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

Jon Kleinberg* Annie Liang† Sendhil Mullainathan‡

August 9, 2017

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, Harvard University

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, and a hidden label 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.

The challenge is that it is generally very difficult to evaluate the best predictive performance achievable in a given domain. Are there non-trivial problem domains in which this activity is feasible? In other words, are there settings that simultaneously (i) contain complex structure and a rich line of published theories but (ii) are also tractable enough that we can establish a benchmark of optimal predictive accuracy for purposes of comparison?

0.1 A Model Domain: Human Generation of Randomness

In this paper, we identify such a domain and study theories in it from the perspective of completeness. 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).

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). We consider the following two natural prediction tasks:

- *Continuation:* We take a string of k coin flips created by a human participant trying to simulate a Bernoulli process, reveal the first $k-1$ flips, and ask for a prediction of the k^{th} flip.

- *Classification:* We take a set of n strings created by a human participant trying to simulate a Bernoulli process, and n strings created by a Bernoulli process,

and try to classify each string based on the source (human or Bernoulli) that produced it.

A number of influential theories in behavioral economics provide methods for estimating the probability that different strings are generated by a human source, and hence lead to predictions for these problems (Rabin, 2002; Rabin and Vayanos, 2010).

What is striking is that despite the richness of the underlying questions, the Continuation and Classification problems are behavioral-science questions where the benchmark of optimal prediction can in fact be feasibly computed. Optimal predictions for this problem can be made via *table lookup*, in which we enumerate all 2^k strings s consisting of 0's and 1's, and for each such string s we record the empirical fraction $g(s)$ of human-generated strings in our sample that are equal to s. With enough samples, this converges to the *human distribution* over the full set of strings. And from this table of empirical frequencies, it is easy to derive optimal predictions for both the Continuation and Classification problems. For Continuation, this is based on looking at the relative frequency of s followed by 0 versus s followed by 1, where s is the length-$(k-1)$ prefix we observe; for Classification, this is based on looking at the human frequency $g(s)$ relative to the Bernoulli probability 2^{-k} for a given length-k string s.

Our analysis in this paper, based on table lookup as a benchmark for optimal prediction, thus has a dual motivation. First, we will uncover a number of new findings about our substantive domain, the human perception of randomness. Second, we are able to undertake a case study of theory completeness for a rich problem, as discussed at the outset of the paper: given existing theories and a benchmark for optimal prediction, we can see how close to optimality the existing theories come, and how this gap varies for different settings of the question. We believe that there are a number of domains in the behavioral sciences where "narrow" feature sets will make this type of baseline possible. The concurrent work of Peysakhovich and Naecker (2017), which uses machine learning as a benchmark for behavioral theories of risk and ambiguity, points to further potential for this argument.

0.2 Overview of the Analysis

We begin by considering a set of human-generated strings of length-8 over the alphabet $\{H, T\}$ (for "heads" and "tails"), produced by participants on Mechanical Turk. For both the Continuation and Classification prediction problems, we consider methods that output predictions consisting of probabilities in the interval $[0, 1]$, which are then evaluated relative to the true label of 0 or 1. We use mean-squared error as our evaluation; thus, predicting a probability of 0.5 for all instances would yield an error of 0.25.

We find that the existing behavioral models are *predictive*: they attain a mean-squared error of 0.249, which improves (to a statistically significant extent) on the error of 0.250 that we would obtain by random guessing. They are not, however, *complete*. Table lookup attains an error of 0.243, and relative to this benchmark, the existing models achieve roughly 15% of the maximum achievable gain over naive guessing for the problem. Thus, there is predictable structure in the problem that has not been fully captured by the existing models.

We then use this domain to consider two broad lines of questions related to our notion of theory completeness. The first is a question of what explains the improvement of table lookup over the behavioral models. In particular, when we say that human-constructed theories only achieve a relatively small fraction of the available performance gain over naive guessing, is this (a) because they are not using crucial features of the problem, or (b) because they are not combining them effectively? To separate these possibilities, we take a set of human-constructed features based on research in the area, and apply standard machine algorithms to learn combinations of these features for prediction. We find that in fact these algorithms come close to the performance of table lookup. Moreover, a substantial amount of the improvement over the behavioral models persists even when these algorithms are restricted to use of only a small number of features (comparable to the number of free parameters in the behavioral models). These results suggest that the answer to the question above may be more (b) than (a)—the research community approximately knows the "right" features for the problem, but may not be combining them as effectively as the machine learning algorithms for the goal of prediction.

These results bear also on the feasibility of the table lookup benchmark. While this set of tasks related to the human perception of randomness made it possible to

construct the benchmark of optimal prediction explicitly, in many domains it will be infeasible in general to construct a perfect benchmark. The performance of machine learning methods such as Lasso regression and decision trees for our task suggests that in some domains, scalable algorithms come close to the performance of table lookup, and may serve as reasonable proxies for the optimal benchmark. (Here too we find support in the results of Peysakhovich and Naecker (2017).)

The second question pertains to the robustness of these results to small variations in the framing of the task. In particular, does table lookup succeed by capturing specific features of the generation of length-8 strings of heads/tails that do not generalize even to closely related problems? To address this question, we build a table-lookup predictor using the original data of length-8 coin flips, and then we use this predictor for strings generated in a set of related but non-identical domains. Specifically, we set up prediction problems using binary alphabets other than H and T, and strings of different lengths (using seven flips to predict one additional flip at different indices in the string). We find that in these modified prediction problems, the existing models produce up to 22% of the improvement in prediction error obtained using table lookup, suggesting that the benchmark and ratio discovered previously are indeed stable across local problem domains.

0.3 Applications to Field Data

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.

The first is a task involving sequential decision-making—specifically, data on baseball umpires calling balls and strikes. Chen et al. (2016) find that umpire calls are negatively auto-correlated: in aggregate, umpires tend to avoid long runs of the same call (i.e. calling many strikes in a row or many balls in a row). Within this setting, we ask: knowing only an umpire's most recent $k-1$ calls, how well can we predict the current call?

Our second field study uses data from repeated play of Rock-Paper-Scissors on the Facebook app Roshambull, which was collected by Batzilis et al. (2016). Each unit of observation is a game, where a game consists of a sequence of matches that

conclude when one of the players wins two matches. In this setting, we ask: knowing only the choices (rock, paper, or scissors) that a player made in his or her first $k-1$ matches, how well can we predict the choice in the current match?

In both problems, we find that table lookup can achieve significant gains over naive guessing. Moreover, when we evaluate the completeness of the model based on Rabin and Vayanos (2010), we find its completeness in both domains to be qualitatively similar to what we obtained in our basic experimental framework. This shows that the completeness of the model is relatively stable across domains that are quite different, and in all cases there is significant room for theories to achieve stronger predictive gains.

Taken together, our results suggest that (1) there is a significant amount of structure in the problem of predicting human generation of randomness that existing models have yet to capture, and (2) our approach via the optimal predictive benchmark allows for evaluation of the completeness of theories in the given domain. Such an approach can be applied more generally in settings where this benchmark can be feasibly determined or approximated.

1 Main Testbed: Human Generation of Coin Flips

1.1 Data

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. Our main experiment includes 537 subjects, each of whom produced 50 binary strings of length eight, attempting to generate them as if these strings were the realizations of 50 experiments in which a fair coin was flipped eight times. In a second experiment, an additional 101 subjects were asked to each generate 25 binary strings of length eight. The task was described to subjects using the text below:

> 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.

To discourage use of an external randomizing device, we gave subjects 30 seconds to generate each string. 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] The complete set of directions can be found in Appendix A.

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. In response to this, we removed all subjects who repeated any string in more than five rounds.[3] 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 to this selection criteria by considering two alternative choices in Appendix C.1—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 object in $\{1,0\}^8$. The 21,975 strings are aggregated into a single dataset, which induces an empirical 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 $p \approx 0$.[4]

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

[2]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."

[3]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.

