Simplicity bias and model-free forecasting in the discretised logistic map

Kamaludin Dingle[1,2], Guillermo Valle Pérez[2,3], Ard A. Louis[2]
[1] *Centre for Applied Mathematics and Bioinformatics, Department of Mathematics and Natural Sciences, Gulf University for Science and Technology, Kuwait,*
[2] *Rudolf Peierls Centre for Theoretical Physics, University of Oxford, Parks Road, Oxford, OX1 3PU, United Kingdom*
[3] *FLOWERS Laboratory INRIA, Bordeaux, France*
(Dated: June 7, 2022)

The logistic map is an iconic example of complex and chaotic behaviour emerging from a simple set of rules. An open question for such systems is how well we can predict patterns in their behaviour. Here we apply recent results from algorithmic information theory (AIT) which predict simplicity bias, an exponential upper bound decay in the probability of an output appearing with linear increases in its Kolmogorov complexity. It is not theoretically clear whether or not the logistic map obeys the criteria for simplicity bias, so we approach this question empirically. We randomly sample parameters of the map, generate trajectories, and then discretize them to produce binary output strings. The complexity of the strings is measured with a measure based on Lempel Ziv, which is an approximation to Kolmogorov complexity that has been shown to work well for other input-output maps. We find that the logistic map exhibits clear simplicity bias behaviour over the full range of parameters, but that the simplicity bias disappears in the chaotic regime. We show that, on average, we can use AIT to predict the probability ratios of different trajectories with high accuracy. Finally, we study sequence forecasting, making use of algorithmic probability estimates for trajectory extrapolation. Using techniques from AIT, many non-trivial predictions can be made which utilise almost no details of the underlying map, and in this sense constitute model-free forecasting. Furthermore, the general methods which we describe can be applied to other systems and time series, not just the logistic map.
Keywords: Logistic map; Kolmogorov complexity; algorithmic probability; universal prediction; chaos; entropy; randomness

I. INTRODUCTION

do intro to Logistic equation, starting with its popularization by Robert May – it has been very heavily studied since. Here we apply a new set of idea around simplicity bias, which may be more generally applicable, if it works

Note this model is related to a much wider set of non-linear models, including epidemiology, (Sunetra Gupta) etc... (make some list

One branch of theoretical computer science, *algorithmic information theory* [11–13] (AIT) directly connects computation, computability theory, and information theory. The central quantity of AIT is *Kolmogorov complexity*, $K(x)$, which measures the complexity of an individual object x as the amount of information required to describe or generate x. $K(x)$ is more technically defined as the length of a shortest program which runs on an optimal prefix *universal Turing machine* (UTM) [14], generates x, and halts. An increasing number of studies show that AIT and Kolmogorov complexity can be successfully applied in physics, including thermodynamics [15–17], quantum physics [18], and entropy estimation [19, 20]. Further, applications to biology [21–23], other natural sciences [24], and engineering, are also numerous [25–27].

An important result in AIT is Levin's [28] coding theorem, establishing a fundamental connection between $K(x)$ and probability predictions. Mathematically, it states that $P(x) \sim 2^{-K(x)}$ where $P(x)$ is the probability that an output x is generated by a (prefix optimal) UTM fed with a random binary program. Given the broad reaching and striking nature of this theorem, it is somewhat surprising that it is not more widely studied in the natural sciences. The reason in part for this inattention is that AIT results are often difficult to apply directly in real-world contexts, due to a number of issues including the fact that $K(x)$ is formally uncomputable and the ubiquitous use of UTMs (see the Discussion for more on this topic).

Conscious of these points, we recently studied coding theorem-like behaviour in real-world input-output maps, leading to our observation of a phenomenon called *simplicity bias* (SB) [29] (see also ref. [30]). SB is captured mathematically as

$$P(x) \leq 2^{-a\tilde{K}(x)-b} \tag{1}$$

where $P(x)$ is the (computable) probability of observing output x on random choice of inputs, and $\tilde{K}(x)$ is the approximate Kolmogorov complexity of the the output x: complex outputs from input-output maps have lower probabilities, and high probability outputs are simpler. The constants $a > 0$ and b can be fit with little sampling and often even predicted without recourse to sampling [29]. Examples of systems exhibiting SB are wide ranging, and include molecular shapes such as protein structures and RNA [22], finite state machines outputs [31], as well as models of financial market time series and ODE systems [29], among others. The ways in which SB differs from Levin's coding theorem include that it does not assume UTMs, uses approximations of complexities, and for many outputs $P(x) \ll 2^{-K(x)}$. Hence the abundance of low complexity, low probability outputs [31] is a signature of SB.

A full understanding of exactly which systems will, and will not, show SB is still lacking, but the phenomenon is

expected to appear in a wide class of input-output maps, under fairly general conditions. Some of these conditions were suggested in ref. [29], including (1) that the number of inputs should be much larger than the number of outputs, (2) the number of outputs should be large, and (3) that the map should be 'simple' (technically of $O(1)$ complexity) to prevent the map itself from dominating over inputs in defining output patterns. Indeed, if an arbitrarily complex map was permitted, outputs could have arbitrary complexities and probabilities, and thereby remove any connection between probability and complexity. Finally (4), because many AIT applications rely on approximations of Kolmogorov complexity via standard lossless compression algorithms [32, 33] (but see [34, 35] for a fundamentally different approach), another condition proposed is that the map should not generate pseudo-random outputs like $\pi = 3.1415...$ which standard compressors cannot handle effectively. The presence of such outputs may yield high probability outputs which appear 'complex' hence apparently violating SB, but which are in fact simple.

Motivated by exploring further the presence of SB in physics, and also in maps which test the boundaries of the conditions for SB described above, in this work we examine the output trajectories of the famous logistic map from chaos theory. This map is attractive because its trajectory outputs can depict simple as well as complex, chaotic, and even pseudo-random patterns. Although not restricted to studying binary strings, most AIT results are framed in the context of binary strings and hence applications are easier in the same framework. Thus in this work we will study discretised binary trajectories of the logistic map. Additionally, the trajectories of the logistic map can be viewed as time series, and we utilise this aspect also, to investigate another AIT-inspired phenomenon, which is forecasting using essentially only the complexity of trajectories, hence a kind of model-free forecasting. We derive a conditional version of Eq. (1) to make algorithmic probability-based predictions of these time series. The results described here can be applied to a variety of time-series or other real-world systems, and the logistic map is chosen primarily as an example test-case system.

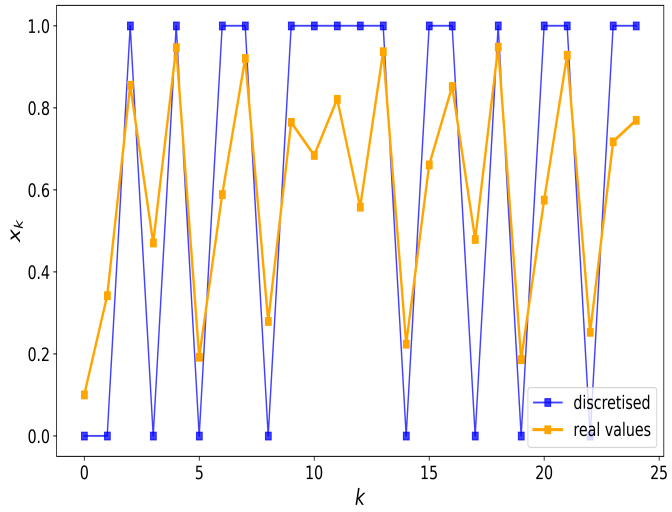
II. RESULTS

A. Problem description

The AIT coding theorem is framed in terms of random inputs or 'programs' and resultant output strings or patterns. Hence we will view the logistic map in this framework as follows: We will study binary sequence trajectories resulting from random discretised realisations of the logistic map

$$x_{k+1} = \mu x_k(1 - x_k) \tag{2}$$

where the inputs are the pair of values $x_0 \in$(0,1) and $\mu \in$. For the outputs, the map is first iterated to obtain a sequence of n real values (including x_0) in [0,1]. Similar to the field of symbolic dynamics [36], the real valued trajectory becomes a binary sequence output x by applying a threshold, and writing 1 if $x_k \geq 0.5$ and 0 otherwise [37].



