The Theory is Predictive, but is it Complete? An Application to Human Perception of Randomness

Jon Kleinberg* Annie Liang† Sendhil Mullainathan‡

April 23, 2019

Abstract

When testing a theory, we should ask not just whether its predictions match what we see in the data, but also about its "completeness": how much of the predictable variation in the data does the theory capture? Defining completeness is conceptually challenging, but we show how methods based on machine learning can provide tractable measures of completeness. We also identify a model domain—the human perception and generation of randomness—where measures of completeness can be feasibly analyzed; from these measures we discover there is significant structure in the problem that existing theories have yet to capture.

When we test theories, it is common to focus on what one might call their *predictiveness*: do the predictions of the theory match what we see in the data? For example, suppose we have a theory of the labor market that says that a person's wages depend on their knowledge and capabilities. We can test this theory by looking at whether more education indeed predicts higher wages in labor data. Finding this relationship would provide evidence in support of the theory, but little guidance

*Department of Computer Science, Cornell University
†Department of Economics, University of Pennsylvania
‡Department of Economics, University of Chicago

towards whether an alternative theory may be even more predictive. The question of whether more predictive theories might exist—and how much more predictive they might be—points toward a second issue, distinct from predictiveness, which we will refer to as *completeness*: how close is the performance of a given theory to the best performance that is achievable in the domain? In other words, how much of the predictable variation in the data is captured by the theory?

At a conceptual level, completeness is an important construct because it lets us ask how much room there is for improving the predictive performance of existing theories. Simultaneously, it helps us to evaluate the predictive performance that has already been achieved. This evaluation is not straightforward, because the same level of predictive accuracy can have very different meanings in different problems—for example, an accuracy of 55% is strikingly successful for predicting a (discretized) stock movement based on past returns, but extremely weak for predicting the (discretized) movement of a planet based on the relevant physical measurements.[1] These two problems differ in the best achievable prediction performance they permit, and so the same quantitative level of predictive accuracy must be interpreted differently in the two domains.

One way to view the contrast between these two problem domains is as follows. In each case, an instance i of the prediction problem consists of a vector x_i of measured features or covariates, and a hidden outcome y_i that must be predicted. In the case of astronomical bodies, we believe that the measured features—mass, position, velocity, and the corresponding values for nearby bodies—are sufficient to make highly accurate predictions over short time scales. In the case of stock prices, the measured features—past prices and returns—are only a small fraction of the information that we believe may be relevant to future prices. Thus, the variation in stock movements *conditioned on the features we know* is large, while planetary motions are well predicted by known features.

The point then is that prediction error represents a composite of two things: first, the opportunity for a better model; and second, intrinsic noise in the problem due to the limitations of the feature set. If we want to understand how much room there is for improving the predictive performance of existing theories within a given

[1] For example, the planet's mass and position, and the masses and positions of all large nearby bodies.

domain—holding constant the set of features that we know how to measure—we need a way to separate these two effects.

We demonstrate below a non-trivial social science domain—human perception and generation of randomness—in which this activity is feasible. Because it is possible to search the space of predictive models to optimality, we can construct a benchmark for the best achievable level of prediction, and use this to evaluate the predictive performances of leading economic theories in the domain. Relative to this benchmark, we find that these existing theories explain roughly 13-15% of the explainable variation in experimental data.

The closest paper to ours is Peysakhovich and Naecker (2017), which concurrently proposed use of machine learning to establish a benchmark against which to compare the performance of existing theories. Peysakhovich and Naecker (2017) considers prediction of choices under uncertainty and under ambiguity, and uses regularized regression to construct a benchmark. Because we focus on a setting in which it is possible to optimize over the complete class of models, our algorithm discovers the best achievable level of prediction for the feature set, while regularized regression is itself a model which may be incomplete relative to the best achievable level. In practice, methods like regularized regression and other more scalable machine learning algorithms may be effective ways to construct a benchmark in cases in which the feature space is large. The two papers thus collectively point to the potential for machine learning methods to deliver insight into theory completeness in various social science domains.

1 Testbed: Human Generation of Coin Flips

1.1 Background

The problem we consider is one with a long history of research in psychology and behavioral economics: human generation of random sequences. It is well documented that humans misperceive randomness (Bar-Hillel and Wagenaar, 1991; Tversky and Kahneman, 1971), and that this fact is significant not only for its basic psychological interest, but also for the ways in which misperception of randomness manifests itself in a variety of contexts: for example, investors' judgment of sequences of (random)

stock returns (Barberis et al., 1998), and professional decision-makers' reluctance to choose the same (correct) option multiple times in succession (Chen et al., 2016), and people's execution of a mixed strategy in a game (Batzilis et al., 2016).

