Kendall, Chad; Oprea, Ryan

Article

On the complexity of forming mental models

Quantitative Economics

Provided in Cooperation with:
The Econometric Society

Suggested Citation: Kendall, Chad; Oprea, Ryan (2024) : On the complexity of forming mental models, Quantitative Economics, ISSN 1759-7331, The Econometric Society, New Haven, CT, Vol. 15, Iss. 1, pp. 175-211,
https://doi.org/10.3982/QE2264

This Version is available at:
https://hdl.handle.net/10419/296357

Supplement to "On the complexity of forming mental models"

(*Quantitative Economics*, Vol. 15, No. 1, January 2024, 175–211)

Chad Kendall
Department of Finance and Business Economics, Marshall School of Business, University of Southern California

Ryan Oprea
Economics Department, University of California

Appendix A: Additional data analysis

A.1 *Details on complexity notions and behavioral responses*

Figure A.1 provides a detailed view of the relationship between each of the complexity notions considered in the paper and extraction rates. It includes a separate panel for each complexity notion, plotting the rank value of the complexity metric on the x-axis and the mean extraction rate for each DGP on the y-axis. A dashed line plots a LOESS fit to the raw data. A corresponding figure for articulation rates is provided in Figure A.2, revealing similar patterns.[1]

Table 3 and Table 4 show the data in tabular form. First, in Table 3, we report metrics for our Implementation and Relational notions, which do not vary by input/output string across data sets. We also include mean extraction rates, articulation rates, and decision times. We provide these for the Unique treatment only, because there are multiple (sometimes a great many) rationalizing DGPs in the Multiple treatment. Second, in Table 4, we present string/sequence notions for each of the data sets in the Unique treatment (we do not use string/sequence notions in our analysis of the Multiple treatment). In this table, for each DGP, we assign a number to each of the four data sets used in the experiment (i.e., 1, 2, 3, or 4) and affix this identifier to the name of the corresponding DGP. Once again, for each data set we also include mean extraction rates, articulation rates, and decision times.

A.2 *Structural model*

In order to understand what types of models subjects tend to extract when multiple are available, we estimate a structural model of model-selection adapted from structural

Chad Kendall: chadkend@marshall.usc.edu
Ryan Oprea: roprea@gmail.com

[1] Because it is not clear how to cardinally interpret many of these complexity notions, we focus entirely on their ordinal explanatory value by normalizing all of our notions to their ordinal ranks. For instance, although our design varies the number of states between 2 and 3, we normalize them to vary between 1 and 2, by rank.