(a)

FIG. 1. An example of a real-valued (orange) and discretised/binarised (blue) trajectory of the logistic map, with $\mu = 3.8$ and $x_0 = 0.1$. The discretisation is defined by writing 1 if $x_k \geq 0.5$ and 0 otherwise, resulting in the pattern $x =$ 0010101101111101101011011 which has length 25 bits.

Hence a binary sequence output x of 25 bits is generated for each input pair μ and x_0.

By way of example, consider choosing $\mu = 3.8$ and $x_0 = 0.1$, then from iterating Eq. (2) with $k = 0, 1, 2, ..., 24$ we obtain the real-valued trajectory

$$0.10, 0.34, 0.86, \ldots, 0.25, 0.72, 0.77$$

which after discretisation becomes the binary string

$$x = 0010101101111101101011011$$

Figure 1 illustrates the trajectory and discretisation procedure. The choice of n is a balance between being long enough that many different patterns can emerge, and short enough that decent frequency and probability estimates can be made without the need for excessive sampling. Different μ and x_0 inputs will yield different binary strings $x \in \{0, 1\}^n$. The main focus of this paper is examining the probability $P(x)$ of the different possible output binary strings x, and the presence of SB, upon uniform random sampling of input parameters $x_0 \in$(0,1) and μ in different parameter regimes of the interval $(0, 4.0]$.

These parameter intervals are are standard ranges in which the map is studied and it can be shown [38], for example, that for $\mu > 4$ trajectories are not confined to [0,1]; similarly if $x_0 \notin (0, 1)$ the behaviour can be unbounded or trivial. For some large values of μ (eg $\mu = 4.0$), almost all initial values x_0 yield complex and chaotic outputs [39], and the the distribution over (discretised) trajectories is roughly uniform, with little bias (note that chaotic trajectories do not occur until $\mu \geq 3.57$ [39]). Without strong bias, SB is not possible, which would preclude our investigation. On the other hand, for small values of μ, only fairly simple patterns can emerge, restricting the variety of patterns we

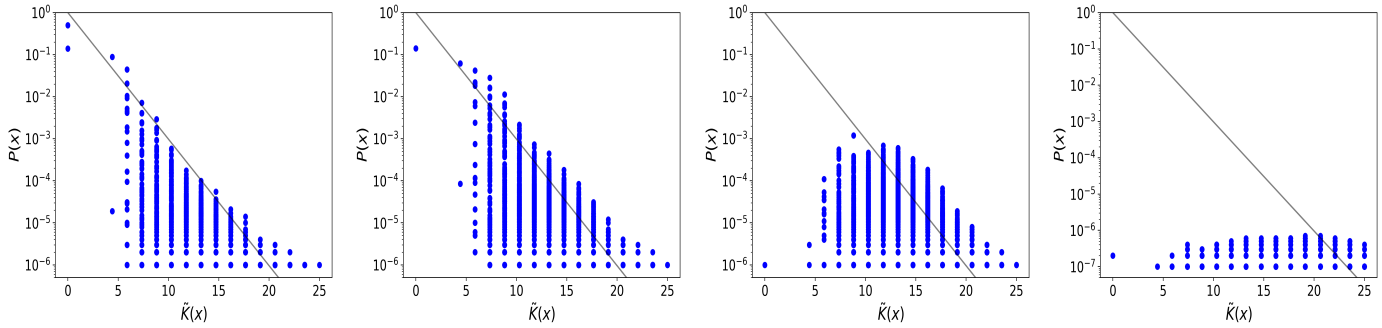


FIG. 2. Simplicity bias (SB) in the discretised logistic map from random samples with $x_0 \in (0, 1)$ and μ sampled in different intervals. Each blue datapoint corresponds to a different binary discretised trajectory x of length 25 bits. The black line is the upper bound prediction of Eq. (1). (a) Clear SB for $\mu \in (0,\ 4.0]$ with $P(x)$ closely following the upper bound, except for low frequency and high complexity outputs which suffer from increased sampling noise; (b) SB is still present for $\mu \in [3.0,4.0]$; (c) the distribution of $P(x)$ becomes more flat (less biased) and SB is less clear when $\mu \in [3.57,4.0]$ due to constraining the sampling to μ-regions more likely to show chaos; (d) the distribution of $P(x)$ is roughly uniform when using $\mu = 4.0$, with almost no bias, and hence no possibility of SB.

can study. Hence to allow for strong bias and both simple and complex output patterns, we will separately investigate various intervals for μ, namely μ sampled uniformly from (0,4], [3.0,4], [3.57,4.0], and also fixing $\mu = 4.0$.

Importantly, although uniform sampling can be expected to produce simpler trajectories more often (due to a larger fraction of μ values producing only simpler patterns) what we wish to study here is not just a vague general relation between complexity and the probability of binary strings, but whether the form of $P(x)$ follows closely that of SB of Eq. (1), with an exponential upper bound decay in probability with a linear increase in complexity.

B. Predicting the probability $P(x)$ of a sequence x using its complexity

1. *Theory*

The Kolmogorov complexity $K_U(x)$ of a string x with respect to U, is defined [11–13] as

$$K_U(x) = \min_p \{|p| : U(p) = x\} \tag{3}$$

where p is a binary program for a prefix optimal UTM U, and $|p|$ indicates the length of the program p in bits. Due to the invariance theorem [26] for any two optimal UTMs U and V, $K_U(x) = K_V(x) + O(1)$ so that the complexity of x is independent of the machine, up to additive constants. Hence we conventionally drop the subscript U in $K_U(x)$, and speak of 'the' Kolmogorov complexity $K(x)$. Informally, $K(x)$ can be defined as the length of a shortest program that produces x, or simply as the size in bits of the compressed version of x. If x contains repeating patterns like $x = 1010101010101010$ then it is easy to compress, and hence $K(x)$ will be small. On the other hand, a randomly generated bit string of length n is highly unlikely to contain any significant patterns, and hence can only be described via specifying each bit separately without any compression, so that $K(x) \approx n$ bits. Other more expressive names for $K(x)$ are *descriptional complexity*, *algorithmic complexity*, and *program-size complexity*, each of which highlight the idea that $K(x)$ is measuring the amount of information to describe or generate x precisely and unambiguously. Note that Shannon information and Kolmogorov complexity are related [40], but differ fundamentally in that Shannon information quantifies the information or complexity of a random source, while Kolmogorov complexity quantifies the information of individual sequences or objects. More details on AIT can be found in refs. [26, 41, 42].

Levin's coding theorem [28] states that

$$P(x) = 2^{-K(x)+O(1)} \tag{4}$$

where $P(x)$ is the probability that UTM U generates output string x on being fed random bits as a program (again we have dropped the subscript U). Thus, high complexity outputs have exponentially low probability, and simple outputs must have high probability. This is a profound result which links notions of data compression and probability in a direct way. $P(x)$ is also known as the *algorithmic probability* of x.