A common experimental framework in this area is to ask human participants to generate fixed-length strings of k (pseudo-)random coin flips, for some small value of k (e.g. $k = 8$), and then to compare the produced distribution over length-k strings to the output of a Bernoulli process that generates realizations from $\{H, T\}$ independently and uniformly at random (Rapaport and Budescu, 1997; Nickerson and Butler, 2009). Following in this tradition, we use the platform Mechanical Turk to collect a large dataset of human-generated strings designed to simulate the output of a *Bernoulli(0.5) process*, in which each symbol in the string is generated from $\{H, T\}$ independently and uniformly at random.

1.2 Data

Our data set consists of 29,375 binary strings of length eight, generated as if the realizations of eight consecutive flips of a fair coin. To incentive effort, we told subjects that payment would be approved only if their (set of) strings could not be identified as human-generated with high confidence.[2,3]

Despite these incentives, some subjects did not attempt to mimic a random process, generating for example the same string in each of the fifty rounds. We thus chose to remove all subjects who repeated any string in more than five rounds.[4] This selection eliminated 167 subjects and 7,400 strings, yielding a final dataset with 471 subjects and 21,975 strings. We check that our main results are not too sensitive

[2]In one experiment, 537 subjects each whom produced 50 binary strings of length eight. In a second experiment, an additional 101 subjects were asked to each generate 25 binary strings of length eight.

[3]Subjects were informed: "To encourage effort in this task, we have developed an algorithm (based on previous Mechanical Turkers) that detects human-generated coin flips from computer-generated coin flips. You are approved for payment only if our computer is not able to identify your flips as human-generated with high confidence."

[4]This cutoff was selected by looking at how often each subject generated any given string, and finding the average "highest frequency" across subjects. This turned out to be 10% of the strings, or five strings. Thus, our selection criteria removes all subjects whose highest frequency was above average.

to this selection criteria by considering two alternative choices in Appendix B—first, keeping only the initial 25 strings generated by all subjects, and then, removing the subjects whose strings are "most different" from a Bernoulli process under a χ^2-test. We find very similar results under these alternative criteria.

Throughout this paper, we identify Heads with '1' and Tails with '0,' so that each string is an element in $\{1,0\}^8$. The 21,975 strings are aggregated into a single dataset, which induces a distribution over $\{1,0\}^8$. This observed human distribution over strings turns out to be statistically different from a true Bernoulli(0.5) process: we can reject the hypothesis that the data is generated from a uniform distribution over $\{1,0\}^8$ under a χ^2-test with $p < 10^{-4}$.[5]

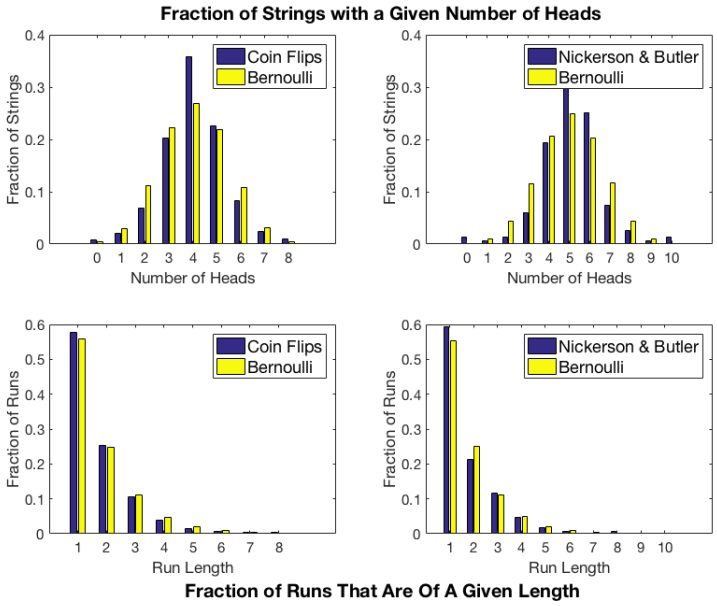


Figure 1: (a) Top row: the fraction of generated strings that include m Heads, where m is the label on the x-axis. *Left*—comparison of our MTurk data (purple) with simulated Bernoulli strings (yellow); *Right*—comparison of Nickerson & Butler (2009) data (purple) with simulated Bernoulli strings (yellow). (b) Bottom row: the fraction of runs that are of length m, where m is the label on the x-axis. *Left*—comparison of our MTurk data (purple) with simulated Bernoulli strings (yellow); *Right*—comparison of Nickerson & Butler (2009) data (purple) with simulated Bernoulli strings (yellow).

[5]This suggests also that our subjects are not in fact using external randomizing devices.

Moreover, the ways in which the observed distribution over strings differs from a Bernoulli process are consistent with the literature (Rapaport and Budescu, 1997; Nickerson and Butler, 2009). For example, subjects exhibit an over-tendency to alternate (52.68% of flips are different from the previous flip, as compared to an expected 50% in a Bernoulli(0.5) process), an under-tendency to generate strings with "extreme" ratios of Heads to Tails (see the top row of Figure 1), and an under-tendency to generate strings with long runs (see the bottom row of Figure 1).