 https://doi.org/10.3982/QE2264

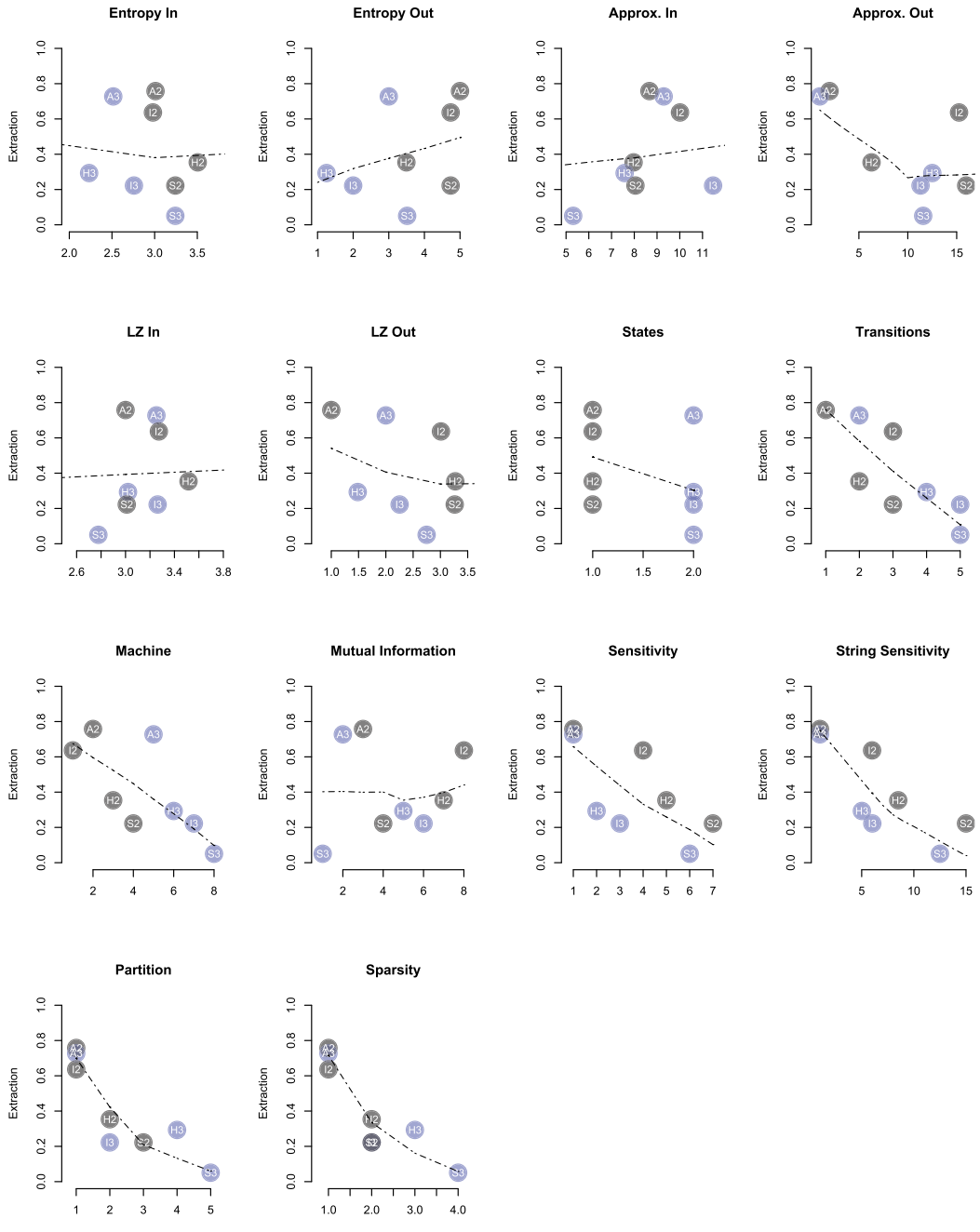


FIGURE A.1. Average extraction rate as a function of complexity metrics. *Notes*: Data is from 792 choices made in the Unique treatment. Each panel corresponds to one of our 14 complexity metrics. Each pictures the relationship between the rank ordering of the complexity metric (x-axis) and the mean rate of extraction (y-axis) across DGPs. Note that for the data set/string measures these averages aggregate over multiple data sets, producing fractional values for the rank ordering.

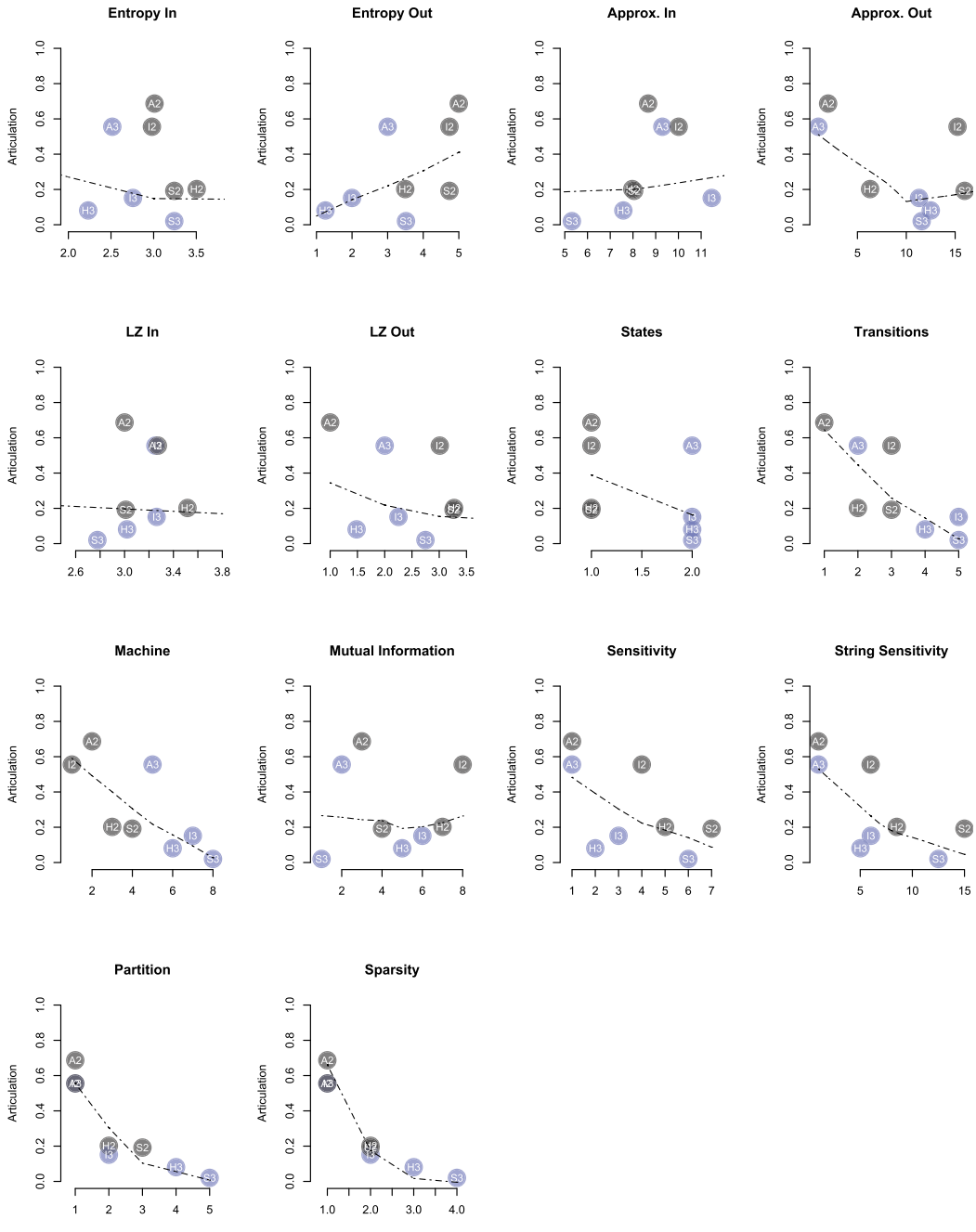


FIGURE A.2. Average articulation rate as a function of complexity metrics. *Notes*: Data is from 792 choices made in the Unique treatment. Each panel corresponds to one of our 14 complexity metrics. Each pictures the relationship between the rank ordering of the complexity metric (x-axis) and the mean rate of articulation (y-axis) across DGPs. Note that for the data set/string measures these averages aggregate over multiple data sets, producing fractional values for the rank ordering.