To estimate complexity in ref. [29] we used

$$C_{LZ}(x) = \begin{cases} \log_2(n), & x = 0^n \text{ or } 1^n \\ \log_2(n)[N_w(x_1 ... x_n) + N_w(x_n ... x_1)]/2, & \text{otherwise} \end{cases} \tag{5}$$

where the number of words $N_w(x)$ forms the basis for this complexity measure Lempel and Ziv [32], and where the simplest strings 0^n and 1^n are separated out because $N_w(x)$ assigns complexity $K = 1$ to the string 0 or 1, but complexity 2 to 0^n or 1^n for $n \geq 2$, whereas the true Kolmogorov complexity of such a trivial string actually scales as $\log_2(n)$ for typical n, because one only needs to encode n. Having said that, the minimum possible value is $K(x) \approx 0$ for a simple set, and so eg for binary strings of length n we can expect $0 \lesssim K(x) \lesssim n$ bits. Because for a random string of length n the value $C_{LZ}(x)$ is often much larger than n,

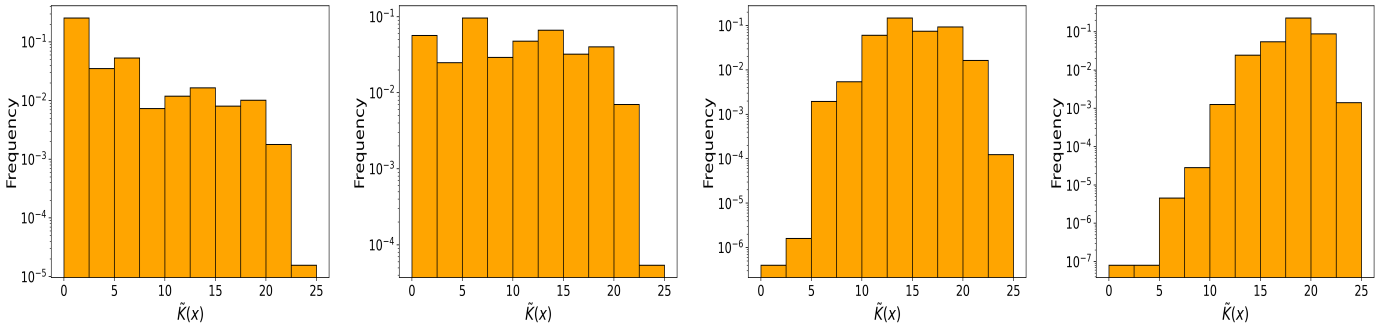


FIG. 3. The distribution $P(K(x) = k)$ of output complexity values, with $x_0 \in (0, 1)$ and μ in different intervals. (a) A roughly uniform complexity distribution for $\mu \in (0, 4.0]$, with bias towards lower complexities; (b) Close to uniform distribution of complexities for $\mu \in [3.0, 4.0]$; (c) the distribution leans to higher complexities when $\mu \in [3.57, 4.0]$; (d) the distribution is biased to higher complexities values when $\mu = 4.0$.

especially for short strings, we scale the complexity so that a in Eq. (1) is set to $a = 1$ via

$$\tilde{K}(x) = \log_2(M) \cdot \frac{C_{LZ}(x) - \min_x(C_{LZ})}{\max_x(C_{LZ}) - \min_x(C_{LZ})} \qquad (6)$$

where M is the maximum possible number of output patterns in the system, and the min and max complexities are over all strings x which the map can generate. $\tilde{K}(x)$ is the approximation to Kolmogorov complexity that we use throughout. This scaling results in $0 \leq \tilde{K}(x) \leq n$ which is the desirable range of values. We will assume that $b = 0$ throughout, which is a default assumption as argued and discussed in [29].

C. Simplicity bias appears when bias appears

Following the protocol for generating binary strings via logistic map trajectories described in the Introduction, we now examine the probability $P(x)$ that the discretised trajectory x is produced on random sampling of input parameters μ and x_0. Using Eq. (1), we make the prediction for the upper bound decay after 10^6 samples for the inputs (and 10^7 for $\mu = 4.0$). Figure 2(a) shows that the upper bound prediction agrees well with the data, and we see that the logistic map displays SB when sampling $\mu \in (0, 4]$, such that high probability outputs have low complexities, and complex outputs have low probability. The gradient prediction ($a = 1$) used no information about the map, except that we assumed that almost all 2^n outputs are realisable. Thus, we can predict $P(x)$ quite accurately, with high probability, just by using the complexity values output strings. Note that many output strings fall below the upper bound prediction as we expected from our earlier studies [29], but nonetheless it is known that randomly generated outputs tend to be close to the bound [31]. In Figure 2(b) we make a similar plot to (a) except that we restrict the sampling to $\mu \in [3.0, 4]$, but the pattern is similar, although SB is somewhat less clear than in (a). For Figure 2(c) the sampling interval was chosen so that $\mu \in (3.57, 4.0]$ which is the region for truly chaotic trajectories [38]. The bias and SB is less clear in this plot. In Figure 2(d) a plot is given in which μ is no longer sampled, but is fixed at $\mu = 4.0$, so that almost all x_0 values yield chaotic trajectories. As expected, there is very little bias in the distribution and the distribution of $P(x)$ is roughly uniform, and hence no SB. Figure 2(d) still shows some very simple strings, which can be understood from the fact that $x_0 \approx 0 \Rightarrow x_k \approx 0$ for $k \approx 1$, and similarly that $x_0 \approx 1 \Rightarrow x_k \approx 1$ for $k \approx 1$ even if the trajectories may become much more complex for $k \gg 1$. These trajectories with many 0s or 1s will have very low complexity. In the Appendix we list, for each complexity value, the binary strings which had the highest and lowest probability, for each graph (a)—(d) in Figure 2.

As a quick confirmation that SB (dis)appears along with bias, we compared measures of bias and SB while sampling μ in progressively smaller intervals, namely $[t, 4]$ for $t = 0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 3.6, 3.7, 3.8, 3.85, 3.9, 3.95, 4.0$. Bias was measured by the Shannon entropy of the distribution $P(x)$, and SB was measured by the slope of the upper bound (fit to the max probability value for each unique complexity value). A clear positive correlation is found (Pearson r =XX, p-val<XX), showing that SB appears in this system as soon as bias appears. This correlation is not a trivial finding: the entropy will be low if *any* small collection of outputs are highly probable, and because the logistic map is famous specifically for producing complex sequences, *a priori* this small set could have consisted of complex output sequences.

By itself, SB does not necessarily imply that the distribution of complexity values, $P(K(x) = k)$, will be skewed towards low complexities values. Rather SB says that the individual probability of some simple strings will be higher than that of more complex strings. Because the probability of observing a given complexity k depends on the product of the number of strings of complexity k *and* their individual probabilities, the distribution $P(K(x) = k)$ may in fact peak at high complexities r. To investigate this distribution in the logistic map, in Figure 3 we plot the distribution of complexities for the different sampling intervals of μ. As can be seen in Figure 3(a) and (b), for μ sampled from across (0,4] and (3,4] the distribution of complexities