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

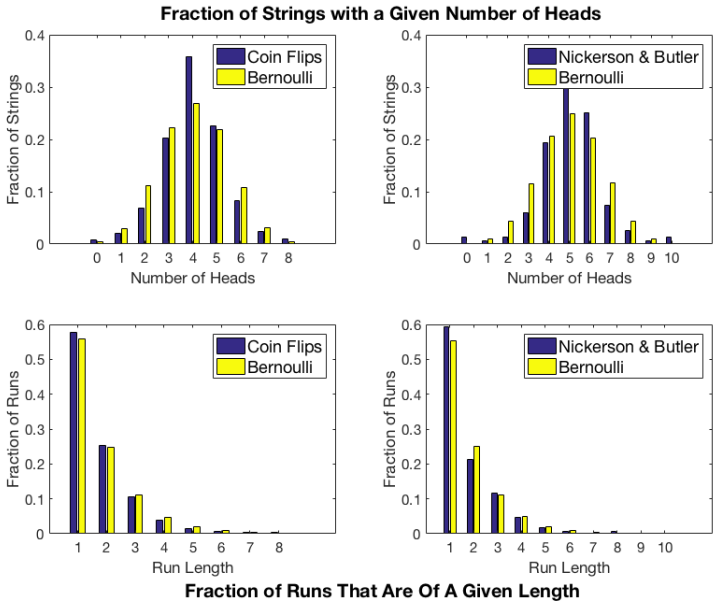


Figure 1: (a) Top row: the fraction of generated strings that include m Heads, where m is the label on the x-axis. *Left*—comparison of MTurk data (purple) with simulated Bernoulli strings (yellow); *Right*—comparison of Nickerson & Butler (2009) data (purple) with simulated Bernoulli strings (yellow).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

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).[5]

1.2 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

[5]See Section A.1 in the online appendix for further comparisons with the literature.

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 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

[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 1.1.

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

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.

1.3 Prediction Tasks

We test these theories by looking at how well they predict the dataset of human-generated strings described in Section 1.1. 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*, 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.

As a brief remark, we note that these tests are different from one another: a model of human generation can perform well on one prediction problem and poorly on the other. This is because Continuation asks how well we can predict the probability that a string ends in '1,' *conditional* on the first seven entries in the string, while Classification asks how well we can predict the *unconditional* distribution over entire strings. For example, if the human distribution over strings were to contain non-Bernoulli structure in its first few entries, but were essentially uniform over the final entry independently of what had preceded it, then it would be possible to perform well in Classification but not in Continuation.

If we assume that strings are generated according to either of the models described in Section 1.2, then there is a "best" prediction rule that minimizes expected prediction error (see Appendix B for more detail). 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 (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 0.250 that we would obtain by random guessing. This improvement is statistically significant in both problems for the model based on Rabin and Vayanos (2010).

[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.

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. Based on the numbers in Table 1 alone, it is difficult to evaluate the significance of these improvements. 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.

1.4 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]

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

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. Thus, the table lookup prediction errors approximate the best possible prediction errors in these problems. See Section A.3 in the online appendix for more detail.

We emphasize that these are the best possible prediction errors *conditioning on initial flips alone*; for example, if subject ids are available, then prediction error can be reduced by learning different models for different subjects, and if timing data is available, then prediction error can be further reduced by conditioning on speed of response. In the main text, we focus on the question of how to construct a benchmark for a *fixed* feature set (initial flips).[10] We discuss subsequently in Section 5 how to compare and interpret benchmarks across different feature sets.

Table 2 reports the (tenfold cross-validated) prediction errors achieved by table lookup. These errors are then used as benchmarks against which to compare the prediction errors achieved using the behavioral models discussed above.

[10]This is a natural choice for evaluating the completeness of the Rabin (2002) and Rabin and Vayanos (2010) models, which also condition only on initial flips.

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	0.10	0.2489	0.14
	(0.0007)		(0.0008)	
Rabin & Vayanos (2010)	0.2492	0.13	0.2488	0.15
	(0.0007)		(0.0006)	
Table Lookup	0.2439	1	0.2422	1
	(0.0019)		(0.0010)	

We find that table lookup achieves a prediction error of 0.2439 in the Continuation task and 0.2422 in the Classification task. The 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. This emphasizes that naively comparing achieved prediction error to perfect prediction can, and in this case does, misrepresent the performance of the existing theories.

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%.[11] These results suggest that existing models produce between 13-15%

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

of the achievable improvement in prediction error.[12]

2 Features Versus Combination Rules

The previous section found that human-constructed theories achieve only a fraction of the achievable performance gain over naive guessing. We ask now whether the limitations of existing behavioral models relative to table lookup arise because: (a) the behavioral models miss crucial predictive properties of the initial flips, or (b) they use the "right" features, but do not combine them as effectively for prediction.

To distinguish between these possibilities, we construct a feature space based on the existing models and related literature, and apply standard machine learning algorithms (Lasso regression and decision trees) to learn rules for combining these known features. We find that these algorithms predict significantly better than the behavioral models, and in fact closely approximate the performance of table lookup. Moreover, a substantial amount of the improvement over the behavioral models persists even when these scalable algorithms are restricted to use of only a small number of features (comparable to the number of free parameters in the behavioral models). This suggests that the reason in (b) accounts for at least a part of the gap between the behavioral models and table lookup: alternative models based on similar features can substantially improve performance.

We then ask whether the use of additional properties of the initial flips, not yet captured in existing models, can further improve predictive performance. Towards this goal, we define a rich set of "atheoretical" features. Each binary feature corresponds to a possible substring pattern, and takes value '1' when the pattern appears in the initial flips. We use machine learning algorithms to discover the most predictive features from this rich set of patterns, and then predict based on combinations of these features. Use of the decision tree algorithm with this feature space repre-

[12] As a robustness check, we repeat this exercise in Section A.2 of the online appendix for different string lengths. For Continuation, this means using flips 1 through $k-1$ to predict the k-th flip, where k varies from 2 to 7. For Classification, this means separating length-k Bernoulli strings from the first k flips generated by a human, where k varies from 2 to 7. We find that prediction accuracy roughly increases in the length of the string (so that conditioning on a larger number of initial flips results in better prediction of the subsequent flip), but neither the errors nor the measures of completeness vary significantly for lengths near $k = 8$.

sents a compression of the table lookup predictor—instead of assigning a prediction to each unique binary string, it partitions the space of strings, and learns a constant prediction for each partition element.

We find that prediction rules based on behavioral features are substantially more predictive than prediction rules based on an equal number of (best) atheoretical patterns. Moreover, when we combine the set of features, but continue to impose a restriction on the number of features, we find that only behavioral features are selected by the algorithms. This suggests that the behavioral features are more predictive than the best (small) subset of atheoretical patterns.[13]

These results collectively suggest that gap between table lookup and the behavioral models in our domain is better explained by (b) than (a)—the research community approximately knows the "right" features for the problem, but may not combine them as effectively for prediction as the machine learning algorithms.

2.1 Prediction Rules Based on Behavioral Features

We begin by constructing a feature space based on the relevant literature, including the following features: the proportion of alternation in the string (averaged across all flips); the total number of runs of length k in the string (for Classification, we allow k to vary from 2 to 8, and for Continuation, we allow k to vary from 2 to 7); the number of Heads in the string; the length of the longest run at the beginning of the string; the length of the longest run at the end of the string; and all of their pairwise interactions. This makes for 55 features in the Continuation task, and 66 features in the Classification task. Every binary string is recoded as a feature vector for each of the prediction tasks, so that prediction rules are maps from feature vectors to probabilities.[14]

We use two standard machine learning algorithms—Lasso regression and decision

[13]This comparison is not precise: the atheoretical features are binary-valued, while the behavioral features take values from a larger set, so the information content in the latter features is intrinsically larger.

[14]In the Continuation task, a prediction rule is a map from the set of feature vectors describing length-7 strings to a probability that the final flip is Heads. In the Classification task, a prediction rule is a map from the set of feature vectors describing length-8 strings to a probability that the string was generated by a human.

trees (see e.g. Hastie et al. (2009))—to select a prediction function based on the above features. From Table 3, we see that the cross-validated prediction errors obtained using these approaches closely approximate the table lookup prediction errors. In both problems, the best algorithm achieves 80-90% of the achievable reduction in prediction error. This means that for our domain, there is relatively little loss in using the best scalable machine learning algorithm (trained on behavioral features) as a substitute benchmark for table lookup.