1.3 Existing Models

Several frameworks have been proposed for modeling human misperception of randomness, and we will consider two influential approaches proposed in Rabin (2002) and Rabin and Vayanos (2010). Although both of these frameworks are models of mistaken *inference* from data, and not directly models of human generation of random sequences, they are easily adapted to our setting, as we will discuss below.[6]

In Rabin (2002), subjects in the underlying model observe independent, identically distributed (i.i.d.) signals, but mistakenly believe the signals to be negatively autocorrelated. Specifically, subjects observe a sequence of i.i.d. draws from a Bernoulli(θ) distribution, where $\theta \in [0, 1]$ is an unknown rate drawn from distribution π. Although subjects know the correct distribution π over the Bernoulli parameter θ, they have a mistaken belief about the way in which the realized rate θ determines the signal process. Subjects believe that the observed signals are instead drawn *without replacement* from an urn containing θN '1' signals and $(1-\theta)N$ '0' signals, so that a signal of '1' is less likely following observation of '0', and vice versa. For tractability, it is additionally assumed that the urn is "refreshed" every other round, meaning that the composition is returned to its original composition of θN '1' signals and $(1-\theta)N$ '0' signals.

To use this model in our setting, we modify it in two ways: first, since subjects are informed that the coin they should mimic is fair, we fix the prior distribution π over rates so that subjects believe $\theta = 0.5$ with certainty; second, we relax the assumption that the urn is refreshed deterministically every other round, adding a

[6]These adaptations are consistent with comments made in the original papers, in the context of relating these models to the empirical literature discussed in Section 1.2.

second parameter $p \in [0, 1]$, which determines the probability that the urn is refreshed. Thus, in the revised model, subjects generate random sequences by drawing without replacement from an urn that is initially composed of $0.5N$ '1' balls and $0.5N$ '0' balls, and is subsequently refreshed with probability p before every draw.

Rabin and Vayanos (2010) introduces a second framework for modeling human misperception of randomness. The following simple version of their model can be applied to predicting generation of random sequences: each subject generates flip s_1 according a Bernoulli(0.5) distribution, and then each subsequent flip s_k according to

$$s_k \sim \text{Ber}\left(0.5 - \alpha \sum_{t=0}^{k-2} \delta^t (2 \cdot s_{k-t-1} - 1)\right),$$

where the parameter $\delta \in \mathbb{R}_+$ captures a (decaying) influence of past flips, and the parameter $\alpha \in \mathbb{R}_+$ measures the strength of negative autocorrelation.[7] Notice that past realizations of '1' reduce the probability that the k-th flip is '1', and past realizations of '0' increase this probability. Thus, like the previous model, Rabin and Vayanos (2010) predicts generation of negatively autocorrelated sequences.

1.4 Prediction Tasks

We test these theories by looking at how well they predict the dataset of human-generated strings described in Section 1.2. We consider two tests. In the first test, which we call *Continuation*, we ask how well we can predict a subject's eighth flip given the first seven flips. A prediction rule for this problem is any function

$$f : \{0, 1\}^7 \to [0, 1] \tag{1}$$

that maps the initial seven flips into a probability that the next flip is '1'. Given a test dataset $\{s^i\}_{i=1}^n$ of n strings, we evaluate the error of the prediction rule f using:

$$\frac{1}{n} \sum_{i=1}^{n} \left(s_8^i - f(s_{1:7}^i)\right)^2.$$

where s_8 is the eighth flip in string s, and $f(s_{1:7})$ is the predicted probability that the eighth flip is '1' given initial sequence $s_{1:7}$. This loss function, *mean-squared error*,

[7] We make a small modification on the Rabin and Vayanos (2010) model, allowing $\alpha, \delta \in \mathbb{R}_+$ instead of $\alpha, \delta \in [0, 1)$.

penalizes (quadratic) distance from the best prediction. Notice that if subjects are truly generating strings from an i.i.d. Bernoulli(0.5) distribution, then no prediction rule can improve in expectation upon a prediction error of 0.25.

In the second test, which we call *Classification*, we seek to separate strings generated by human subjects from strings generated by a Bernoulli(0.5) process. A prediction rule in this problem is any map

$$c : \{0,1\}^8 \to [0,1] \tag{2}$$

from strings of length eight into a probability that the string was generated by a human subject. Given a test dataset $\{s^i\}_{i=1}^n$ of n strings, we evaluate error by producing an equal number of Bernoulli strings, and finding

$$\frac{1}{2n}\sum_{i=1}^{2n}\left(c^i - c\left(s^i\right)\right)^2,$$

where $c^i = 1$ if the true source of generation for string s^i was a human subject, and $c^i = 0$ otherwise. As above, if the human-generated strings are consistent with a Bernoulli(0.5) process, then we cannot improve on an expected prediction error of 0.25.