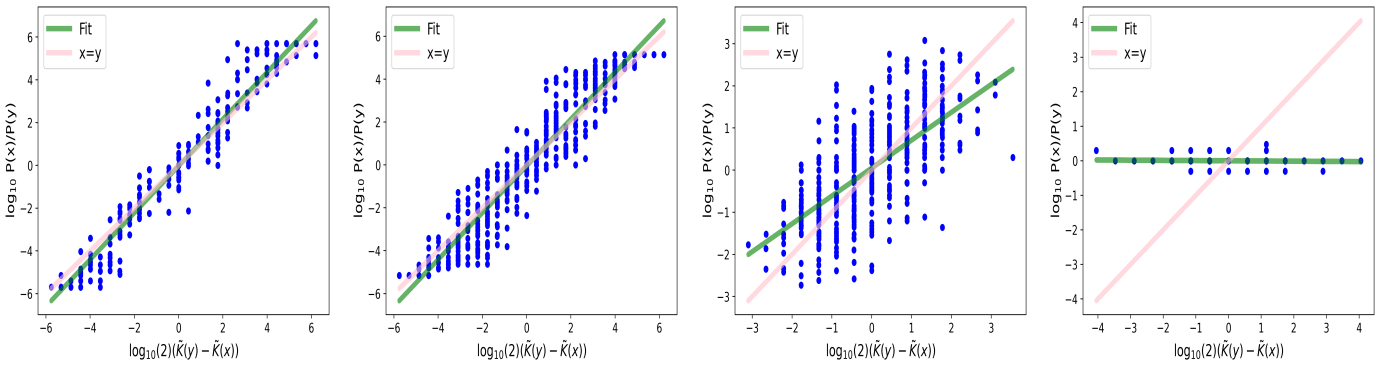


FIG. 4. Predicting the ratio $P(x)/P(y)$ for sampled pairs x and y using the complexity of sequences via Eq. (10). The fit to the data (green) and the prediction $x = y$ (pink) are shown. (a) Very accurate predictions can be made using only minimal knowledge of the map details, and without requiring fits to sampled data. The fit and the $x = y$ line are very close for $\mu \in (0, 4]$; (b) The fit and the $x = y$ line are quite close for $\mu \in (3.0, 4]$; (c) The fit and the $x = y$ line are not very close for $\mu \in (3.57, 4]$; (d) No predictive ability for $\mu = 4.0$.

is roughly uniform (at least on a log scale) and somewhat biased towards lower complexities. But for larger values of μ in Figure 3(c) and (d), the distribution shifts towards higher complexities. The appearance of a roughly uniform distribution is expected from AIT, because (to first order) there are 2^k strings of complexity k, each with probability 2^{-k}, such that the product is $2^k 2^{-k} = O(1)$ is independent of k, and uniform. For prefix complexity, it is known that the number of strings with complexity k is less than 2^k, and hence for a prefix UTM the distribution would be biased to lower k values to some extent. Nonetheless, the brief argument outlined is still valid as a first order approximation, and this rough prediction appears to hold reasonable well here. See also ref. [22] for similar arguments and results in a biological setting.

D. Complex and pseudo-random outputs

From an information theory angle, a question may arise: since the logistic map in Eq. (2) is itself simple with low Kolmogorov complexity, it might be supposed that any x it generates should be simple as well. So, how can we obtain large variations in complexity which would lead to large variations in probability? We can address this question analytically via bounding the complexity of a discretised trajectory written as an n-bit binary string x

$$K(x) \leq K(\text{logistic map}) + K(\mu, x_0) + K(n) + O(1) \quad (7)$$

This bound follows from the fact that x can be described precisely by describing: (a) the logistic map function in Eq. (2) with only a few bits because $K(\text{logistic map})$=$O(1)$ bits, (b) a pair of values of μ and x_0 which yield x with $K(\mu, x_0)$ bits, and (c) the length of the string (ie the number of iterations to perform) via n, which is at most $\log_2(n)$ bits (up loglog terms). From this upper bound, we see that *if* μ and x_0 are simple values with short descriptions, then $K(x) \ll n$ so that x must be simple. However, because μ and x_0 are (truncated) real numbers randomly chosen from their intervals, they need not be simple — rather the opposite — because almost decimal numbers sampled here given to d decimal places will have high complexity values of $\sim \log_2(10)d$ bits. So we can see that outputs need not be simple just because the map is simple, due to the presence of the complexity of the inputs in Eq. (7). Indeed, this argument accords with the fact that in Figure 2 we see that very many outputs x are realised, most of which must be complex because it is well known from AIT that almost all strings of length n are complex and only relatively few are simple.

Extending the preceding discussion on Eq. (7), it is also interesting to ask if there are simple input parameters which generate high entropy pseudo-random outputs. In fact there are, for example if $\mu = 4.0$ then almost all x_0 lead to chaotic trajectories including some simple initial values like $x_0 = 1/3$ which can be described with only a few bits, so that $K(\mu, x_0) = O(1)$ and so $K(x) \leq \log_2(n) + O(1)$ (up to additive loglog terms), but at the same time $\tilde{K}(x) \approx n$ bits because the Lempel-Ziv complexity measure cannot compress pseudo-random strings.

On reflection, these strings with $K(x) \ll \tilde{K}(x)$ must be 'rare' in the space, and not have individually high probability, otherwise we would not see SB, or at least see many apparent violators of the upper bound in Figure 2(a), which we do not. On the other hand, in Figure 2(d) they are perhaps not rare, but at the same time their individually probabilities are low, and hence no substantial violations of the SB is observed.

KD:Add sentence about long strings vs short strings, and affect on SB

E. No simplicity bias in the Bernoulli map

Before leaving the topic of complex trajectories with no SB, consider another prototypical chaotic map, the Bernoulli map, defined via the equation

$$x_{k+1} = (2x_k \mod 1) \quad (8)$$

with $x_0 \in (0,1)$. This map shows sensitive dependence on initial conditions because two inputs x_0 and y_0 which differ by $\epsilon \approx 0$ will eventually diverge in trajectories for large enough k. Given that it is also a 1-D chaotic system with a simple map, it is interesting to ask whether it shows SB like the logistic map does. A little reflection shows that this map does not show SB nor bias, even for a discretised version. The trajectory is defined by multiplying 2 by x_0 ignoring the integer part, so if x_0 is a random real number then the trajectory will be random and incompressible because ($2x_0 \mod 1$) will be another random number, assuming that x_0 is defined to a large number of decimal places. Multiplying a random number by 2 does not remove the randomness. Hence the binary discretised trajectory sequence would look almost random, with small and quickly decaying autocorrelation (see also Section 9.4 of ref. [41] for a similar conclusion). For bias and SB, it is necessary for random inputs to be able to lose a lot of their information and complexity (ie complex inputs must be able to produce simple outputs), but the Bernoulli map does not allow this.

F. Predicting $P(x)/P(y)$ ratios

We now turn to predicting the ratio of the probability of two randomly generated strings x and y. Because the strings are randomly generated, we expect both $P(x)$ and $P(y)$ to be close to the bound of Eq. (1), with high probability [29, 31], so that $P(x) \approx 2^{-a\tilde{K}(x)-b}$ and $P(y) \approx 2^{-\tilde{K}(y)-b}$. Recalling from Eq. (6) that we use $a = 1$ leads to the ratio

$$\frac{P(x)}{P(y)} \approx \frac{2^{-\tilde{K}(x)-b}}{2^{-\tilde{K}(y)-b}} = 2^{-\tilde{K}(x)}/2^{-\tilde{K}(y)} \tag{9}$$

$$\Rightarrow \log_{10}\frac{P(x)}{P(y)} \approx \log_{10}(2)(\tilde{K}(y) - \tilde{K}(x)) \tag{10}$$

notice that b is irrelevant, so that even if b is not known the prediction is unaffected.

To test this ratio prediction, we sampled two input parameter pairs — one pair to generate x and one to generate y — then compared $\log_{10}(P(x)/P(y))$ to the prediction of Eq. (10), then repeated this sampling 500 times. Figure 4 shows the predictions and data for the four different sampling regimes of μ used in earlier sections of this work. In Figure 4(a) the prediction is very accurate, and as before was made using almost no knowledge of the map details, or even any fitting from sampling. The linear correlation coefficient is 0.96 (p-val$< 10^{-6}$). In Figures 4 (b),(c) and (d) the predictions become progressively less accurate, leading to essentially zero accuracy in (d). Note that even if we could not estimate a, this ratio prediction method is still valid for determining which outcome x or y is more likely. Such ratio predictions are important because in many practical situations deciding whether some outcome x is more or less likely than y is the primary objective, rather than specifically knowing the probability values $P(x)$ and $P(y)$ themselves. The fact that we can predict these for $\mu \in (0, 4.0]$ and [3.0,4.0] quite accurately without any details of the map, or any fitting, is striking.