Table 3: The performance of scalable algorithms approximates table lookup.

	Continuation		Classification	
	Error	Completeness	Error	Completeness
Naive	0.25	0	0.25	0
Lasso	0.2475 (0.0007)	0.41	0.2444 (0.0003)	0.72
Decision Tree	0.2443 (<0.0000)	0.93	0.2437 (<0.0000)	0.81
Table Lookup	0.2439 (0.0019)	1	0.2422 (0.0010)	1

These results suggest that new combinations of known features can yield large improvements in prediction. We turn next to considering whether combinations of *small* numbers of known features can also yield large improvements in prediction.

2.2 Restriction to a Small Number of Features

We again train prediction rules using the set of features described above, under a new constraint on the number of parameters. Below, we show the prediction errors obtained by decision trees that are restricted to k splits, where we consider $k = 2$ (thus comparing to the behavioral models), $k = 3$ and $k = 5$. (The $k = 2$ split decision trees can be found in Section B of the online appendix.) With some looseness, we will refer to "splits," "parameters," and "features" interchangeably.[15]

[15]The algorithm that we use constrains the number of "splits" in the decision tree; that is, the number of non-terminal nodes that branch according to a decision criterion. The greater the number of splits, the more dependencies are permitted, and hence the greater the number of degrees of

With two parameters, the best decision tree achieves 38-53% of the possible reduction in prediction error, and with five parameters, the best decision tree achieves up to 64% of the possible improvement. These results suggest that scalable algorithms can attain a substantial improvement on prediction error even when restricted to use of a very small number of features. (Recall for comparison that the best behavioral models achieved up to 15% of the table lookup improvement in these problems.)

Table 4: Machine learning algorithms built on behavioral features predict well even when restricted to use of a small number of features.

	Continuation		Classification	
	Error	Completeness	Error	Completeness
Naive	0.25	0	0.25	0
2 parameters	0.2477 (<0.0000)	0.38	0.2459 (<0.0000)	0.53
3 parameters	0.2470 (<0.0000)	0.49	0.2457 (<0.0000)	0.55
5 parameters	0.2461 (<0.0000)	0.64	0.2451 (<0.0000)	0.63
Table Lookup	0.2439 (0.0019)	1	0.2422 (0.0010)	1

Previously, we considered use of the best *unrestricted* machine learning algorithm as a substitute benchmark for table lookup. If we instead construct a benchmark using the performance of the best two-parameter decision tree—thus, comparing the existing models against a relatively interpretable model—we find that the completeness of the behavioral models is 35% in the Continuation task and 30% in the Classification task.[16] These results suggest that there is room for improvement in prediction error even when we condition on only a small number of interpretable features.

freedom in the model. The number of splits need not equal the number of features used in the model, since for example a single feature can be used to determine the decision criteria at multiple nodes. In the decision trees that we estimate for this problem, different features are in fact used at different branching nodes.

[16]Continuation: $(0.25 - 0.2492)/(0.25 - 0.2477) = 0.35$; Classification: $(0.25 - 0.2488)/(0.25 - 0.2459) = 0.29$.

2.3 Comparison with Algorithmic Features

We turn next to the question of whether it is possible to improve upon the performance of the algorithms above by discovering new features from a rich set of algorithmic patterns. To study this, we build a second feature space using features corresponding to the presence of specific patterns in the sequences of coin flips. We define a *pattern* p to be a length-k binary sequence over the alphabet $\{1, *\}$, and we say that the length-k string $s \in \{1, 0\}^k$ *contains* the length-k pattern $p \in \{1, *\}^k$ if $s_i = p_i$ at every index i where $p_i = 1$. Thus, for example, the string 1011011 contains the patterns $1{*}1{*}{*}{*}{*}$ and ${*}{*}{*}{*}{*}11$, but not the pattern $11{*}{*}{*}{*}{*}$.

We construct this second feature space by enumerating each substring pattern p (where $p \in \{1, *\}^7$ for the Continuation task and $p \in \{1, *\}^8$ for the Classification task), and defining indicator variables for the appearance of that pattern in each possible input string s (where $s \in \{1, 0\}^7$ for Continuation $s \in \{1, 0\}^8$ for Classification). We allow machine learning algorithms to select the $k = 2$, 3, and 5 most useful such features for prediction from among this set of pattern indicators, and we show the resulting prediction errors in Table 5. Again, the decision trees for $k = 2$ can be found in the appendix (see Figure 3).

Table 5: Prediction rules using a small number of algorithmic features perform worse than prediction rules using the same number of behavioral features.

	Continuation		Classification	
	Error	Completeness	Error	Completeness
Naive	0.25	0	0.25	0
2 parameters	0.2488 (0.0001)	0.20	0.2492 (0.0001)	0.10
3 parameters	0.2483 (0.0001)	0.28	0.2491 (0.0001)	0.11
5 parameters	0.2479 (0.0001)	0.34	0.2486 (0.0001)	0.18
Table Lookup	0.2439 (0.0019)	1	0.2422 (0.0010)	1

Table 5 shows that these prediction errors are higher than the corresponding errors in Table 4 for every prediction task and restriction on number of parameters that we

consider. These results provide further evidence that the gap between table lookup and the behavioral models does not necessarily imply that researchers are missing key predictive features—searching over a space of new syntactic patterns for strings does not produce improvements over the combinations of known features that we considered in Section 2.1.

3 Robustness of the Benchmark

In Section 1, we introduced table lookup as a means of approximating the best possible prediction error, and found that relative to the table lookup performance, existing models achieve approximately 13-15% of the achievable improvement in prediction error. Because table lookup is extremely flexible, however, it is possible that it learns a highly precise, but highly specific, model of human generation of coin flips of length-8. For example, it turns out that 57% of the strings in our dataset begin with "Heads." Table lookup learns this (domain-specific) asymmetry and uses it for prediction, but the probability of '1' in the first flip is fixed to be 0.5 in both the Rabin (2002) and Rabin and Vayanos (2010) models. This raises the question of to what extent the improvement of table lookup over existing models can be attributed to idiosyncratic features related to generation of length-8 coin flips.

To test the robustness of the table lookup predictor across domains, we ask how well this predictor can predict strings generated in a different but neighboring domain. We focus on *small* changes in the prediction domain, across which we would expect the behavioral models to be stable. If indeed table lookup adapts sensitively to fine details of the original context, we would expect its performance in these transfer prediction tasks to be weak, despite the similarity of the domains. This kind of "conceptual overfitting" represents the possibility for a richly-parametrized model to generalize poorly not because of insufficient data within-domain, but because of an insufficient sampling of contexts across which the parameters vary.

The new problem domains we consider are the following: first, we change the *alphabet* from which the realizations are drawn; second, we change the *length* of the flips to be predicted; finally, we consider prediction of the strings generated in the work of Nickerson and Butler (2009), which were produced under different conditions and by a different subject pool from our main dataset. For each of these, we train

prediction rules on the original dataset of length-8 coin flips, and then ask how well these rules predict strings that are generated under the new framing. We find that the transfer performance of table lookup is comparable to its "within-domain" performance in the previous section, in which the test and training data were generated from the same context. Moreover, the measure of completeness is also stable across local problem domains: existing models produce no more than 22% of the improvement in prediction error obtained by using table lookup for transfer prediction.

3.1 New Datasets

We briefly describe below strings generated in these new domains (see Appendix 3.2 for summary statistics describing these datasets).

New alphabet. The first transfer domain re-labels the outcome space from $\{H, T\}$ to $\{r, 2\}$. We asked 124 subjects on Mechanical Turk to generate 50 binary strings of length eight as if these strings were the realizations of 50 experiments in which a fair coin labelled 'r' on one side and '2' on another was flipped 8 times. This yielded a total of 6,200 strings. As in the main experiment, subjects were given only 30 seconds to complete each string, and payment was conditioned on whether their strings looked plausibly random. These properties of the experimental design are consistent across all of the experiments, so we will not repeat them below.