If we assume that strings are generated according to either of the models described in Section 1.3, then there is a "best" prediction rule that minimizes expected prediction error. We can therefore test these models by examining how well their prediction rules perform in Continuation and Classification. Specifically, we estimate the free parameters of these models on training data and report their out-of-sample prediction errors (tenfold cross-validated) in Table 1.[8]

Throughout, we compare these errors with a naive baseline that corresponds to random guessing—that is, we predict that the next flip is '1' with probability 0.5 for all initial substrings in the Continuation task, and we classify each string as human-generated with probability 0.5 in the Classification task. We find that the Rabin

[8]We randomly partition the data into ten equally-sized subsets, estimate the free parameters of the model on nine subsets (the training set), and predict the strings in the tenth (the test set). The reported prediction error is an average over the ten possible choices of the test set (from the ten folds), and the reported standard error is the standard deviation of the prediction errors across the test sets. This is a common approximation to the standard error for a cross-validated loss.

(2002) and Rabin and Vayanos (2010) models are predictive: their prediction errors are between 0.2486 and 0.2494, all of which improve on the error of 0.25 that we would obtain by random guessing.

Table 1: Rabin (2002) and Rabin and Vayanos (2010) are predictive: they improve upon the prediction error achieved by guessing at random.

	Continuation	Classification
Naive	0.25	0.25
Rabin (2002)	0.2494	0.2489
	(0.0007)	(0.0008)
Rabin and Vayanos (2010)	0.2492	0.2488
	(0.0007)	(0.0006)

But the *margin* of improvement over guessing at random is very small (no larger than 0.0014), and the gap between the best prediction errors and a perfect zero is large. How should we interpret the achieved reductions in prediction error?

To answer this, we need a benchmark against which to evaluate the prediction errors. Crucially, this benchmark should not be perfect prediction: deviations from a true i.i.d. process make it possible to improve upon the naive baseline of 0.25, but the observed process is far from deterministic. Conditioning on initial flips alone, there is a limit to how well we can hope to predict in these problems. A more suitable benchmark is then the *best possible* prediction error—we propose now an approach for finding this.

1.5 Benchmark

Our proposed approach for constructing a benchmark for this problem is to use a *table lookup* algorithm, in which we enumerate all 2^k binary strings and record the empirical frequency of each string. Given enough samples, this table of empirical frequencies approximates the "human distribution" over the full set of strings. And from this table, we can derive optimal predictions for both the Continuation and

Classification problems.[9]

Definition 1 (Table Lookup)**.** *Let g be the empirical distribution over strings in the training data. The table lookup continuation rule is*

$$f_{TL}(s) = \frac{g(s1)}{g(s)} \quad \forall\ s \in \{1,0\}^7. \tag{3}$$

where '$s1$' is the concatenation of the string s and the outcome '1'. The table lookup classification rule is

$$c_{TL}(s) = \frac{g(s)}{g(s) + 1/256} \quad \forall\ s \in \{1,0\}^8.$$

In the Continuation task, the table lookup prediction rule assigns to every string $s \in \{1,0\}^7$ the empirical frequency with which s is followed by '1' in the training data. In the Classification task, the table lookup prediction rule compares the empirical frequency of generation of string s to the theoretical frequency of generation of string s in a Bernoulli process.

Notice that the table lookup Continuation rule has 2^7 free parameters (corresponding to the 2^7 unique strings of length seven), and the table lookup Classification rule has 2^8 free parameters (corresponding to the 2^8 unique strings of length eight). With over 21,000 observed strings, we have enough observations per unique string to densely populate each cell of the lookup table. The table lookup prediction errors thus approximate the best possible prediction errors in these problems.

Table 2 reports the (tenfold cross-validated) prediction errors achieved by table lookup, and uses these errors as benchmarks against which to compare the prediction errors achieved by the behavioral models discussed above.

We find that table lookup achieves a prediction error of 0.2439 in the Continuation task and 0.2422 in the Classification task. This performance of table lookup is far worse than perfect prediction, showing that there is a large amount of irreducible noise in the problem of predicting human-generated coin flips. Naively comparing achieved prediction error against perfect prediction (which would suggest a completeness measure of at most 0.4%) thus grossly misrepresents the performance of the existing theories.

[9]The table lookup prediction error is a consistent estimator for the *irreducible error* in the problem, also known as the *Bayes error rate*.

Table 2: Comparison of prediction errors achieved using existing models with prediction errors achieved using table lookup. The behavioral models explain up to 15% of the explainable variation in the data.

	Continuation		Classification	
	Error	Completeness	Error	Completeness
Bernoulli	0.25	0	0.25	0
Rabin (2002)	0.2494 (0.0007)	0.10	0.2489 (0.0008)	0.14
Rabin & Vayanos (2010)	0.2492 (0.0007)	0.13	0.2488 (0.0006)	0.15
Table Lookup	0.2439 (0.0019)	1	0.2422 (0.0010)	1

A more appropriate notion of the achievable performance in this problem is the error achieved using table lookup. The errors of 0.2439 and 0.2422 above represent the predictive limits of the problems: conditioning only on initial flips, it is not possible to reduce prediction error from the naive baseline by more than 0.0061 in Continuation and 0.0078 in Classification. We propose as a simple measure of the *completeness* of the existing theories, then, the ratio of the reduction in prediction error achieved by the best behavioral model (relative to the naive baseline) to the reduction achieved by table lookup (again relative to the naive baseline). In the Continuation task, we find the completeness of the Rabin (2002) and Rabin and Vayanos (2010) models to be up to 13%, and in the Classification task, we find the completeness of these models to be up to 15%.[10] These results suggest that existing models produce between 13-15% of the achievable improvement in prediction error.