G. Predicting future sequence probabilities $P(x \to y)$ using conditional complexities $K(y|x)$

1. Theory

According to AIT, for a long sequence x which is randomly generated by a simple $O(1)$ map, the most likely future string y can be predicted by choosing the string y which yields the smallest $K(y|x)$ value [26]. Such prediction by compression in time series has been studied before, notably by Ryabko and coworkers [43, 44]. Here we advance the field of applied AIT by using SB and ideas from algorithmic probability to make probability predictions for all 2^n possible n bit binary series extrapolations. In other words, predicting the next n bits of a logistic map binary trajectory x, given the initial n bits (an n-step forecasting horizon). In time series analysis, it is often considered more useful to have predicted probabilities for all possible outcomes, known as *density forecasting* [45], not just stating the single outcome which is most likely. As far as we are aware, we are the first to do this.

The theory for these longer horizon predictions in AIT is based on infinite strings [26], but in our finite and computable setting we can proceed with the following argument: Given x, we can enumerate inputs whose outputs have x as their first n bits, then print the full $2n$ length string xy. This gives a discrete and computable probability distribution giving each $P(x \to y)$. We can use a Shannon-Fano code to make a prefix code using $\log_2(1/P(x \to y)) + O(1)$ bits to describe each xy, or simply each y by removing the initial segment x. So we have

$$K(y|x) \leq \log_2(1/P(x \to y)) + O(1) \tag{11}$$

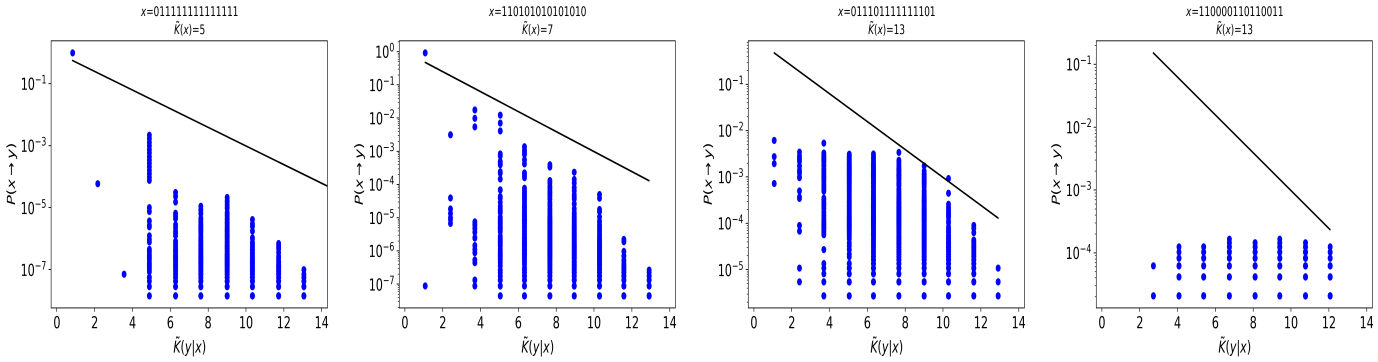
rearranging yields the upper bound

$$P(x \to y) \leq 2^{-K(y|x)+O(1)} \tag{12}$$

And then following our approximation and scaling procedure, we can write

$$P(x \to y) \lesssim 2^{-\tilde{K}(y|x)-b} \tag{13}$$

which is a weaker form of the full AIT conditional coding theorem [46]. From our earlier work [31], we expect this upper bound to be close for x, y pairs which are generated by random inputs. It is possible the bound is not tight for many output pairs. To perform the correct scaling of $\tilde{K}(y|x)$ requires estimating the total number $N_y(x)$ of possible y strings which can be accessed as future extensions of x [29]. A priori, we do not know the values of $N_y(x)$, so we will assume that (close to) all possible 2^n bit strings are possible for y, ie $N_y(x) \approx 2^n \ \forall x$. Having said that, this is a strong assumption which may not be valid, and hence the accuracy of the upper bound prediction may be strongly lacking. Nonetheless, even without a good a priori estimate for $N_y(x)$ (hence a), the ordering of predictions will not be affected because higher complexity outputs will still have lower probability predictions regardless of the value of a. So we can still predict some form of SB, even if we might not be able to predict the actual bound accurately. Finally, as before we will assume $b = 0$.



(a) (b) (c) (d)

FIG. 5. Predicting the future: Estimates of the probability $P(x \to y)$ of future extrapolations y of a given string x. Both x and y have length $n = 15$ bits. In (a) $\mu \in (0, 4.0]$ and (b) $\mu \in (3.0, 4.0]$ the transition probability $P(x \to y)$ upper bound decays with increasing conditional complexity $\tilde{K}(y|x)$; (c) using $\mu \in (3.57, 4.0]$ a weak relation of probability and conditional complexity is observed; (d) A roughly uniform distribution with no relation of probability to complexity when $\mu = 4.0$. Plot titles show the initial string x and its complexity $\tilde{K}(x)$.

.

Because for almost all $x, y \in \{0,1\}^n$ the strings x and y share almost no information, then $K(y|x) \approx K(y)$ for such strings. Hence Eq. (12) implies that for most x, y pairs $P(x \to y) \lesssim 2^{-K(y)}$. Combining this with Eq. (1) suggests that most strings y, we might expect

$$P(x \to y) \sim P(y) \tag{14}$$

to hold roughly. However, while this argument suggests that x may be largely irrelevant in estimating the probabilities $P(x \to y)$ *for most outputs* y, nearly all the probability mass is likely to be associated to only a small fraction of the possible outputs, those for which $K(y|x)$ is low. In fact, when $K(y|x) \approx 0$ but $K(y) \gg 0$ then we might expect

$$P(x \to y) \gg P(y) \tag{15}$$

and these outputs may absorb much of the probability mass. In other words, with high probability $P(y)$ may not in fact be a good estimator of $P(x \to y)$, even if it is for most outputs y. In the case that x is very simple with $K(x) \approx 0$, then $K(y|x) \approx K(y)$ for all y, and so we can also predict that Eq. (14) will hold when x is very simple.

To estimate the conditional complexity $\tilde{K}(y|x)$ we employ the approximation (as used in [47])

$$\tilde{K}(y|x) \approx \tilde{K}(xy) - \tilde{K}(x) \tag{16}$$

where $\tilde{K}(xy)$ is the compressed length of the concatenation of strings x and y. Note that for true prefix Kolmogorov complexity the relation $K(y|x) \approx K(x,y) - K(x)$ only holds to within logarithmic terms [26], but that is close enough for our purposes.

2. *Computational experiments*

Because our computational experiments are based on sampling, to get decent statistics for up to 2^n different strings y for each different initial string x, we must use very short binary strings to avoid excessive sampling. Here we choose $n = 15$ as a tradeoff between strings which are too short so that finite size effects dominate and complexity estimates are very noisy, and too long such that sampling errors are very large. We made 10^9 random samples with (x_0, μ), and μ in the four different regimes as above. Figure 5 shows example transition probability $P(x \to y)$ plots for a selection of different strings x (shown in the plot titles). For Figure 5(a) and (b) SB is observed, with decay in upper bound probability as conditional complexity $\tilde{K}(y|x)$ increases, as predicted by Eq. (13). The upper bound prediction is not a very tight bound on the data. It is not clear why this is, but may relate to the value of $N_y(x)$ having to be roughly estimated, as discussed above. Studying why exactly the upper bound slope prediction is not very accurate in these transition probability plots will be left for future work. But the general inverse relation of complexity and probability remains as a validated prediction, even if not the exact value of the slope. When μ is sampled from [3.57,4.0) SB is barely observed, and not at all when $\mu = 4.0$, as shown in Figure 5(c) and (d).

