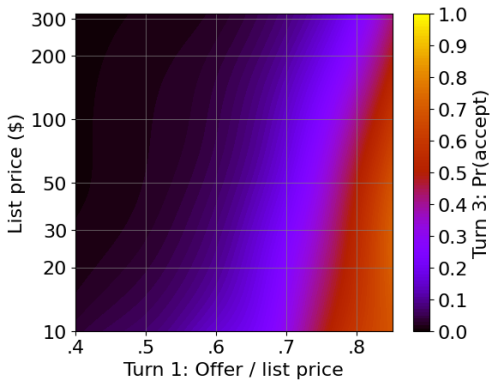
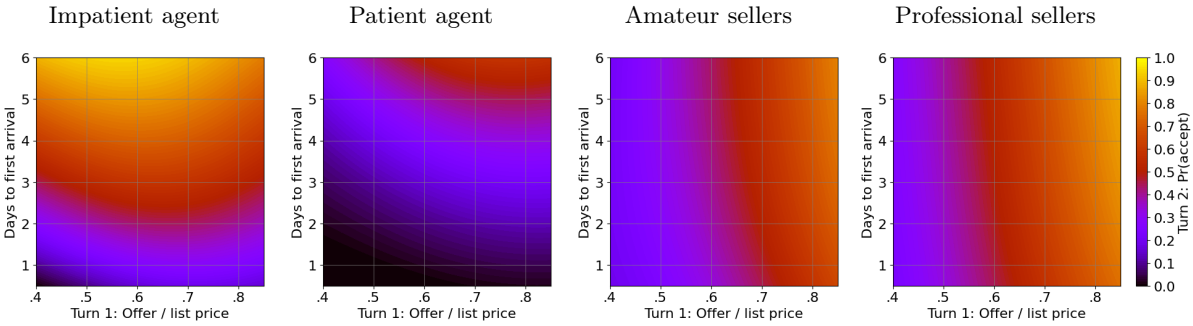


Figure A23: Rates at which agents (left two panels) and humans (right two panels) accept first offers.



Note: Professional and amateur sellers are those with and without an eBay store, respectively.