1.6 The Predictive Power of a Feature Set

Recall that table lookup quantifies the best achievable accuracy *for a fixed feature set*. When we measure completeness, we thus separate two reasons why the model does not achieve perfect prediction—suboptimal use of the available features, versus

[10] For example, the completeness of the Rabin and Vayanos (2010) model in the Continuation task is computed as $(0.25 - 0.2492)/(0.25 - 0.2439) = 0.13$.

basic limits to the predictive power of the feature set. In particular, completeness tells us how far prediction error can be reduced *without* the discovery or addition of new features. For example, in the Classification task, the Rabin and Vayanos (2010) error can be reduced from 0.2488 to 0.2422 by using initial flip data differently, but cannot be reduced beyond this unless the model takes into account features beyond initial flips.

The gap between 0.2422 and a perfect zero reflects instead basic predictive limitations of the feature set that we consider. We have so far explored use of table lookup as a way to evaluate completeness of a model given a fixed feature set, but an alternative use of the table lookup benchmark is as a measure of the predictiveness of that feature set. By comparing the table lookup errors of different feature sets, we can better understand "how predictive" those feature sets are.

We illustrate this potential comparison by considering a few simple alternative feature sets for our problem. Instead of utilizing the full feature set (2^8 strings for Classification and 2^7 strings for Continuation), we predict using various "compressed" table lookup algorithms built on different properties of the string, where strings of the same "type" are bucketed into the same bin. We consider three properties: *Number of Heads*—where we partition the set of strings based on the total number of Head flips—*Number of Alternations*—where we partition the set of strings based on the number of alternations in the string—and *Flips 4-8*—where we partition the set of strings based on outcomes including and after flip 4. Just as table lookup returned an approximation of the highest level of predictive accuracy using the full structure of initial flip data, these compressed table lookup algorithms approximate the highest level of predictive accuracy that is achievable using a particular kind of structure in the strings (e.g. number of Heads).

Table 3: Comparison of the value of various feature sets.

	Continuation		Classification	
	Error	Completeness	Error	Completeness
Bernoulli	0.25	0	0.25	0
Flips 4-8	0.2478 (0.0010)	0.36	0.2485 (0.0008)	0.19
Number of Alternations	0.2500 (0.0003)	0	0.2465 (0.0009)	0.45
Number of Heads	0.2464 (0.0009)	0.59	0.2459 (0.0008)	0.53
Full Table Lookup	0.2439 (0.0019)	1	0.2422 (0.0010)	1

We find that these simple feature sets attain large fractions of the achievable improvement. For example, using just the number of Heads as a feature achieves 59% of the improvement of our full table lookup in the Continuation problem, and 53% in the Classification problem.

In Table 3, the completeness measures for these alternative feature sets are reported relative to our main feature set, but we can in turn view that feature set as producing partial improvements relative to an even richer feature set. In the context of the generation and perception of randomness, it becomes interesting—both as a goal in itself and as a perspective on the limitations of our current feature set—to consider what might constitute a set of unmeasured features of the human participant's behavior that would significantly improve predictive accuracy if we chose to add them as columns to the table. For example, in the Continuation task, prediction accuracy may improve further if we add in subject ids as a feature (so that each row corresponds to a subject id and sequence of seven flips), or descriptions of the age and education level of the subject. We leave these questions to future work.

2 Applications of Completeness Measures in Other Domains

Finally, we ask whether our methodology can also be used to evaluate theory completeness in real-life settings where human perception of randomness is believed to play a role. We focus principally on two such settings: data on baseball umpires calls—where we seek to predict an umpire's current call given their recent $k-1$ calls—and data from repeated play of Rock-Paper-Scissors—where we seek to predict a player's choice given their choice in the previous $k-1$ matches. We first introduce the two settings below, and then compare completeness of the Rabin and Vayanos (2010) model across these domains.

Baseball Umpires. Baseball umpires determine whether pitches should be called as *balls* or *strikes* when the batter does not swing. While the definition of a strike is objective,[11] the judgment of whether or not the pitch constitutes a strike is not.