It is intuitively reasonable that given a simple history x, it will be easy to forecast the likely extrapolations, ie the conditional entropy over all 2^n futures will be low. On the other hand, given a complex history x, it is inuitive that the future extrapolations will maintain the complexity trend, and hence be hard to forecast, leading to high conditional entropy values. We made a brief computational check of this intuitive relation: The conditional entropy is defined as

$$H(Y|X = x) = \sum_y P(x \to y) \log_2(1/P(x \to y)) \tag{17}$$

over future strings y and testing for a correlation of this entropy the complexity of x, ie

$$\tilde{K}(x) \propto H(Y|X = x) \tag{18}$$

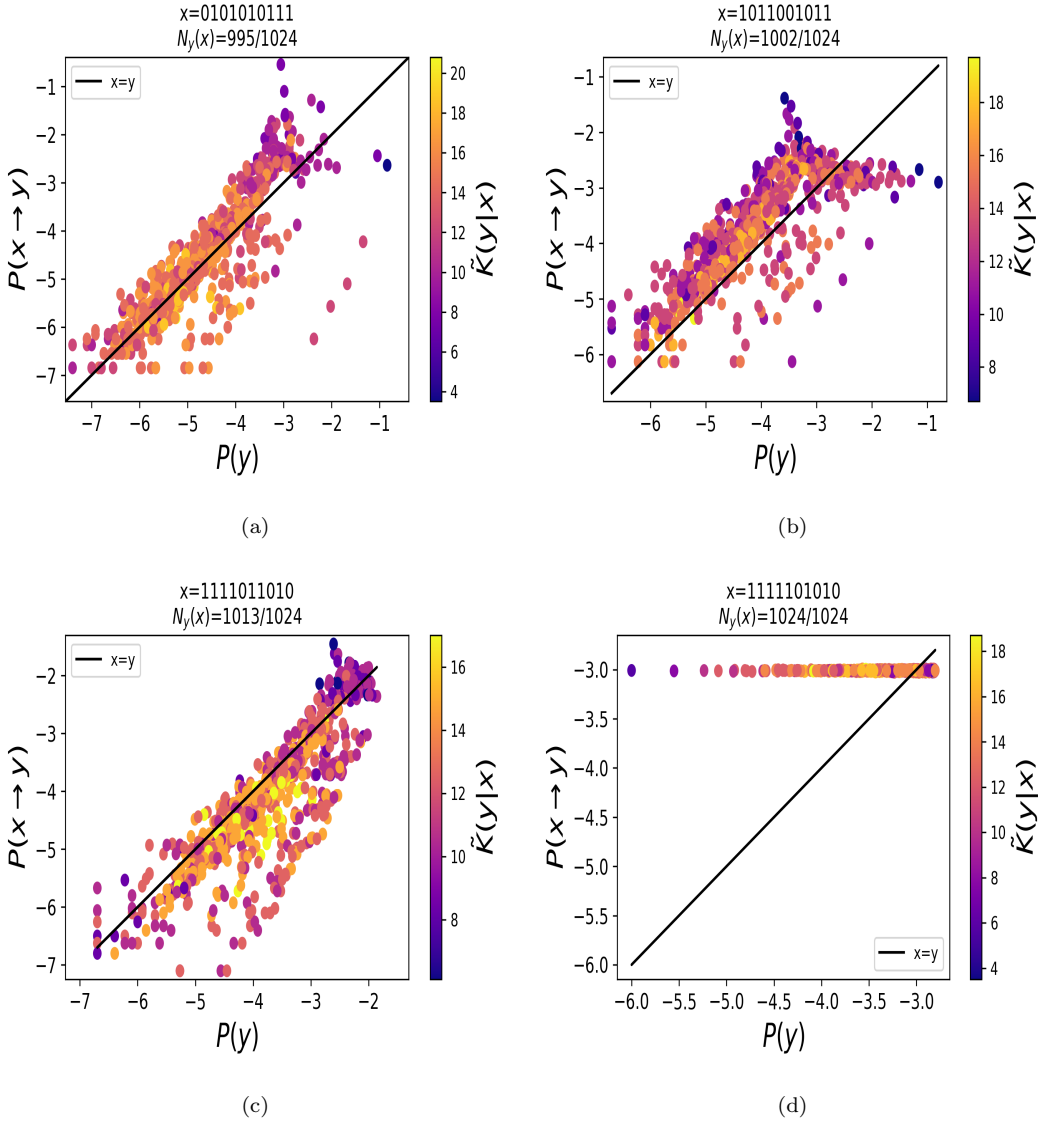


FIG. 6. Transition probabilities $P(x \to y)$ correlate with $P(y)$ for randomly chosen x, and most extrapolations y, except for when $\mu = 4.0$. The black line is $x = y$, it is not a fit. The points are colored by their conditional complexity values $\tilde{K}(y|x)$. Both x and y have length $n = 10$ bits. (a) $\mu \in (0, 4.0]$; (b) $\mu \in (3.0, 4.0]$; (c) $\mu \in (3.57, 4.0]$; (d) $\mu = 4.0$.

We tested this relation computationally for a random sample of 20 strings, and found that the linear correlation coefficient between $\tilde{K}(x)$ and $H(Y|X = x)$ is quite strong at $r = 0.77$ with p-value$< 10^{-4}$.

3. *Examining transitions*

Looking more closely at the transition probabilities $P(x \to y)$, we make some more observations now. In order to get a more fine grained look at these probabilities and especially better sampling statistics, we will shorten the length of x and y so that both are $n = 10$ bits. Eq. (14) pertains to almost all strings, so to get reliable statistics for all possible strings implies having shorter strings with less computational cost. Shorter strings also means that the complexity estimates are perhaps more noisy, however.

We performed 5×10^9 samples with $n = 10$, for the four different μ regions used throughout this work. The x strings were extrapolated to the different possible 10 bit future extensions y of x. We found essentially all $2^{10} = 1024$ possible extrapolations appeared in sampling for each of the x that we studied, ie $N_y(x) \approx 1024$ in each case. This suggests that all short binary strings are realisable in the logistic map, even though some may have very low probability. According to Eq. (14) we should expect a linear correlation between $\log P(x \to y)$ vs. $\log P(y)$ if we study all y values. In Figure 6 we show a selection of such plots, one for each μ sampling interval. In the titles of the plots, the number of different y values found is stated, as a fraction of the 1024 possible y strings found. For μ sampled in (0,4.0], (3.0,4.0], and (3.57, 4.0] we see a clear positive correlation, as expected. However, for $\mu = 4.0$ there is no correlation at all, but rather we see a roughly uniform distribution over ex-