TABLE 3. Measures for implementation and relational complexity notions by DGP, unique treatment.

DGP	Extraction	Time	Articulation	States	Transitions	Machine	Mutual	Sensitivity	Partition	Sparsity
A2	0.758	39.097	0.687	2	2	3	0.385	0	2	1
H2	0.354	60.824	0.202	2	3	4	0.637	10,908	3	2
I2	0.636	46.735	0.556	2	4	2	0.693	8190	2	1
S2	0.222	72.502	0.192	2	4	7	0.387	90,114	4	2
A3	0.727	60.231	0.556	3	3	9	0.371	0	2	1
H3	0.293	67.498	0.081	3	5	12	0.414	7344	5	4
I3	0.222	62.844	0.152	3	6	16	0.569	8188	3	2
S3	0.051	102.901	0.02	3	6	19	0.364	60,060	1000	1000

Note: Extraction, Time, and Articulation variables show averages from the data. The remaining column gives corresponding complexity metrics for Implementation and Relational notions.

models of strategy choice from the experimental repeated games literature, specifically the Strategy Frequency Estimation Method (SFEM) developed by Dal Bó and Fréchette (2011). For each task, we estimate the proportion of subjects extracting each model. As in SFEM, we allow for random deviations between the actual output of the model and a subject's guess: $y_{ist}(m^k) = I\{m_{ist}(m^k) + \gamma\varepsilon_{ist} \geq 0\}$, where $I()$ is the indicator function. y_{ist} indicates the observed guess by subject i, on task s, in period t, where guesses of x are coded as 0, and guesses of y are coded as 1. $m_{ist}(m^k)$ is the actual output for model, m^k, coded as -1 for x and 1 for y. ε_{ist} is an error term that is assumed independent across subjects, tasks, and periods, and γ_{is} parameterizes the probability of a mistake. We assume that the error term has a logistic functional form such that it results in the standard logistic form for the likelihood that subject i matches model m^k on task s:

$$p_{is}(m^k) = \prod_T \left(\frac{1}{1 + e^{-\frac{m_{ist}(m^k)}{\gamma_{is}}}} \right)^{y_{ist}} \left(\frac{1}{1 + e^{\frac{m_{ist}(m^k)}{\gamma_{is}}}} \right)^{1-y_{ist}},$$

where T is the number of periods (guesses). Allowing each subject to match a different model on each task, the overall log-likelihood we estimate is given by

$$\mathcal{L} = \sum_S \sum_I \ln\left(\sum_{K_s} \tau_s(m^k) p_{is}(m^k) \right), \tag{1}$$

where S is the number of tasks, I is the number of subjects, and K_s is the number of consistent machines on task s. $\tau(m^k)$ are the main parameters of interest, representing the fraction of subjects that best match each model, m^k, in task s. We estimate the mixture model, (1), with the Expectation-Maximization (EM) algorithm, which provides better convergence properties than maximum likelihood.[2] We allow the "noise" parameter, γ_{is}, to vary by task to reflect the fact that some tasks are more difficult than others. We ran the estimation both with a homogeneous (across subjects) γ_s for each task and

[2]We estimate with 50 different starting points and choose the maximum likelihood across runs, but each run gives very similar estimates.

TABLE 4. Measures for string/sequence complexity notions by the data set used in the unique treatment.

Data set	Extraction	Time	Articulation	Entropy_In	Entropy_Out	LZIn	LZOut	ApproxIn	ApproxOut	String_Sensit
A2-1	0.708	43.967	0.708	0.679	0.69	4	3	0.353	−0.008	0
A2-2	0.72	37.936	0.76	0.637	0.69	5	3	0.648	−0.008	0
A2-3	0.833	31.629	0.667	0.679	0.69	6	3	0.667	−0.008	0
A2-4	0.769	42.612	0.615	0.693	0.69	5	3	0.248	−0.008	0
H2-1	0.375	57.275	0.417	0.693	0.617	4	5	0.471	0.33	13
H2-2	0.48	68.948	0.16	0.679	0.666	6	6	0.471	0.371	17
H2-3	0.083	56.342	0.042	0.679	0.666	6	4	0.592	0.241	16
H2-4	0.462	60.427	0.192	0.693	0.617	6	6	0.471	0.33	14
I2-1	0.708	58.079	0.667	0.693	0.69	4	5	0.471	0.576	12
I2-2	0.68	38.444	0.6	0.679	0.69	6	6	0.592	0.519	12
I2-3	0.542	45.188	0.458	0.679	0.69	5	4	0.592	0.519	12
I2-4	0.615	45.665	0.5	0.637	0.666	6	5	0.583	0.519	12
S2-1	0.167	56.762	0.208	0.693	0.69	4	6	0.31	0.576	78
S2-2	0.4	85.068	0.32	0.679	0.69	6	5	0.667	0.519	78
S2-3	0.083	64.667	0.083	0.679	0.69	5	4	0.471	0.63	78
S2-4	0.231	82.181	0.154	0.679	0.666	5	6	0.471	0.519	78
A3-1	0.792	69.9	0.75	0.637	0.617	4	4	0.592	−0.016	0
A3-2	0.72	59.108	0.6	0.562	0.617	6	4	0.438	−0.016	0
A3-3	0.75	47.175	0.542	0.679	0.617	6	4	0.471	−0.016	0
A3-4	0.654	64.438	0.346	0.693	0.617	5	4	0.604	−0.016	0
H3-1	0.5	67.504	0	0.679	0.429	5	4	0.533	0.492	8
H3-2	0.16	69.052	0.2	0.562	0.54	5	3	0.438	0.381	16
H3-3	0.208	55.212	0.083	0.679	0.429	4	4	0.31	0.492	11
H3-4	0.308	77.338	0.038	0.637	0.429	6	3	0.592	0.492	10
I3-1	0.083	60.062	0.083	0.679	0.429	6	4	0.667	0.287	9
I3-2	0.32	54.54	0.12	0.679	0.54	5	3	0.471	0.313	12
I3-3	0.125	59.812	0.125	0.637	0.617	4	5	0.592	0.653	14
I3-4	0.346	76.196	0.269	0.679	0.54	6	5	0.592	0.591	13
S3-1	0.042	114.888	0.042	0.693	0.54	5	5	0.416	0.39	42
S3-2	0.04	102.872	0.04	0.679	0.666	5	4	0.474	0.428	60
S3-3	0.042	72.725	0	0.679	0.666	3	5	0.004	0.345	54
S3-4	0.077	119.719	0	0.679	0.666	6	5	0.471	0.519	52