Chen et al. (2016) demonstrate that umpire calls exhibit negative auto-correlation: in aggregate, umpires are less likely to call a pitch a strike after calling the previous pitch a strike, and even less likely if they called the last two pitches strikes—thus, umpire's past calls are predictive of future calls. We explore further context-dependencies, asking how well one can hope to predict umpire calls using the five immediately prior calls. (Compared to Chen et al. (2016), our focus is on finding the limits of predictability, as opposed to characterizing the nature of this predictability.) We build a dataset of umpire calls from Chen et al. (2016)'s original dataset of 1.5 million pitches over 12,564 games by 127 different umpires, using all non-overlapping sequences of consecutive calls of length 6 that occurred within the same game.[12] This dataset includes 15,127 strings, where the frequency of strikes (across all pitches) is approximately 27%, and the mean frequency of strikes in the final call is 33%.

The Continuation task in this domain is to predict the final flip given the first five, and the Classification task is to separate strings of umpire calls from synthetic

[11] A designated strike zone takes the shape of a vertical right pentagonal prism located above home plate, and the umpire should call "strike" whenever the ball is within the strike zone as it passes the location of home plate, and "ball" otherwise.

[12] Consecutive calls are not separated by uncalled pitches.

Bernoulli strings, with the probability of '1' set to equal the average flip in the umpire data (0.27).

Rock Paper Scissors Our second domain involves games of Rock-Paper-Scissors played by Facebook users in 2007 on the Facebook app Roshambull. Each game consisted of two players and lasted until either player had won two matches.[13] Batzilis et al. (2016) consider a large dataset of play on this app (2,636,417 matches) and study how behavior in these matches deviates from Nash play—one of their key findings is that information shown to players at the start of a game regarding the history of opponent play is predictive of their first throw.

In a different predictive exercise, we ask how well one can predict a player's subsequent throws based on his initial throws (without conditioning on rich features such as information provided to players). To do this, we extract from Batzilis et al. (2016)'s dataset the initial consecutive six choices (for a given player) in all games that lasted at least six matches, and consider each of these a string.[14] This selection yields a total of 29,864 strings, where the overall frequency of Rock is 37.42%, Paper is 33.58%, and Scissors is 29.00%. Unlike the other domains studied in this paper, the Rock-Paper-Scissors strings in our dataset exhibit positive autocorrelation: the probability that a throw is followed by a different throw is 0.64, which is slightly less than the expected level of alternation given independent throws. When we apply Rabin and Vayanos (2010) for prediction, we therefore relax the constraint that the free parameters α, δ have positive values, so that the model serves as a general model of autocorrelated strings.

The Continuation task in this domain is to predict the final throw given the first five. A prediction rule maps strings in $\{r, p, s\}^5$ to probability vectors in Δ^2, where each coordinate corresponds to the probability of realization of r, p, or s. The realized throw is represented as a binary vector of length 3, which takes value '1' in the coordinate corresponding to the throw in that observation. We use the loss function $\frac{1}{n}\sum_{i=1}^{n} \|\mathbf{y}_i - \mathbf{q}_i\|_2$, where $\mathbf{q}_i$ is the predicted probability vector for observation i and $\mathbf{y}_i$ is the outcome. The Classification task is to separate strings of Rock-Paper-Scissors throws from synthetic strings of length 6, where each element is randomly drawn from

[13] The two winning matches need not be consecutive.

[14] The average game lasted 4.29 matches.

$\{r, p, s\}$.

Completeness. We now report in Table 4 the performance of the Rabin and Vayanos (2010) model in each of these two settings. As before, we report these performances next to a naive baseline. For the Classification tasks, we again use a naive prediction rule that classifies each string as human-generated with probability 0.5. The naive prediction rules for the Continuation task are, however, different from before: In the Rock-Paper-Scissors domain, we naively predict $(1/3, 1/3, 1/3)$ for all histories; and in the umpire domain, we predict for each string the unconditional probability of '1,' which turns out to be 0.27. As in our main setting, we find that the Rabin and Vayanos (2010) model improves upon these naive baselines.

	Baseball Umpires				RPS			
	Continuation		Classification		Continuation		Classification	
	Error	Complete	Error	Complete	Error	Complete	Error	Complete
Naive	0.221	0	0.25	0	0.817	0	0.25	0
	(0.003)		(–)		(–)		(–)	
RV	0.220	0.21	0.249	0.17	0.816	0.06	0.249	0.11
	(0.004)		(0.0005)		(0.0005)		(0.0003)	
TL	0.217	1	0.243	1	0.813	1	0.242	1
	(0.004)		(0.002)		(0.002)		(0.001)	

Table 4: We use table lookup to establish a benchmark for evaluating the completeness of Rabin and Vayanos (2010) in these new settings.