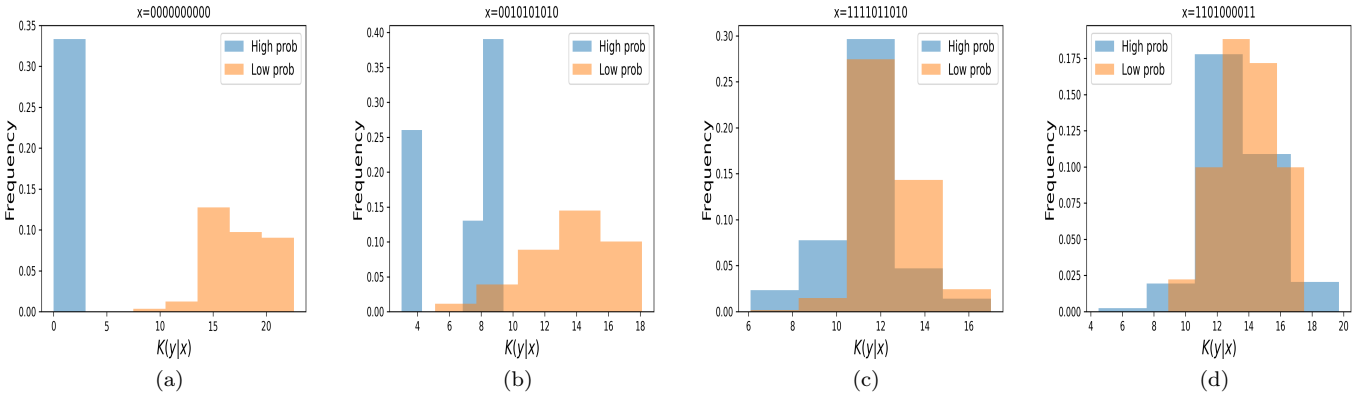


FIG. 7. Histograms of the conditional complexity $\tilde{K}(y|x)$ for strings y, separated into high probability and low probability sets. These example histograms show the kinds of behaviour observed for different μ sampling intervals. (a) With $\mu \in (0, 4.0]$ 90% of the time x will transition to y which has a smaller $\tilde{K}(y|x)$ value than typical $\tilde{K}(y|x)$ values; (b) With $\mu \in (3.0, 4.0]$ behaviour similar to as in (a) is seen; (c) With $\mu \in (3.57, 4.0]$ the high probability and low probability sets have similar distributions of $\tilde{K}(y|x)$; (d) With $\mu = 4.0$ the distributions are approximately the same.

trapolations y, similar to what we saw in preceding sections of this article. The average linear correlation value calculated from 20 random x strings was 0.88, 0.83, 0.87, and -0.02 for the four μ sampling intervals. The data points in Figure 6 are coloured by their conditional complexity value, $\tilde{K}(y|x)$. It is noteworthy that the data points in figure panels (a), (b) and (c) with very high $P(x \to y)$ values tend to have dark colours, indicating very low conditional complexity values. This behaviour was also discussed in relation to Eq. (15).

We argued above that while Eq. (14) should hold on average across all extrapolations y, this may not be very relevant to prediction because most of the probability mass may be assigned to a small number of y for which the correlation $\log P(x \to y) \sim \log P(y)$ does not hold, or is possibly very weak. We investigated this as follows: for each x string, we separated the corresponding y strings into the 'high probability' strings which together account for 90% of the probability $P(x \to y)$, and the 'low probability' strings which together account for only 10% of the probability. These high probability strings were typically only a small fraction of all y strings. We calculated the linear correlation of $\log P(x \to y)$ vs. $\log P(y)$ focussing only on the high probability strings, and found that the correlation was only 0.37, 0.52, 0.57, -0.02, which is quite weak correlation and substantially lower than values we found above for the full set of strings (except for $\mu = 4.0$ which is unchanged). Further, in Figure 7 we show histograms of the conditional complexity values for the high probability and low probability strings. For μ sampled from (0,4.0] and (3.0,4.0] we observe in Figures 7(a) and (b) that 90% of the transitions $x \to y$ are associated to y which have low conditional complexity. On the other hand, for μ sampled from (3.57,4.0] and $\mu = 4.0$ the distributions overlap strongly, especially for the latter in which the distributions are essentially identical.

III. DISCUSSION

We have numerically tested the ability to predict discretised sequence output trajectories in the logistic map using ideas from algorithmic information theory (AIT). Our main conclusion is that we observe simplicity bias (SB) and that we can make non-trivial model-free predictions regarding the probability of sequences, the ratios of sequence probabilities, and forecasting future sequence trajectory given some observed history, via conditional complexity. The accuracy of some of these a priori predictions is very high. Due to the qualitatively different behaviours exhibited by the logistic map for different μ values, we separately studied different regimes by sampling μ from (0,4.0], (3.0,4.0], (3.57,4.0] and also $\mu = 4.0$. In general, for SB and prediction accuracy was higher for μ sampled across the full range (0,4.0] and decreased for smaller ranges, until completely losing all accuracy for $\mu = 4.0$.

It is not especially noteworthy that by sampling $\mu \in (0, 4]$ we get simpler outputs more frequently because for most values in (0,4], only relatively simple trajectories can be generated. Instead, what *is* noteworthy is that the form of the decay in probability with complexity follows the SB upper bound well, with essentially no fitting. Indeed, this logistic map is one of the cleanest examples of SB we have, which is interesting given that we did not necessarily expect to observe SB when outputs can be pseudo-random. It appears that in this map pseudo-random outputs are sufficiently rare that they do not cause strong violations of the SB upper bound. Additionally, we found SB can be 'tuned' via altering the μ interval: sampling from the full interval (0,4] yields a biased (low entropy) distribution over output strings along with SB, while sampling from higher values of $\mu \approx 4$ yields low bias (high entropy) distributions and little or no SB. In future work, it may be fruitful to investigate which, if any, properties of the logistic (or other chaotic) map can be predicted or bounded using the SB bound. That is, if we assume that the bound holds, what other dynamical properties of the map might follow from

this. Even if these properties are already known, it would still be useful to see if they could be derived (perhaps more simply) from AIT ideas.

A weakness of our predictions is that they only constitute an upper bound on the probabilities, Figure 2(a) shows that many output trajectory patterns x fall far below their respective upper bounds. Following the hypothesis from [31], these low-complexity low-probability outputs are presumably patterns which the logistic map finds 'hard' to make, yet are not intrinsically very complex. Further, the presence of these low-complexity low-probability patterns may indicate the non-universal power of the map [31]. An avenue for future work would be studying what types of patterns occur far from the bound may, and possible approaches to improving on their probability predictions. Another avenue for future work we suggested was studying and improving the slope predictions of the transition probabilities.

The word "complexity" can take on many meanings [48, 49], and can be vague. In this work we are precise about what we mean by complexity, which is Kolmogorov complexity, and in practice using lossless compression methods which is a standard and theoretically motivated approximation to the true uncomputable quantity. Bialek et al. [50, 51] discuss complexity, and argue that Kolmogorov complexity lacks in its intuitive appeal for a measure of the complexity of time series. Many would agree that a series with rich complicated structure and long-range correlations is truly complex, whereas a random string of bits is merely an irregular and in a sense trivial pattern. In contrast, Kolmogorov complexity assigns the highest complexity to such random strings, precisely because they do not contain any correlations or structure. Having noted this, the discussion does not directly bear upon our work, because we are not studying 'complexity' in a general sense or trying to argue for one or other metric, rather we are studying specifically simplicity bias and AIT arguments as a mathematical framework for understanding probability and dynamical systems. AIT allows one to make quantitative bounds and predictions about various systems, as we illustrate here, regardless of whether or not the term 'complexity' is being used in it truest or most correct sense.