New length. The second domain changes the string length from eight to 15. We asked 120 subjects on Mechanical Turk to generate 25 binary strings of length fifteen as if these strings were the realizations of 25 experiments in which a fair coin was flipped 15 times. This yielded a total of 3,000 strings. We use these length-15 strings to construct seven datasets of length-8 strings, each including only flips k through $k + 7$, where $k \in \{2, \ldots, 8\}$.[17]

New subject pool. The final prediction problem uses data from Nickerson and Butler (2009), in which thirty Tufts undergraduates were asked to each produce 100 binary sequences, "as if 100 people had each tossed a coin 10 times, and the results had been recorded in a table of 100 rows and 10 columns, with each row corresponding to an individual."[18] As above, we construct truncated datasets of length-8 strings,

[17] We leave out $k = 1$, which would correspond to the original setting.

[18] We use a public version of this dataset that includes the responses of 28 subjects and a total of 2,800 strings.

including flips k through $k+7$, this time taking $k = 1, 2, 3$.

3.2 Transfer Prediction

We build a table lookup predictor from the original coin flip data described in Section 1.1, and use this to predict strings generated in the related domains described above. For prediction of strings from $\{r, 2\}^8$, the transfer Continuation task is to predict the final outcome from $\{r, 2\}$ given the first seven, and the transfer Classification task is to separate the human-generated strings from $\{r, 2\}^8$ from an equal number of Bernoulli strings. For prediction of substrings from the $\{H, T\}^{15}$ data, we treat each set of truncated strings as a different dataset, and define separate transfer Continuation and Classification problems for each cut. The prediction errors we report in these problems are averaged across the seven truncated datasets.

Throughout, we use the arbitrary convention that 'H' (in the original coin flip data) is identified with 'r' in the first transfer problem, and with 'H' in others.[19] Prediction errors obtained assuming the reverse mappings are presented in Tables 11-13 of the appendix, and do not differ substantially from those shown below. All errors presented in Tables 6 and 7 are mean-squared errors with bootstrapped standard errors.

Since the strings used for training and testing are generated from different distributions, it is possible for the prediction rules we consider to perform worse than guessing at random.[20] This turns out not to be the case: across each of the new prediction domains, the prediction rules we consider are predictive. Additionally, the prediction errors are comparable in absolute terms across domains: for example, table lookup achieves prediction errors ranging from 0.2325 to 0.2434 in these transfer problems. These prediction errors are similar to the previous errors of 0.2422 and 0.2439 in the original task.[21]

[19] Thus, the strings 'HTHTHTHT' and 'r2r2r2r2' are identically coded as '10101010'.

[20] For example, it may be that while strings exhibit negative autocorrelation in our primary setting, they exhibit positive autocorrelation in one of the new prediction domains.

[21] The lower prediction errors in the transfer tasks reflect settings in which there is less randomness in the strings to be predicted.

Table 6: Prediction errors in the *transfer Continuation problem*

	$\{r,2\}^8$		$\{H,T\}^{15}$		N-B (2009)	
	Error	Completeness	Error	Completeness	Error	Completeness
Naive	0.25	0	0.25	0	0.25	0
Rabin (2002)	0.2497 (0.0005)	0.05	0.2481 (0.0007)	0.14	0.2474 (0.0010)	0.15
R-V (2010)	0.2496 (0.0004)	0.06	0.2472 (0.0005)	0.20	0.2491 (0.0003)	0.05
Table Lookup	0.2434 (0.0011)	1	0.2361 (0.0018)	1	0.2325 (0.0023)	1

Table 7: Prediction errors in the *transfer Classification problem*

	$\{r,2\}^8$		$\{H,T\}^{15}$		N-B (2009)	
	Error	Completeness	Error	Completeness	Error	Completeness
Naive	0.25	0	0.25	0	0.25	0
Rabin (2002)	0.2497 (0.0009)	0.02	0.2476 (0.0006)	0.22	0.2496 (0.0003)	0.03
R-V (2010)	0.2495 (0.0003)	0.06	0.2479 (0.0006)	0.19	0.2497 (0.0004)	0.02
Table Lookup	0.2415 (0.0003)	1	0.2392 (0.0013)	1	0.2377 (0.0015)	1

Finally, the measures of completeness that we find in these transfer problems (not exceeding 22%) are not much larger than the within-domain estimates, as we would expect them to be if table lookup were indeed much less robust than the existing models. Thus, the table lookup benchmark and induced measures of completeness do appear to generalize across local changes in the problem domain.

4 Application of Approach in Field Domains

We now consider application of the proposed approach to testing theory completeness in two field domains. Our first field study uses data from Chen et al. (2016), and

consists of the sequential judgments made by baseball umpires regarding whether to call a pitch a *strike*. Our second field study uses data from Batzilis et al. (2016) and consists of the repeated decisions by Facebook players in online games of rock-papers-scissors. We represent the umpire data as binary sequences, where the outcome is whether the pitch is called as a *strike* or a *ball*. We represent the Rock-Paper-Scissors data as ternary sequences of sequential throws, where the outcome corresponds to whether the player chose rock, paper, or scissors.

In these settings, we study questions close to those considered in this paper so far: Given an initial set of calls by an umpire (or throws in Rock-Paper-Scissors), can we predict subsequent calls (throws)? And, as a classification task, given a set of strings, half of which correspond to sequences of umpire calls (or Rock-Paper-Scissors throws), and half of which correspond to independent realizations of a random variable, can we determine the source of generation?

We find that table lookup achieves a non-trivial improvement over naive guessing in both of these domains, and can again be used to construct a benchmark for attainable predictive accuracy in these problems. Interestingly, in both field domains, the model of Rabin and Vayanos (2010) achieves levels of predictive gain relative to this table lookup benchmark that are qualitatively similar to what we observed in our experimental data in Section 1.4. This suggests that the completeness of this theory is relatively stable despite wide differences in the nature of the domains.

4.1 Datasets

We now describe each of these datasets in greater detail.

Baseball Umpires. An important role of baseball umpires is to determine whether pitches should be called as *balls* or *strikes* when the batter does not swing. 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. While the definition of a strike is objective, 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 calls are predictable based on their past 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.[22] 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%.

Rock-Paper Scissors. In 2007, a Facebook app called Roshambull allowed Facebook users to play games of Rock-Paper-Scissors against one another. Each game consisted of two players and lasted until either player had won two matches.[23] 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 focus on the question of 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.[24] 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.

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

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

[24] The average game lasted 4.29 matches.

Within these new domains, where strings are generated by people making sequential decisions in real environments, we return to the question of the previous sections: how well can we predict individual entries in these human-generated strings, and how well can we distinguish them from Bernoulli strings with corresponding parameters?

4.2 Establishing a Benchmark

Following the approach outlined in Section 1, we use table lookup to construct a benchmark for the achievable level of prediction in these new field domains.

In the umpire setting, the objects of prediction are binary strings of length 6. 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 Bernoulli strings, with the probability of '1' set to equal the average flip in the umpire data (0.27). As before, the naive prediction error in the Classification problem is found by predicting 0.50 unconditionally. Because of the asymmetry in '1's and '0's, the naive prediction rule in the Continuation problem is different: we learn the average final flip in the training data, and predict this average unconditionally in the test data. These naive rules yield prediction errors of 0.2213 in the Continuation task and 0.25 in the Classification task.[25]

Using Rabin and Vayanos (2010) to predict in this setting, we find cross-validated prediction errors of 0.2204 in the Continuation task and 0.2489 in the Classification task. Thus, as in our main setting, the behavioral models are predictive.[26] We ask next how complete they are, and answer this again by using table lookup to establish a benchmark for the best achievable predictive accuracy in this domain. We find that the table lookup prediction errors are 0.2171 in the Continuation task and 0.2433 in the Classification task. Comparing the improvement upon the naive baseline achieved by the Rabin and Vayanos (2010) prediction rule, and by the table lookup rule, we find that Rabin and Vayanos (2010) achieves 17-21% of the attainable improvement in this problem. Robustness checks are reported in Appendix E, in which we vary the length of the strings, predicting instead strings of length 5 and length 4.