Also similar to our main setting, the gains of the Rabin and Vayanos (2010) model are small in *absolute value* and hard to interpret—no larger than 0.001 in any of the problems we look at. Constructing a table lookup benchmark allows us to better interpret these improvements. In particular, notice that the Rabin and Vayanos (2010) model is not only predictive in both domains, but reduces the naive prediction error by similar margins. When we construct the upper benchmarks for the two domains, we find that these margins of improvement mean different things in the two problems. In particular, the achieved improvement constitutes a larger fraction of the achievable improvement in the umpire domain (17-21%) than in the Rock-Paper-Scissors domain (6-11%). This conclusion is not surprising, since sequences

of umpire calls are approximately a direct test of the Rabin and Vayanos (2010) model, while Rock-Paper-Scissors involves (un-modeled) strategic considerations. But a straightforward comparison of prediction errors does not suffice to reveal that the improvement is more substantial in one domain than the other. Having table lookup as a baseline in both settings makes it possible to demonstrate this and to quantify the difference.

Moreover, the table lookup benchmark allows us to compare the *predictive limits* of the various problems. Focusing on the Classification task, we see that the table lookup error is strikingly similar across all domains—0.2433 in the baseball umpire setting, 0.2422 in our original experimental setting, and 0.2419 in the Rock-Paper-Scissors setting. Among these three settings, sequences of baseball umpire calls appear to be least predictable (closest to Bernoulli), and the sequences of Rock-Paper-Scissors throws most predictable.

3 Conclusion

When evaluating the predictive performance of a theory, it is important to know not just whether the theory is predictive, but also how complete its predictive performance is. To assess theory completeness, we need a notion of what constitutes the best *achievable* predictive performance for a given problem. We demonstrate a domain in which it is possible discover this best achievable level of performance, and show how this benchmark can help us to evaluate the performance of existing models.

Besides the problem domain we consider here, table lookup is useful in other social science problems. For example, Fudenberg and Liang (2018) apply this approach to evaluate predictions of initial play in 3×3 matrix games. The "predictive limit" is difficult to guess ex-ante: if play in most games is close to degenerate on a single action, then (approximate) perfect prediction is feasible; on the other hand, if play is close to uniform over the actions, then it will be hard to improve over random guessing. Table lookup thus again offers a way to quantify the performance of various models; for certain prediction tasks, the completeness of existing models in that setting turns out to be as high as 80%.

Finally, we note that all the tests mentioned so far involve training and testing models on data drawn from the same "domain." An interesting question for subse-

quent work would be to compare the transferability of models across domains. Indeed, we may expect that economic models that are outperformed by machine learning models in a given domain have higher transfer performance outside of the domain. In this sense, completeness may provide an incomplete measure of the "generalizability" of the model, and we leave development of such notions to future work.

References

Bar-Hillel, M. and W. Wagenaar (1991): "The Perception of Randomness," *Advances in Applied Mathematics*.

Barberis, N., A. Shleifer, and R. Vishny (1998): "A Model of Investor Sentiment," *Journal of Financial Economics*.

Batzilis, D., S. Jaffe, S. Levitt, J. A. List, and J. Picel (2016): "How Facebook Can Deepen our Understanding of Behavior in Strategic Settings: Evidence from a Million Rock-Paper-Scissors Games," Working Paper.

Chen, D., K. Shue, and T. Moskowitz (2016): "Decision-Making under the Gambler's Fallacy: Evidence from Asylum Judges, Loan Officers, and Baseball Umpires," *Quarterly Journal of Economics*.

Fudenberg, D. and A. Liang (2018): "Predicting and Understanding Initial Play," Working Paper.

Nickerson, R. and S. Butler (2009): "On Producing Random Sequences," *American Journal of Psychology*.

Peysakhovich, A. and J. Naecker (2017): "Using Methods from Machine Learning to Evaluate Behavioral Models of Choice Under Risk and Ambiguity," *Journal of Economic Behavior and Organization*.

Rabin, M. (2002): "Inference by Believers in the Law of Small Numbers," *The Quarterly Journal of Economics*.

Rabin, M. and D. Vayanos (2010): "The Gambler's and Hot-Hand Fallacies: Theory and Applications," *Review of Economic Studies*.