Note: Extraction, Time, and Articulation variables show averages from the data. The remaining column gives corresponding complexity metrics for string/sequence notions.

with heterogeneous γ_{is}. The results are very similar across the two specifications but we report the results for the heterogeneous model because a likelihood ratio test rejects the homogeneous model in its favor.

A.3 *Logit analysis*

For robustness, we reconduct analysis from the main text that is supported by chi-squared tests using logit regressions. This includes results 1, 2, and 5. For both the Unique and Multiple treatment, we ran logit regressions that include (i) subject and period fixed effects and (ii) standard errors clustered at the subject level. For each treatment, we ran the following three specifications:

- **Null**: Includes only fixed effects.
- **DGP**: Includes dummy variables for DGPs/tasks.
- **DGP+Data set** includes dummy variables for DGPs/tasks and also data sets.

First, we produce alternative evidence for result 1 by comparing the Bayesian Information Criterion (BIC) between the Null and DGP specifications. The BIC is a standard metric for examining whether additional parameters improve the explanatory value of a model while penalizing for the number of parameters to avoid overfitting. The BIC for the DGP specification is lower than for the Null specification (1157 vs. 1434), indicating that taking account of variation in the DGP is important for statistically understanding extraction rates. Thus, this analysis supports our claim that extraction rates vary meaningfully across DGPs.

Second, we produce alternative evidence for result 2 by comparing the BIC between DGP and DGP+Data set. We find that the BIC for DGP+Data set is *not* lower than that for DGP (1212 vs. 1157), suggesting that variation across DGPs is more important than variation across data sets for accounting for variation in extraction rates.

Finally, in Figure A.3, we plot coefficient estimates of the task dummies for both Unique and Multiple from the DGP specification and include 95% confidence intervals. Unsurprisingly, the point estimates look similar to Figure 1. However, the standard error bars give us additional evidence in support of Results 1 and 5. First, nonoverlap in confidence intervals across many pairwise comparisons of DGPs in the Unique treatment,

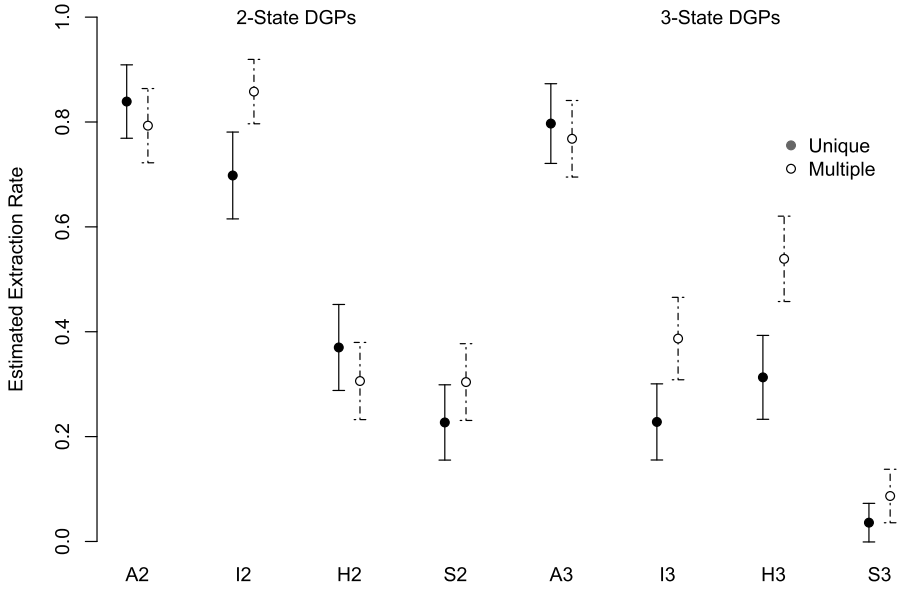


FIGURE A.3. Marginal effects from logit analysis of extraction rates. *Notes*: Results are from two logit regressions (one for Unique, another for Multiple) that include (i) dummy variables for each task and (ii) subject and period fixed effects. Standard errors are clustered at the subject level. Error bars indicate 95% confidence intervals.

shows that extraction rates statistically differ across DGPs. Second, comparing Unique and Multiple, tasks I2, I3, and H3 have nonoverlapping confidence intervals, suggesting that extraction rates differ across Unique and Multiple treatments. This finding exactly matches that using chi-squared tests reported in the main text in the discussion surrounding result 5.