[25] The naive prediction error in Continuation is less than 0.25 because of the asymmetry of '1's and '0's.

[26] In the Classification task, the improvement in prediction error is statistically significant, although it is not for Continuation.

Table 8: Predicting umpire calls: Rabin and Vayanos (2010) explains 17-21% of the explainable variation in the data.

	Continuation		Classification	
	Error	Completeness	Error	Completeness
Naive	0.2213[a]	0	0.25	0
Rabin and Vayanos (2010)	0.2204	0.21	0.2489	0.17
	(0.0044)		(0.0005)	
Table Lookup	0.2171	1	0.2433	1
	(0.0039)		(0.0024)	

[a]The standard error of the above estimate is 0.0030.

Turning now to the Rock-Paper-Scissors data, the objects of prediction become *ternary* strings of length 6. The Continuation task in this domain is to predict the final throw given the first five, and 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 $\{r, p, s\}$. In the former problem, a prediction rule is a map from strings in $\{r, p, s\}^5$ to probability vectors in $[0, 1]^3$, where each coordinate corresponds (in order) 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 observation i (so that for example, throw of r is represented by $(1, 0, 0)$). 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. Naively guessing a probability of $1/3$ for each outcome yields a prediction error of 0.8165 in the Continuation task, and guessing $1/2$ unconditionally yields a prediction error of 0.25 in Classification.

Using the relaxed Rabin and Vayanos (2010) prediction model (which allows $\alpha, \delta < 0$), we obtain a cross-validated prediction error of 0.8160 in the Continuation task and 0.2491 in the Classification task. Thus, these models are again predictive. Turning to the question of how complete they are, we find that the table lookup errors are 0.8129 in the Continuation task and 0.2417 in the Classification task. Comparing the improvement upon the naive baseline achieved by the Rabin and Vayanos (2010) prediction rule, and by table lookup, we find that Rabin and Vayanos (2010) achieves

6-11% of the attainable improvement in this problem. Again, see Appendix E for robustness checks in which we predict strings of length 4 and 5.

Table 9: Predicting Rock-Paper-Scissors throws: Rabin and Vayanos (2010) explains 6-11% of the explainable variation in the data.

	Continuation		Classification	
	Error	Completeness	Error	Completeness
Naive	0.8165	0	0.25	0
Rabin and Vayanos (2010)	0.8160	0.06	0.2491	0.11
	(0.0005)		(0.0003)	
Table Lookup	0.8129	1	0.2417	1
	(0.0019)		(0.0013)	

Notice from Tables 8 and 9 that the measures of completeness in these field settings (which vary from 6 to 21%) do not differ substantially from the measure of completeness elicited using the experimental data in Section 1.4 (approximately 15%). This suggests that the extent to which the Rabin and Vayanos (2010) model captures predictable structure is stable across the different domains we have considered. Notice that this stability holds despite substantial differences in the nature of these domains: the experimental setting can be understood as one of pure generation, whereas the baseball umpire setting involves inference about an underlying state, and the Rock-Paper-Scissors setting involves beliefs about the random sequences generated by other players.

Although the completeness measure is stable within a range, the variations across the settings are revealing. Compare, for example, the table lookup prediction errors attained in the Classification problem in each of the domains.[27] This error is 0.2433 in the baseball umpire setting, 0.2422 in the experimental setting, and 0.2419 in the Rock-Paper-Scissors setting. Recalling that the table lookup prediction error represents the lowest achievable prediction error, this comparison suggests that there is less predictable structure in the sequences of baseball umpire calls, relative to the other two domains.[28]

[27] Comparisons of absolute levels in the Continuation task are harder to interpret, since the average flip varies across each of the domains.

[28] An interesting feature of the baseball domain — and a contrast with our other domains — is that

Additionally, the contrast between Tables 8 and 9 shows that the model based on Rabin and Vayanos (2010) is a more complete predictor of umpire calls than of Rock-Paper-Scissors throws. Although this difference is intuitive—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—illustration of this difference requires a notion such as our proposed measure of completeness. 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. A straightforward comparison of prediction errors, therefore, does not 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.

5 Discussion: Feature Sets

So far, we have emphasized the usefulness of table lookup as a tool towards the goal of evaluating the completeness of a model. We can alternatively focus on the benchmark itself as a way of quantifying the predictive power of a feature set. Recall that table lookup quantifies the best achievable accuracy *for a fixed feature set*; thus, the gap between this benchmark and perfect prediction reflects the limitations of that feature set. As we vary the feature set, the "best possible predictive accuracies" changes also.

Formally, consider a comparison between the lookup table we currently have, with columns corresponding to features from set X, and a larger lookup table in which we

there is a "true" answer for each pitch (ball or strike) that the umpire is trying to judge as accurately as possible. As a result, the sequence of umpire calls depends both on this sequence of true answers, and on the sequence of umpire judgments. Since the data of Chen et al. (2016) contained the true answer for each pitch, we investigated the extent to which table lookup could be used to predict this true answer from the sequence of prior true answers (independently of the umpire's call), and we found that for this Continuation task on true answers there is very little predictable structure in the sequence. Thus, the prediction performance on the sequence of umpire calls obtained by table lookup in this domain (at least for the Continuation task) appears to be coming primarily from the stucture in the sequence of calls themselves, rather than from the sequence of true answers. This is consistent with analysis of Chen et al. (2016), who argued that the auto-correlation in the sequence of calls arises principally from the umpire's decisions.

add a new set of currently unmeasured features from set Z, with a new column for each of these new features. Write $X \cup Z$ for the combined set of features, with a feature vector over this set of features denoted by $(\mathbf{x}, \mathbf{z})$. Focusing on prediction of a binary outcome, write $p_{X \cup Z}(\mathbf{x}, \mathbf{z})$ for the probability that an instance with features $(\mathbf{x}, \mathbf{z})$ will result in an outcome of '1'. Using the larger lookup table, we would approximate the prediction of $p_{X \cup Z}(\mathbf{x}, \mathbf{z})$ for each observed instance with features $(\mathbf{x}, \mathbf{z})$. If instead we observe only features from X, then the smaller lookup table would predict

$$p_X(\mathbf{x}) = \sum_{\mathbf{z}} p_{X \cup Z}(\mathbf{x}, \mathbf{z}) \Pr(\mathbf{z}|\mathbf{x})$$

for each instance with features $\mathbf{x}$, where $\Pr(\mathbf{z}|\mathbf{x})$ is the conditional probability that the unmeasured features take values $\mathbf{z}$ when the observed features take values $\mathbf{x}$. With sufficient data, the larger table provides (weakly) better predictive accuracy than the original smaller table, and the limitations in using the smaller table for prediction can be thought of as the consequence of implicitly averaging over all the possible values for the unmeasured features.

The extent of this limitation is determined by the power of the unmeasured features Z, and how they interact with the measured features X. Below, we provide one illustration of this comparison on the Continuation task, by predicting flip 8 using the k flips immediately prior to flip 8, where k is taken to vary from 1 to 7. Since each feature set with k prior flips nests any feature set with $k' < k$ prior flips, we can understand the difference in their table lookup errors as the value of uncovering $k - k'$ flips beyond the k' immediately previous.[29] For example, uncovering *two more* flips beyond flip seven, we can decrease prediction error from 0.2494 to 0.2482.[30]

The above comparison of different subsets of initial flips illustrates how table lookup can be used to compare the power of different feature sets. This comparison can be conducted for other sets of features as well. For example, in the Continuation task, we can ask how the best achievable prediction accuracy changes if we add subject ids as a feature (so that each row corresponds to a subject id and sequence

[29] This differs from the exercise in Appendix A.2, where we vary instead the flip to be predicted.

[30] This comparison need not be restricted to nested subsets. Figure 5 in Section C of the online appendix repeats this exercise for all possible features sets constructed from the previous seven flips; that is, all subsets of the first seven flips.

Table 10: Variation in the table lookup Continuation error for different choices of (nested) feature sets: the k flips immediately prior to flip 8.