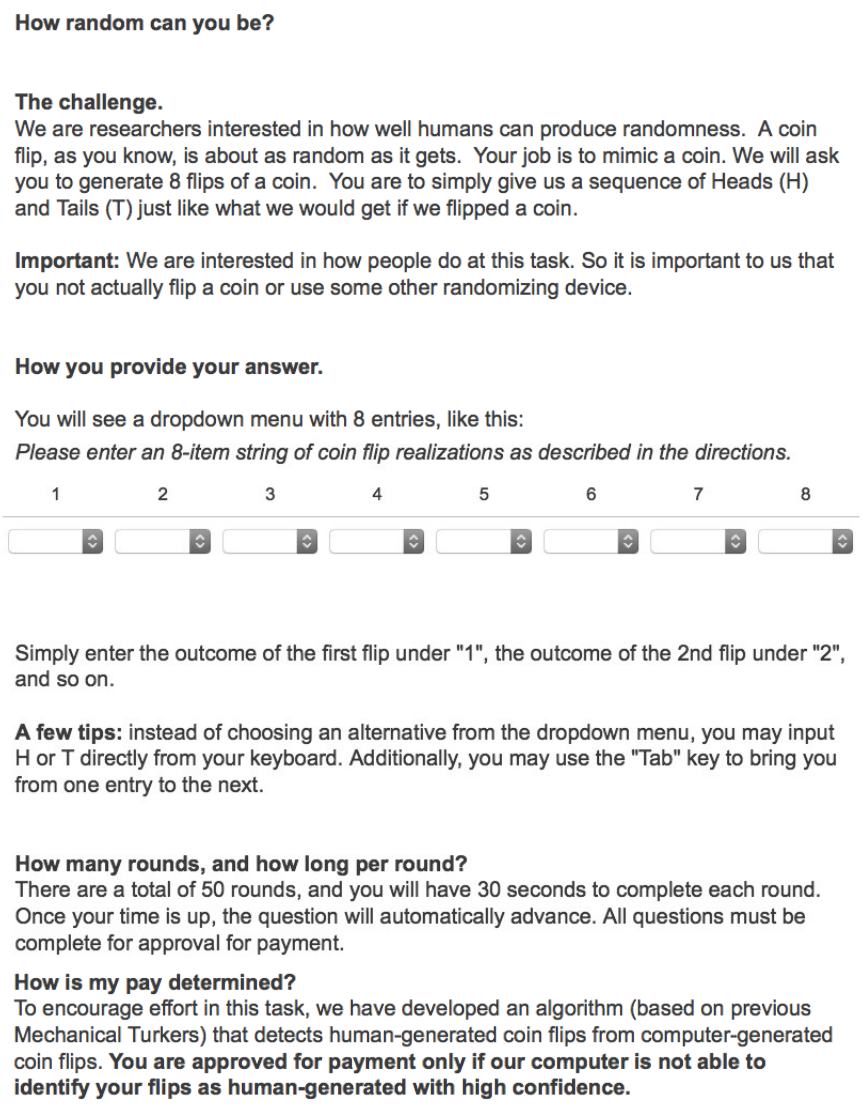
Rapaport, A. and D. Budescu (1997): "Randomization in Individual Choice Behavior," *Psychological Review*.

Tversky, A. and D. Kahneman (1971): "The Belief in the Law of Small Numbers," *Psychological Bulletin*.

Appendix

A Experimental Instructions

Subjects on Mechanical Turk were presented with the following introduction screen:



How random can you be?

The challenge.
We are researchers interested in how well humans can produce randomness. A coin flip, as you know, is about as random as it gets. Your job is to mimic a coin. We will ask you to generate 8 flips of a coin. You are to simply give us a sequence of Heads (H) and Tails (T) just like what we would get if we flipped a coin.

Important: We are interested in how people do at this task. So it is important to us that you not actually flip a coin or use some other randomizing device.

How you provide your answer.

You will see a dropdown menu with 8 entries, like this:

Please enter an 8-item string of coin flip realizations as described in the directions.

1 2 3 4 5 6 7 8

Simply enter the outcome of the first flip under "1", the outcome of the 2nd flip under "2", and so on.

A few tips: instead of choosing an alternative from the dropdown menu, you may input H or T directly from your keyboard. Additionally, you may use the "Tab" key to bring you from one entry to the next.

How many rounds, and how long per round?
There are a total of 50 rounds, and you will have 30 seconds to complete each round. Once your time is up, the question will automatically advance. All questions must be complete for approval for payment.

How is my pay determined?
To encourage effort in this task, we have developed an algorithm (based on previous Mechanical Turkers) that detects human-generated coin flips from computer-generated coin flips. **You are approved for payment only if our computer is not able to identify your flips as human-generated with high confidence.**

B Different Cuts of the Data

We repeat the main analysis in Section 1 using alternative cuts of the data.

Only initial strings. We consider a cut of the data in which we keep all subjects, but use only their first 25 strings. This selection accounts for potential fatigue in generation of the final strings, and leaves a total of 638 subjects and 15,950 strings. Prediction results for our main exercise are shown below using this alternative selection.

	Continuation		Classification	
	Error	Completeness	Error	Completeness
Bernoulli	0.25	0	0.25	0
Rabin & Vayanos (2010)	0.2491 (0.0008)	0.05	0.2480 (0.0006)	0.15
Table Lookup	0.2326 (0.0030)	1	0.2367 (0.0030)	1

Chi-Squared Test. For each subject, we conduct a Chi-squared test for the null hypothesis that their strings were generated under a Bernoulli process. We order subjects by p-values and remove the 100 subjects with the lowest p-values (subjects whose generated strings were most different from what we would expect under a Bernoulli process). This leaves a total of 538 subjects and 24,550 strings. Prediction results for our main exercise are shown below using this alternative selection.

	Continuation		Classification	
	Error	Completeness	Error	Completeness
Bernoulli	0.25	0	0.25	0
Rabin & Vayanos (2010)	0.2491 (0.0005)	0.12	0.2487 (0.0005)	0.15
Table Lookup	0.2427 (0.0016)	1	0.2415 (0.0009)	1