A.4 *Additional figures*

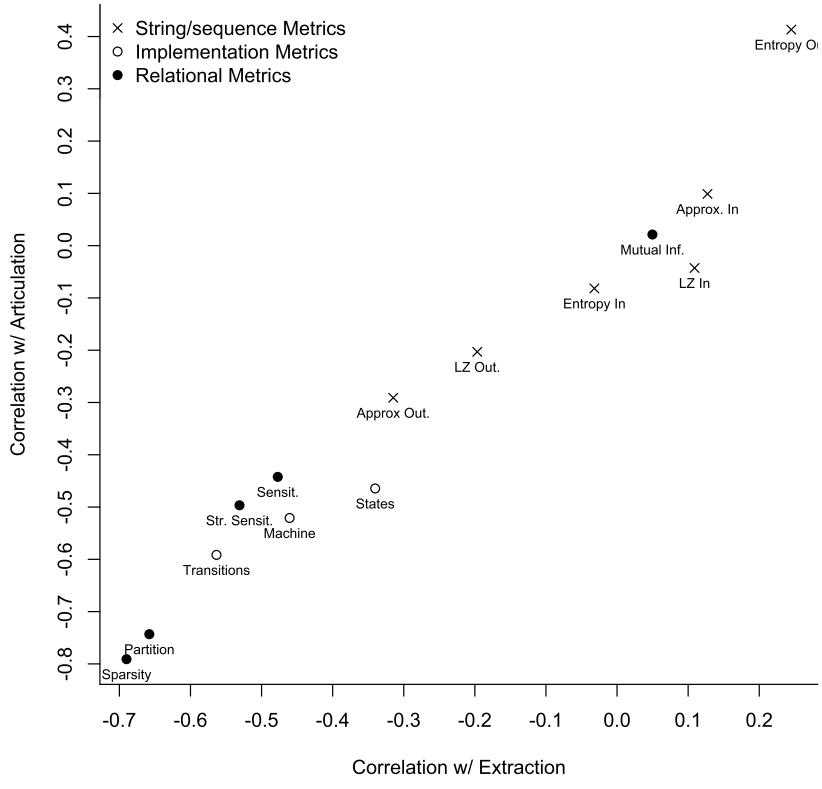


FIGURE A.4. Correlation between complexity metrics and articulation/extraction. *Notes*: This is a version of Figure 6 with raw correlations plotted rather than their absolute values.

APPENDIX B: DETAILS ON COMPLEXITY NOTIONS

B.1 *Partition complexity*

Lipman (1995) describes a general model of bounded rationality based on limited abilities to process information. A decision maker that is trying to learn a state of the world observes a series of inputs, but may not fully process the information contained in those inputs. In particular, Lipman (1995) describes a class of partitional models in which the decision maker partitions the state of the world into several elements, where a finer

partition corresponds to more fully processing information. Our partitions complexity measure builds upon this general idea. Intuitively, models which require more of the information in the data set to be processed (i.e., a finer partition) may be more difficult to infer.

In our setting, both the state of the world and the "input" that the decision-maker processes can be taken to be the observed data set: the ordered set of the inputs and outputs up to time T: $h^T \equiv \{v_0, u_1, v_1, u_2, \ldots, v_{T-1}, u_T\}$, where $v_t \in V$ denotes an output of the model and $u_t \in U$ an input.[3] The information contained in the data set is processed to determine the subsequent output, v_T. In what follows, we define a model-based measure of complexity, allowing $T \to \infty$.[4]

We assume that decision makers process the data set by looking for *patterns*: recurring combinations of inputs and past outputs that lead to the same output. Formally, we define a *subdata set*, $h^t \equiv \{v_0, u_1, v_1, u_2, \ldots, v_{t-1}, u_t\}$ with $t \in \{1, 2, \ldots, T\}$. Let a generic element of the subdata set be denoted $e_t \in \{u_t, v_t\}$. We denote a subset of h^t by S. A pattern, denoted $S \to v$ is then defined to be a mapping from some S to a constant v, such that $v_t = v$ for all of the subdata sets in the data set, except for perhaps the subdata sets given by $t = 1, \ldots, t'$ for some finite t'. Examples of simple patterns are $u_t = a \to x$, $v_{t-3} = x \to x$, and $u_t = a$, $v_{t-1} = x \to x$.

A set of patterns forms a *partition* of the set of subdata sets if each subdata set is associated with one and only one pattern in the set. We denote such a set of patterns, $R^* \in \mathbf{R}$ where $\mathbf{R}$ is the set of all possible sets of patterns. Our *partitions complexity* measure is then taken to be the minimum number of partition elements in any $R^* \in \mathbf{R}$: models, which require more patterns to describe the data set require more information processing, and are therefore more complex. For example, I2 can be described by two patterns, $u_t = a \to x$ and $u_t = b \to y$. A3 can also be described by two patterns, $v_{t-3} = x \to x$ and $v_{t-3} = y \to y$. S2 requires four patterns, each of which maps a value of v_{t-1} and a value of u_t to an output.

It is important to note that in constructing a partition of the set of subdata sets through patterns as defined, we are not allowing for any partition. For example, one partition that is ruled out is "all of the subdata sets for which $v_t = x$ and all of the subdata sets for which $v_t = y$." Although a possible partition, this partition is not particularly useful in thinking about the complexity of a model because it leads to a complexity measure of two for all models. Put another way, to construct this partition, a decision maker must infer the model, which if allowable, can tell us nothing about how difficult a model is to infer.