Feature Set	Prediction Error
s_7	0.2494
	(0.0007)
s_6, s_7	0.2493
	(0.0008)
s_5, s_6, s_7	0.2482
	(0.0013)
s_4, s_5, s_6, s_7	0.2478
	(0.0013)
s_3, s_4, s_5, s_6, s_7	0.2470
	(0.0009)
$s_2, s_3, s_4, s_5, s_6, s_7$	0.2455
	(0.0018)
$s_1, s_2, s_3, s_4, s_5, s_6, s_7$	0.2439
	(0.0019)

of seven flips), or descriptions of the age and education level of the subject.[31] 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.

Our main exercise of finding the best possible prediction error for a fixed feature set, and this second exercise of asking how the best possible prediction error varies across feature sets, thus represent two conceptually distinct ways to improve prediction: measuring completeness helps us to understand how far prediction error can be reduced *without* the discovery or addition of new features.

[31] It is also interesting to note that certain feature sets, even if they were not of substantive interest for the domain, might reduce the error of the best predictive model by a very large amount: e.g. extremely detailed timing data on the keystrokes of the subject as they perform the task.

6 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. This is difficult to assess in general, but we introduce a social science domain—human perception and generation of randomness—in which it is possible to search the space of predictive models to optimality. This permits construction of a benchmark for the best achievable level of prediction, which we use to evaluate the predictive performances of leading economic theories in the domain: We find that these theories explain roughly 13-15% of the explainable variation in experimental data, and show moreover that table lookup can be used to construct benchmarks for prediction of field data that correspond to natural instantiations of human generation of randomness.

Two additional future directions are of interest: first, are there other social science problems in which brute force techniques such as table lookup might apply? Second, when brute force techniques are not be feasible, can we still use approaches from machine learning to construct a benchmark for the attainable level of prediction, and how do such approaches perform in practice? Our analyses here, together with results in the contemporaneous work of Peysakhovich and Naecker (2017), suggest that "approximate" benchmarks based on scalable machine learing algorithms may be effective practical solutions.

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*.

Hastie, T., R. Tibshirani, and J. Friedman (2009): *The Elements of Statistical Learning*, Springer.

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*.

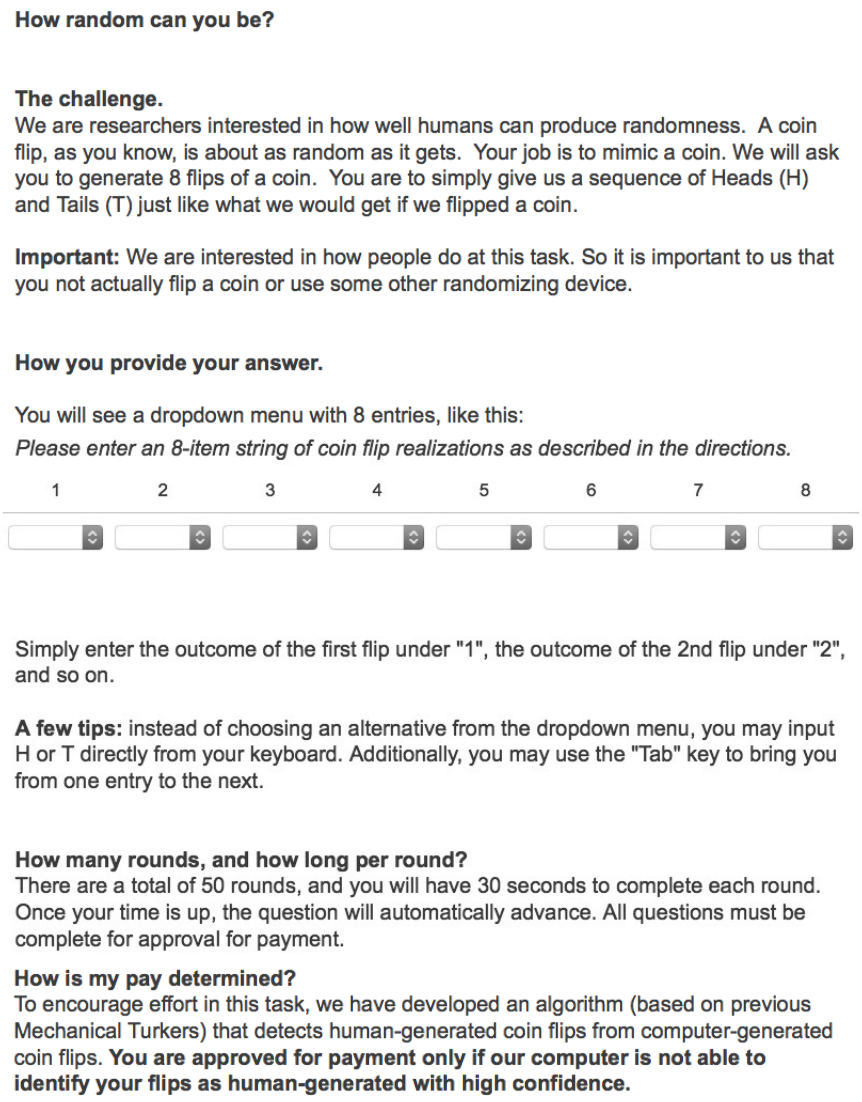
Rapaport, A. and D. Budescu (1997): "Randomization in Individual Choice Behavior," *Psychological Review*.

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

Appendix I

A Supplementary Materials to Section 1.1

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

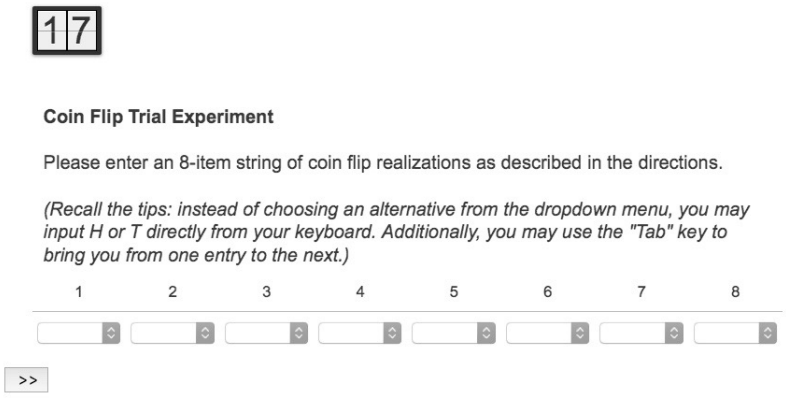
Simply enter the outcome of the first flip under "1", the outcome of the 2nd flip under "2", and so on.

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.**

Following a trial round and statement of consent, subjects were presented with 50 identical screens that looked like the following:



Subjects were given 30 seconds to complete each string, and a timer displayed their remaining time. Other versions of the experiment used similar instructions with small variations.

B Supplementary Materials to Section 1.2

The prediction rules based on Rabin (2002) include free parameters $p \in [0,1]$ and $N \in \mathbb{N}$. For Continuation, the probability that string $s \in \{1,0\}^7$ is followed by '1' is predicted to be

$$f_R(s) = 0.5p + \sum_{k=0}^{6} p(1-p)^k \left(0.5N - \textstyle\sum_{j=7-k}^{7} s_k\right)/N.$$

where s_k is the k-th flip in string s. For Classification, first define

$$q(s|r) = 0.5 \cdot r_k + \frac{1-r_k}{N}\left(0.5N - \textstyle\sum_{j=1}^{\min\ \{j\,:\,r_{k-j}=1\}} \mathbb{1}(s_{k-j} = s_k)\right)$$

to be the probability of generation of string s, when the urn is refreshed at every '1' in $r \in \{1,0\}^8$. Averaging over the distribution over refresh patterns, the probability of generation of $s \in \{1,0\}^8$ is

$$g_R(s) = \sum_{r \in \{0,1\}^8} \left(p^{\|r\|_1}(1-p)^{8-\|r\|_1}\right) q(s|r).$$