In limiting to patterns as defined, it may not always be possible to construct a partition with a finite number of patterns. It turns out though that for the eight models we study, finite partitions can be constructed for seven of the them.[5] Importantly, given that

[3] This construction is similar to the description in Lipman (1995) of the history of a repeated game as both the state of the world and the input. In his formulation, a strategy partitions the history and more complex strategies use finer partitions.

[4] A model-based measure is somewhat easier to work with than a measure based on a data set of finite length because it avoids having to deal with exceptions that may occur at the beginning of the data set.

[5] S3 is the exception, requiring a countably infinite number of simple patterns. It requires one to look for the pattern, "if the input is b and the output was x when the third most recent input of b occurred, the next

only one of models requires an infinite number of partitions, our partitions complexity measure can still be used to rank the eight machines. Thus, although we could allow for more complex patterns, simpler patterns are sufficient for our purposes.[6] In addition, allowing for more complex patterns would significantly increase the algorithmic complexity of finding the partition with the minimum number of elements. In fact, even with relatively simple patterns, it is a nontrivial problem.

B.2 *Sparsity complexity*

Gabaix (2014) provides a general framework for studying standard optimization problems in economics when decision makers must pay costs to attend to inputs to the optimization process. In his framework, the decision maker first decides which inputs to attend to by comparing the attention cost to an approximation of the value of the input. He shows that decision makers will optimally consider only a sparse set of inputs, ignoring those with relatively little value. A natural complexity measure inspired by this framework, then is the number of inputs a decision maker must attend to (i.e., must avoid ignoring) in order to behave optimally. In our setting, the complexity of a model is the number of input/output elements in the data set the DM must examine in order to correctly apply the model correctly. In our operationalization, each element of the data set is a potential input that might be processed, and our measure of complexity is the number of elements in the data set that must be attended to in order to correctly predict the output. For example, A2 requires the DM to attend to v_{t-2} only to guess the next output and is therefore one of the simplest possible models. S2 is more complex, requiring the DM to attend to both u_t and v_{t-1} in order to predict. As with partitions complexity, S3, is the only one of our DGPs out of the eight that we study for which there is no finite set of elements that will always predict the output, and we code it as maximally complex under this metric.

B.3 *Algorithms for determining complexity measures*

We discuss the algorithm used to find partition complexity. Sparsity complexity is determined at the same time because once each partition that can describe the model is identified, both the number of patterns in the partition and the number of data set elements used in constructing the patterns are known. Partition complexity is the minimum number of patterns in any partition and sparsity complexity is the minimum number of elements in any partition.

We take a brute force approach to finding the partition with the minimum number of elements. For a given number, τ, time periods, we form all the possible partitions with the subset of elements, $\{v_{T-\tau-1}, u_{T-\tau}, \ldots, v_{T-1}, u_T\}$. If a partition cannot be found,

output will be x," which when restricted to simple patterns, requires a partition with a countably infinite set of simple patterns due to the fact that an arbitrary number of a inputs can occur between b inputs.

[6] In the selection treatment analysis, several of the models consistent with the data sets also require a countably infinite number of simple patterns so that we cannot rank all consistent models. However, for each of the eight data sets, at least one model can be characterized by a finite number of patterns. As we are only interested in whether or not the simplest model is most often selected, this suffices.

we set partition complexity to infinity and otherwise set it to the number of patterns in the smallest partition. We cannot search over all partitions as $\tau \to \infty$ and, therefore, cannot guarantee we have found the partition of minimum size. However, $\tau = 3$ appears to be sufficient in the sense that the complexity measures do not change for a sample of larger values. Intuitively, the minimum number of patterns should use only elements from the most recent three periods for automata with at most three states, but given all the ways in which partitions can be formed, we have been unable to prove this formally. Proposition 1 does show though that if a partition cannot be formed using elements from the two most recent periods, then adding a finite number of prior elements does not help. So, we can be certain that the cases in which we set the partition complexity to infinity do indeed require an infinite number of patterns.

PROPOSITION 1. *If the subset of elements, $v_{t-2}, u_{t-1}, \ldots, u_t$, does not produce a partition with a finite number of elements, then incorporating an additional finite set of prior elements does not either.*

PROOF. For any 1- or 2-state automaton, v_{t-1} and u_t, always produce a partition with at most four elements because the output, v_{t-1}, directly maps to the current state of the automaton which, together with the input, determines the next state and output. For 3-state automata in which two states map to the same output, v_{t-1} and u_t are no longer sufficient. Without loss of generality, assume a single state corresponds to the output, y, and two states, x_1 and x_2, correspond to x. For v_t not to be determined by some subset of past elements, it must be the case that the state of the automaton is indeterminate after the subset because if the state is uniquely determined, then so is the output, v_t.

For the subset of v_{t-2}, u_{t-1}, v_{t-1}, and u_t, not to determine the automaton state, it must be the case that either x_1 and x_2 both transition back to themselves on the same input, or that the each transitions to the other on the same input. To see this, suppose to the contrary that neither of these conditions holds. Then for each input, u, either one of the x states transitions to the other and the other transitions back to itself, or one or both of the x states transitions to y. In the first case, v_{t-1} and $u_t = u$ uniquely determines the state, and in the second case, the subset of v_{t-2}, u_{t-1}, v_{t-1}, and u_t uniquely determines the state.

Finally, if x_1 and x_2 transition back to themselves on the same input, u, then no finite number of past outputs and inputs necessarily determines the state because u could repeat indefinitely. Similarly, if each of x_1 and x_2 transitions to the other on the same input, u, then no finite number of past outputs and inputs necessarily determine the state. Therefore, no partition with a finite number of elements exists. □

APPENDIX C: DESIGN DETAILS

C.1 *Data sets used in the design*

In this section, we provide all of the data sets used in the experiment, discuss how they were selected, and use examples to illustrate how data sets in Unique could be used to uniquely determine the automaton while those in Multiple could not.

TABLE 5. Data sets used in the unique treatment.

	Input String	Output String
A2-1	000011011010111001011110	010101010101010101010101
A2-2	000111001000001100101011	010101010101010101010101
A2-3	010111100011110011100110	010101010101010101010101
A2-4	110011100100000100111011	010101010101010101010101
I2-1	111101001000000011101101	011110100100000011101101
I2-2	011010011101001101010011	001101001110100110101011
I2-3	010110010111010111000100	001011001011101011100100
I2-4	010011011111000110001101	001001101111100011000 1101
H2-1	111100010100001000100110	010100001010000100010 0100
H2-2	100111100101111100000101	010010100010101010000 0101
H2-3	010110111001001111001001	001010010100100101000 1001
H2-4	100111011000100000100100	010010101000010000010 0100
S2-1	111010100100010100010011	010110011100000110000 11101
S2-2	011010001111100111001111	001001111010100010111 0101
S2-3	000110101100001011010010	000010011011111100100 11100
S2-4	010101110000110000110001	001100101111110111111 011110
A3-1	111101010011101100110111	010010010010010010010 0100
A3-2	010001100000101000000110	010010010010010010010 0100
A3-3	100101111001010110010100	010010010010010010010 0100
A3-4	000111101001010011111000	010010010010010010010 0100
I3-1	100111010000111001101100	000001100000001100010 0100
I3-2	100100110111010110011001	000000001001100001000 1000
I3-3	111101001101011101111101	001110000010000110011 1100
I3-4	101110011010111111101110	000011000100001111110 0110
H3-1	110111100001010010101001	001001000000000000000 0000
H3-2	100111110111100110110001	000001001001000001001 0000
H3-3	111110100100000010011110	001001000000000000000 1000
H3-4	101001101111101010111000	000000010010010000001 0000
S3-1	101010111000110001010010	011000010000010000001 1100
S3-2	001111010110011110011111	000100110001110010000 1001
S3-3	101010101011110101111110	011000011000100011001 0011
S3-4	100111100101011001000100	011100100000110000111 1000

Note: For inputs, 0 was represented as a and 1 as b in the experiment; for outputs, 0 was represented as x and 1 as y.

First, Table 5 lists the data sets used in the Unique treatment. We selected these data sets using an algorithm that selected input/output pairs that could only be rationalized by a single (unique) nontrivial finite automaton of less than four states. To the extent

TABLE 6. Data sets used in the multiple treatment.

	Input String	Output String
A2	100100101101111101011000	010101010101010101010101
I2	101011111110000011001100	010101111111000011001100
H2	001011101011110100010110	000101010101010010010100
S2	101000111110100010010111	011000010101100011100101
A3	010010011000100110011001	010010010010010010010100
I3	110110100010100110111110	001001000000000010011110
H3	100000110111100000101001	000000001001000000000000
S3	100100111100110000011000	011100001000010000001111

Note: For inputs, 0 was represented as a and 1 as b in the experiment; for outputs, 0 was represented as x and 1 as y.

possible, we also selected data sets that were ranked differently according to the various string measures of complexity.

For instance, consider S2-1 a data set produced by the S2 DGP. Subjects observe the first 12 inputs and first 13 outputs (including the initial 0) in part 1 before making guesses. In this case, each of the first three inputs of 1 cause the state to flip. This rules out the alternative DGP of I2, but up to this point the strings are also consistent with A2 or H2. The next zero input causes the output to stay at 1, which then rules out A2 and H2. In this way, the strings shown in part 1 similarly rule out all other nontrival finite automata of less than four states.

Second, Table 6 shows the data sets used in the Multiple treatment. Here, our algorithm deliberately selected data sets that are rationalizable by *more than one* (multiple) nontrivial finite automata of less than four states. Furthermore, we attempted to ensure the different data sets (i) were rationalizable by different numbers of automata in this class and (ii) provided variation in how each of the rationalizable machines was ranked according to the various model measures of complexity.

For instance, consider the data set generated with I2. The first twelve inputs and thirteen outputs shown to subjects in part 1 (prior to making guesses) are consistent with I2, but are also consistent with a version of H2 in which a zero always switches the output and a one always outputs a one (H2 instead switches when the input is one but outputs a zero when the input is zero). This particular string is also consistent with several 3-state machines.

APPENDIX D: INSTRUCTIONS TO SUBJECTS

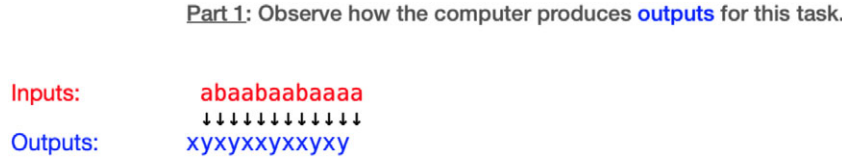
The same instructions were provided to subjects in both treatments. Each set of bullet points corresponds to a different page in the instructions and bullets were revealed to subjects one-by-one. Subjects were required to correctly answer comprehension questions, reproduced below, before proceeding to the experiment.