Finally, the probability that string s was generated by a human is predicted to be

$$c_R(s) = \frac{g_R(s)}{g_R(s) + 1/256}.$$

The prediction rules based on Rabin and Vayanos (2010) have free parameters $\delta, \alpha \in \mathbb{R}_+$. For Continuation, the probability that string $s \in \{1,0\}^7$ is followed by '1' is predicted to be

$$f_{RV}(s) = 0.5 - \alpha \sum_{t=0}^{6} \delta^t (2s_{7-t} - 1).$$

For Classification, define

$$g_{RV}(s) = 0.5 \cdot \prod_{k=2}^{8} \left(0.5 - \alpha \sum_{t=0}^{k-2} \delta^t \mathbb{1}\left(s_{k-t-1} = s_k\right) \right).$$

The probability that string $s \in \{1,0\}^8$ was generated by a human is predicted to be

$$c_{RV}(s) = \frac{g_{RV}(s)}{g_{RV}(s) + 1/256}.$$

C Supplementary Material to Section 1.4

C.1 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

D Supplementary Material to Section 3.2

D.1 Summary Statistics

Below, we briefly compare summary statistics of these new datasets with the original coin flip data. The basic distributional facts are similar: in the original coin flip data, Heads was produced in 52.61% of flips; under the new framings, the symbol 'r' is produced in 50.91% of flips in the $\{r, 2\}^8$ data, and Heads is produced in between 50.53%-51.73% of flips in the different cuts of the $\{H, T\}^{15}$ data. The features of misperception of randomness discussed previously in Section 1.1 appear also in the new data. As we show in Figure 2, subjects under-generate long runs and over-generate balanced strings in all of these settings.[32]

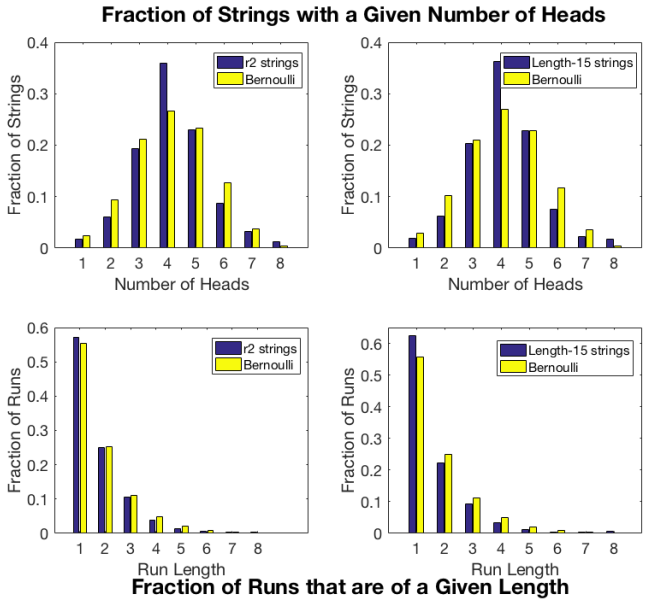


Figure 2: (a) Top row: the fraction of strings that include at least one run of length m, where m is the label on the x-axis. *Left*—comparison of $\{r, 2\}^8$ data (purple) with simulated Bernoulli strings (yellow); *Right*—comparison of the $\{H, T\}^{15}$ data (purple) with simulated Bernoulli strings (yellow). (b) Bottom row: the fraction of generated strings that include m Heads, where m is the label on the x-axis. *Left*—comparison of $\{r, 2\}^8$ data (purple) with simulated Bernoulli strings (yellow); *Right*—comparison of the $\{H, T\}^{15}$ data (purple) with simulated Bernoulli strings (yellow).

[32]The fractions of strings shown for the $\{H, T\}^{15}$ data are averaged across the seven datasets of flips k through $k + 7$. The comparison Bernoulli distributions are found by simulating a dataset for each of $k = 2, \ldots, 8$, where the probability of Heads is set to the mean flip in the dataset of strings k through $k + 7$.

D.2 Alternative Mappings

Table 11: Map 'H' in the original coin flip data to '2' in $\{r, 2\}$ and 'T' in $\{H, T\}$. How do the Continuation prediction errors change from Table 6?

	$\{r,2\}^8$		$\{H,T\}^{15}$	
	Error	Completeness	Error	Completeness
Naive	0.25	0	0.25	0
Rabin and Vayanos (2010)	0.2496 (0.0004)	0.06	0.2480 (0.0006)	0.15
Table Lookup	0.2437 (0.0011)	1	0.2370 (0.0024)	1

Table 12: Map 'H' in the original coin flip data to '2' in $\{r, 2\}$ and 'T' in $\{H, T\}$. How do the Classification prediction errors change from Table 7?

	$\{r,2\}^8$		$\{H,T\}^{15}$	
	Error	Completeness	Error	Completeness
Naive	0.25	0	0.25	0
Rabin and Vayanos (2010)	0.2494 (0.0007)	0.21	0.2488 (0.0006)	0.12
Table Lookup	0.2471 (0.0012)	1	0.2401 (0.0014)	1

Table 13: Map 'H' in the original coin flip data to 'T' in $\{H, T\}$ for the Nickerson and Butler (2009) data. How do the prediction errors change from Tables 6 and 7?

	Continuation		Classification	
	Error	Completeness	Error	Completeness
Naive	0.25	0	0.25	0
Rabin and Vayanos (2010)	0.2491 (0.0004)	0.05	0.2497 (0.0004)	0.03
Table Lookup	0.2336 (0.0017)	1	0.2413 (0.0015)	1

E Supplementary Material to Section 4.2

E.1 Baseball Umpires

In the main text, we predicted consecutive strings of umpire calls of length 6. Below we repeat the exercise, using consecutive strings of length 5 and length 4. There are 27,763 non-overlapping consecutive strings of length 5, and 56,312 non-overlapping consecutive strings of length 4.

Table 14: Predicting sequences of umpire calls of length 5: the model based on Rabin and Vayanos (2010) achieves 9-11% of the attainable reduction in prediction error

	Continuation		Classification	
	Error	Completeness	Error	Completeness
Naive	0.2071	0	0.25	0
	(0.0020)		(—)	0
Rabin and Vayanos (2010)	0.2066	0.11	0.2494	0.09
	(0.0037)		(0.0003)	
Table Lookup	0.2027	1	0.2436	1
	(0.0027)		(0.0011)	

Table 15: Predicting sequences of umpire calls of length 4: the model based on Rabin and Vayanos (2010) achieves 34-46% of the attainable reduction in prediction error

	Continuation		Classification	
	Error	Completeness	Error	Completeness
Naive	0.2237	0	0.25	0
	(0.0027)		(—)	0
Rabin and Vayanos (2010)	0.2209	0.34	0.2483	0.46
	(0.0010)		(0.0004)	
Table Lookup	0.2154	1	0.2463	1
	(0.0029)		(0.0005)	

E.2 Rock-Paper-Scissors

In the main text, we predicted consecutive strings of rock-paper-scissors throws of length 6. Below we repeat this exercise, using consecutive strings of length 4 and 5. There are 61,335 non-overlapping consecutive strings of length 5, and 117,522 non-overlapping consecutive strings of length 4.

Table 16: Predicting Rock-Paper-Scissors throws of length 5: the model based on Rabin and Vayanos (2010) achieves 11-13% of the attainable reduction in prediction error

	Continuation		Classification	
	Error	Completeness	Error	Completeness
Naive	0.8165	0	0.25	0
Rabin and Vayanos (2010)	0.8160 (0.0006)	0.11	0.2492 (0.0002)	0.13
Table Lookup	0.8120 (0.0006)	1	0.2438 (0.0008)	1

Table 17: Predicting Rock-Paper-Scissors throws of length 4: the model based on Rabin and Vayanos (2010) achieves 15-18% of the attainable reduction in prediction error

	Continuation		Classification	
	Error	Completeness	Error	Completeness
Naive	0.8165	0	0.25	0
Rabin and Vayanos (2010)	0.8159 (0.0002)	0.15	0.2494 (0.0002)	0.18
Table Lookup	0.8125 (0.0005)	1	0.2466 (0.0004)	1

Appendix II

For Online Publication

A Supplementary Material to Section 1

A.1 Context-Dependency

Subjects exhibit strong context-dependency: the probability of reversal depends not only on the immediately previous flip, but on the pattern of several prior. Table 18 lists the frequencies with which each possible three-flip pattern is followed by '1,' where the strings are sorted according to this quantity in decreasing order. These frequencies are compared with the data in Rapaport and Budescu (1997) (using Table 1 from Rabin and Vayanos (2010)).