Instructions

- We will start by providing you with **INSTRUCTIONS** for the study.
- We will ask you **COMPREHENSION QUESTIONS** to check that you understand the instructions. You should be able to answer all of these questions correctly.
- Please read and follow the instructions closely and carefully.
- If you **COMPLETE** the main parts of the study, you will receive a **GUARANTEED PAYMENT** of **$2.50**.
- In addition, your **CHOICES** in the **DECISIONS** portion of the study will result in **PERFORMANCE-BASED EARNINGS**. You will experience several **TASKS** worth **REAL MONEY**. Your points from **ONE OF THE TASKS** will be randomly selected by computer and will be converted into an additional payment.

Inputs and Outputs

- The experiment will consist of several separate **TASKS**. Each task will consist of a series of **PERIODS**. The screen will look like this:

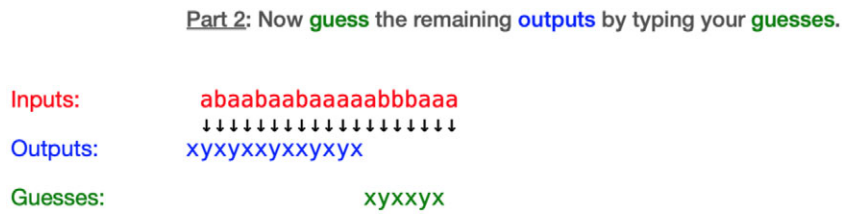


- In every period, we will show you an INPUT (a letter in red) followed by an OUTPUT (a letter in blue) as in the screen above.
- The periods are shown on your screen from left to right—each horizontal location is a different period. In order to help you remember that outputs come **AFTER** inputs in each period, we draw a little arrow from inputs to outputs in each period.
- The computer uses a **RULE** to decide on outputs each period. That rule may depend on the current and/or past inputs or it **may not depend** on the inputs at all. You will **not know**. Most importantly, the rule is **not random**.
- Although outputs **may** depend on inputs, inputs never depend on outputs (the inputs were selected before the experiment began).
- **Comprehension question: Which of the following is true?**
  1. Inputs come before outputs in each period.
  2. Outputs come before inputs each period.
  3. Inputs and outputs happen at the same time each period.
- **Comprehension question: The rule the computer uses to determine outputs:**
  1. is random.

2. is a rule that always depends on inputs.
3. is a rule that never depends on inputs.
4. is a rule that may or may not depend on inputs.

Making Guesses

- In part 1 of each task, we will show you a set of inputs and outputs (one at a time) for 12 periods to help you get a sense of how outputs are determined. Then, in part 2 of the task, we will have a series of 12 additional periods in which we will show you inputs and have you try to **GUESS** the correct output.
-



  The screen will look like the image above. In the first 12 periods, you make no guess but just observe the process, as it automatically happens. In the last 12 periods, you make a guess each period after each new input appears.

- To make your guess, just type (on your keyboard) the letter of the output that you think comes next (outputs can only be "x" or "y"). The more periods in which you correctly guess the outputs, the higher your **BONUS**.
- **IMPORTANT**. The **rule** the computer uses to determine outputs will stay the same throughout the task. But it **may change from task to task**. So, do not assume that the way outputs are determined is the same for every task!
- **IMPORTANT**. The **rule** the computer uses to determine outputs does **not** depend on your guesses. The set of inputs and outputs was determined ahead of time and **does not respond at all** to what you do in the experiment.
- After the last task, the computer will **randomly select one task** and pay you a bonus of $0.35 for each output you correctly guessed in that task.
- **Comprehension question: My guesses influence the inputs and outputs in the experiment**

  1. False
  2. True
  3. It depends

- **Comprehension question: The rule determining outputs**

1. changes throughout each task and across tasks.
2. is the same throughout each task, but changes from task to task.
3. is the same throughout each task and across tasks.

Giving Advice, Describing the Rule

- At the end of each task, when you are done making your guesses, we will have you **give advice** to another participant who will face a similar task in the future.
- The future participant **will not** experience part 1 of the task and will see a **different set of inputs** than you did in part 2.
- However, the **rule** the computer uses to determine inputs will be exactly the same as in your task. That means, you can only help the future participant by trying your best to **describe the rule** you think the computer used during the task.
- In part 3 of the task, we will therefore give you a box to type the rule you think the computer used to determine outputs.
- The better job you do in describing the rule, the more money you can earn. With 10% likelihood, we will show what you typed to a future participant. This will be a sophisticated participant—a graduate student in a quantitative field. **You will earn $0.35 for every letter that participant gets right.** That means, the better job you do of accurately describing the rule, the **more money** you will earn on average in the experiment.
- **Comprehension question: I earn money in part 3 of the task by describing to a future participant**
  1. what my guesses were.
  2. the rule the computer uses in the task.
  3. how people are paid in the experiment.
- **Comprehension question: If I do a good job of describing the rule,**
  1. it will have no impact on anything so I shouldn't bother.
  2. it may help another participant earn more money.
  3. it may may help another participant earn more money, AND the more they earn the greater my bonus will be.

References

Dal Bó, Pedro and Guillaume R. Fréchette (2011), "The evolution of cooperation in infinitely repeated games: Experimental evidence." *American Economic Review*, 101 (1), 411–429. [4]

Gabaix, Xavier (2014), "A sparsity-based model of bounded rationality." *The Quarterly Journal of Economics*, 129 (4), 1661–1710. [9]

Lipman, Barton L. (1995), "Information processing and bounded rationality: A survey." *Canadian Journal of Economics*, 28 (1), 42–67. [7, 8]

Co-editor Garance Genicot handled this manuscript.

Manuscript received 5 October, 2022; final version accepted 28 September, 2023; available online 2 October, 2023.