Table 18: The empirical probability of Heads conditional on the three previous flips.

			Our data	Rapaport and Budescu (1997)	Bernoulli
0	1	0	0.5995	0.588	0.5
1	0	0	0.5406	0.62	0.5
0	0	1	0.5189	0.513	0.5
0	0	0	0.5185	0.70	0.5
1	1	1	0.4811	0.30	0.5
0	1	1	0.4595	0.38	0.5
1	1	0	0.4528	0.487	0.5
1	0	1	0.4415	0.412	0.5

The difference between the probabilities with which '000' and '111' are followed by '1' is significantly smaller in our data than in Rapaport and Budescu (1997); besides this, however, we find that these conditional probabilities are similar. Notably, the strings that are more likely to be followed by '1' in our data are also more likely to be followed by '1' in the Rapaport and Budescu (1997) data (compare the top four rows and bottom four rows in Table 18).

A.2 Predict Different Flips

Table 19: Continuation—Predict the k-th flip from the first $k-1$ realizations

	Continuation			Classification		
k	TL	RV2010	Completeness	TL	RV2010	Completeness
2	0.2499 (0.0004)	0.2500 (<0.0000)	0	0.2494 (0.0001)	0.2500 (<0.0000)	0
3	0.2489 (0.0005)	0.2490 (0.0004)	0.09	0.2489 (0.0001)	0.2500 (<0.0000)	0
4	0.2473 (0.0012)	0.2482 (0.0011)	0.67	0.2484 (0.0002)	0.2497 (0.0003)	0.19
5	0.2433 (0.0031)	0.2475 (0.0008)	0.37	0.2467 (0.0002)	0.2491 (0.0002)	0.27
6	0.2461 (0.0024)	0.2495 (0.0004)	0.13	0.2463 (0.0004)	0.2492 (0.0004)	0.22
7	0.2424 (0.0034)	0.2488 (0.0006)	0.16	0.2436 (0.0007)	0.2489 (0.0004)	0.18
8	0.2439 (0.0019)	0.2492 (0.0007)	0.13	0.2422 (0.0010)	0.2488 (0.0006)	0.15

A.3 Table Lookup Error

How close is the table lookup prediction error, estimated from n samples, to the best possible prediction error?

Continuation. For each string $s \in \{1,0\}^7$, let $p_s \in [0,1]$ be the true probability with which string s is followed by '1.' Table lookup estimates this probability using the sample mean $\overline{p}_s^n$ (the frequency with which s is followed by '1' in the training data). Notice that $\overline{p}_s^n \sim \text{Ber}(n_s, p_s)$, where n_s is the number of times string s is observed in the training data. The expected prediction error for string s can be shown to be

$$\underbrace{\mathbb{E}_X[(X-p_s)^2]}_{\text{irreducible error}} + \mathbb{E}_{\mathcal{D}}[(p_s - \overline{p}_s^n)^2]$$

where $\mathcal{D}$ is the (random) training data, and $X \sim \text{Ber}(p_s)$ is the eighth flip. This expression is the sum of the irreducible error (which we want to estimate) and the variance of the table lookup estimator. Writing q_s as the frequency with which string s is generated as the initial

substring, the expected table lookup prediction error is

$$\sum_{s\in\{1,0\}^7} q_s \cdot \left(\mathbb{E}_X[(X-p_s)^2] + \mathbb{E}_{\mathcal{D}}[(p_s-\overline{p}_s^n)^2]\right),$$

and the approximation error is

$$\sum_{s\in\{1,0\}^7} q_s \cdot \left(\mathbb{E}_{\mathcal{D}}[(p_s-\overline{p}_s^n)^2]\right) = \sum_{s\in\{1,0\}^7} q_s \cdot \frac{\overline{p}_s(1-\overline{p}_s)}{n_s}.$$

Substituting the empirical frequency $\hat{q}_s$ for the true frequency q_s in the expression above, the approximation error for our main dataset is estimated to be 0.0014. Moreover, we can upper bound this quantity, using that $\sum_s q_s \cdot \frac{\overline{p}_s(1-\overline{p}_s)}{n_s} \leq \min_s \left(\frac{\overline{p}_s(1-\overline{p}_s)}{n_s}\right) = 0.0033$.

Classification. For each string $s \in \{1,0\}^8$, let $p_s \in [0,1]$ be the true probability that string s was generated by a human source, and let $\overline{p}_s^n$ be the proportion of instances of s in the data that were generated by a human source. Using arguments similar to those above, the approximation error is

$$\sum_{s\in\{1,0\}^8} q_s \cdot \left(\mathbb{E}_{\mathcal{D}}[(p_s-\overline{p}_s^n)^2]\right) = \sum_{s\in\{1,0\}^8} q_s \cdot \frac{\overline{p}_s(1-\overline{p}_s)}{n_s}.$$

noting that $\mathbb{E}(q_s) = n \cdot p_s$. Again substituting the empirical frequency $\hat{q}_s$ for the true frequency q_s in the expression above, the approximation error for our main dataset is estimated to be 0.0031. An upper bound can be found using $\sum_s q_s \cdot \frac{\overline{p}_s(1-\overline{p}_s)}{n_s} \leq \min_s \left(\frac{\overline{p}_s(1-\overline{p}_s)}{n_s}\right) = 0.0066$.

B Supplementary Material to Section 2.2

Figure 3: *Left*: Continuation—If flip 7 is Heads, predict that the next flip is Heads with probability 0.48. Otherwise, if flips 1-4 are all Heads, predict 0.37, and if not, predict 0.54; *Right:* Classification—If the first flip is Heads, predict that the string was generated with a human with probability 0.53. Otherwise, if flips 2, 5, 6, and 7 are all Heads, predict 0.38, and if not, predict 0.48.

Two-Split Decision Trees Trained On Behavioral Features

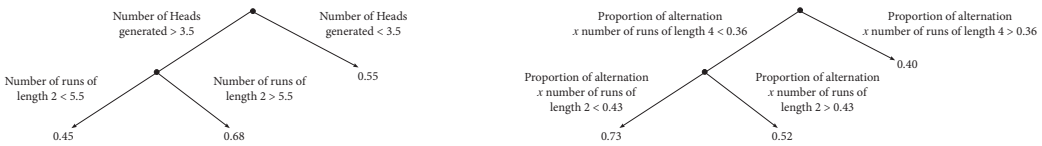


Figure 4: *Left:* Continuation—If the number of Heads generated in the first seven flips is less than 3.5, predict that the final flip is Heads with probability 0.55. Otherwise, look at the number of runs generated of length 2; if this is less than 5.5, predict 0.45; otherwise, predict 0.68; *Right:* Clasification —If the product of the proportion of alternation, and the number of runs of length four in the string, exceeds 0.36, then predict that the string is generated by a human subject with probability 0.40. Otherwise, predict 0.52 if the product of the proportion of alternation, and the number of runs of length two in the string, exceeds 0.43; if not, predict 0.73.

C Supplementary Material to Section 5

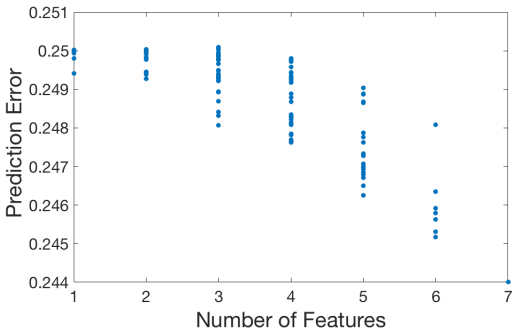


Figure 5: Table Lookup prediction errors for all possible feature sets constructed from the initial seven flips. Different observations above for fixed k correspond to different feature sets of size k.