The application of AIT to real-world science problems suffers from several problems, including that (a) Kolmogorov complexity is uncomputable, (b) the results are framed and proved in the context of universal Turing machine (UTMs) while many real world maps are not Turing complete, and (c) results are valid up to to $O(1)$ terms and therefore, strictly, only accurate in the asymptotic limit of large complexities. Given these, it is surprising that AIT and Levin's coding theorem can be successfully applied at all. However, several lines of reasoning can help us understand why they are in fact applicable, at least approximately. Firstly in response to (a), although technically uncomputable, Kolmogorov complexity is fundamentally merely a measure of the size in bits of the compressed version of a data object. Hence, Vitanyi [25, 52] points out that because naturally generated data is unlikely to contain pseudo random complexities like π, the true complexity is unlikely to be much shorter than that achievable by every-day compressors. For more on this discussion, see ref. [52] and ref. [53] for work on short program estimates via short lists of candidates with short programs. Secondly, in response to (b) it is worth noting that the SB bound is specifically relevant in the computable (ie non-UTM) setting. Moreover building on pioneering work with small Turing machines [34, 35], Zenil et al. [54] have numerically studied coding theorem behaviour at different levels of the Chomsky hierarchy and found that it persists. Indeed, they also found close agreement between complexity estimates obtained from different levels, for the short binary strings which they studied. From the theoretical side, Calude et al. [55] developed an AIT for finite state machines (the lowest on the Chomsky hierarchy), deriving analogous results, suggesting that many fundamental ideas and results from AIT need not only apply to UTMs. Moreover, given that UTMs (at the top of the Chomsky hierarchy) and finite state machines (at the bottom of the Chomsky hierarchy) share many similar mathematical results, it is not unreasonable to assume that the results hold for other systems (such as RNA folding rules) which sit between the two extremes of the hierarchy. Thirdly, we argued earlier [29] that it is common in physics that mathematical formulae apply quite accurately well outside the (eg asymptotic) regions for which they have been proven. Although it is not possible to completely remove $O(1)$ terms in AIT [56], it is still an open and interesting question in theoretical computer science why asymptotic analysis (such as ignoring $O(1)$ terms) is valid and works so well in practical applications [57].

Further reasons to understand why the AIT coding theorem should work in real-world applications can be found in information theory research developed largely independently of Levin's work. The fundamental connection between probability and data compression has also been studied by Cover [58], Langdon [59] and Rissansen [60]. Since then, different communities — eg information theory [61, 62], optimal gambling strategies [63], and password guessing [64] — have studied and exploited the probability-compression connection without utilising Kolmogorov complexity or UTMs. In a review, Merhav and Feder [65] surveyed results in the area known as *universal prediction* and explicitly point to $2^{-LZ(x)}$ as an effective universal probability assignment for prediction based on the results of ref. [66] and others, where $LZ(x)$ is the Lempel-Ziv compression complexity measure, essentially the same as we use here. Additionally, Merhav [64] uses a conditional coding theorem predictor $2^{-LZ(y|x)}$ which is again very closely analogous to the AIT conditional coding theorem relation, $2^{-K(y|x)}$ [46]. These results, together with the arguments given above in response to (a)—(c) all support and motivate the application of AIT coding theorem work in science and engineering, and also extending fundamental research related to the coding theorem [67].

Mathematicians and physicists are fascinated by studying simple systems which can exhibit complex patterns, even while they are not technically random. In this context, Wolfram [68] investigated simple automata, and showed that some have high levels of computing power, leading to his conjecture that many natural systems can be Turing complete, ie that they can implement arbitrary algorithms, and hence produce complex patterns. Despite this focus on complexity, we observe in our work that complexity is

not actually that common — even though the logistic map is famous for its complex behaviour, within the space of all possible parameters, ie $\mu, x_0 \in \mathcal{R}$ and even restricting to $\mu \in (0, 4]$ and $x_0 \in (0, 1)$, chaos and 'complexity' only occur rarely. This observation accords with the fact that while Wolfram highlighted some rule sets that produce randomness, in fact most of his automata rules sets do not produce complex pseudo-random patterns. Coe et al. [69] analytically studied the question of when automata produce 'random' and complex dynamics, and found that relatively few are random.

In this work we have used the logistic map as a toy system in which to explore SB, probability prediction, and forecasting. Our goal was not specifically to advance the study of predicting chaotic trajectories. The methods we describe are general and can apply to other prediction systems. However, in future work it would be interesting to combine universal prediction methods like SB with other machine learning methods to predict chaotic time series (eg [70]), potentially aiding or improving the predictions (see also early [71, 72] and recent [73, 74] studies on Kolmogorov complexity connections to machine learning). One direction may be to use the SB bound to guide choosing a prior, as Hutter has described [75, 76].

XX Discuss learning as compression.....Ard suggested this, not sure what he wants to add here....

Acknowledgments: We thank Boumediene Hamzi for useful discussions.

[1] Mark M Wilde. *Quantum information theory*. Cambridge University Press, 2013.

[2] M. Mezard and A. Montanari. *Information, physics, and computation*. Oxford University Press, USA, 2009.

[3] Annick Lesne. Shannon entropy: a rigorous notion at the crossroads between probability, information theory, dynamical systems and statistical physics. *Mathematical Structures in Computer Science*, 24(03):e240311, 2014.

[4] Toby S Cubitt, David Perez-Garcia, and Michael M Wolf. Undecidability of the spectral gap. *Nature*, 528(7581):207–211, 2015.

[5] Cristopher Moore. Unpredictability and undecidability in dynamical systems. *Physical Review Letters*, 64(20):2354, 1990.

[6] Stephen Wolfram. Undecidability and intractability in theoretical physics. *Physical Review Letters*, 54(8):735, 1985.

[7] Seth Lloyd. Quantum-mechanical computers and uncomputability. *Physical review letters*, 71(6):943, 1993.

[8] Karl Svozil. *Randomness & undecidability in physics*. World Scientific, 1993.

[9] Anthony Aguirre, Zeeya Merali, and David Sloan. Undecidability, uncomputability, and unpredictability, 2021.

[10] Scott Aaronson. Guest column: Np-complete problems and physical reality. *ACM Sigact News*, 36(1):30–52, 2005.

[11] R. J. Solomonoff. A preliminary report on a general theory of inductive inference (revision of report v-131). *Contract AF*, 49(639):376, 1960.

[12] A.N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of information transmission*, 1(1):1–7, 1965.

[13] Gregory J Chaitin. A theory of program size formally identical to information theory. *Journal of the ACM (JACM)*, 22(3):329–340, 1975.

[14] Alan Mathison Turing. On computable numbers, with an application to the entscheidungsproblem. *J. of Math*, 58(345-363):5, 1936.

[15] C.H. Bennett. The thermodynamics of computation – a review. *International Journal of Theoretical Physics*, 21(12):905–940, 1982.

[16] Artemy Kolchinsky and David H Wolpert. Thermodynamic costs of turing machines. *Physical Review Research*, 2(3):033312, 2020.

[17] W.H. Zurek. Algorithmic randomness and physical entropy. *Physical Review A*, 40(8):4731, 1989.

[18] Markus P Mueller. Law without law: from observer states to physics via algorithmic information theory. *Quantum*, 4:301, 2020.

[19] Ram Avinery, Micha Kornreich, and Roy Beck. Universal and accessible entropy estimation using a compression algorithm. *Physical review letters*, 123(17):178102, 2019.

[20] Stefano Martiniani, Paul M Chaikin, and Dov Levine. Quantifying hidden order out of equilibrium. *Physical Review X*, 9(1):011031, 2019.

[21] P. Ferragina, R. Giancarlo, V. Greco, G. Manzini, and G. Valiente. Compression-based classification of biological sequences and structures via the universal similarity metric: experimental assessment. *BMC bioinformatics*, 8(1):252, 2007.

[22] Iain G Johnston, Kamaludin Dingle, Sam F Greenbury, Chico Q Camargo, Jonathan PK Doye, Sebastian E Ahnert, and Ard A Louis. Symmetry and simplicity spontaneously emerge from the algorithmic nature of evolution. *bioRxiv*, 2021.

[23] Alyssa Adams, Hector Zenil, Paul CW Davies, and Sara Imari Walker. Formal definitions of unbounded evolution and innovation reveal universal mechanisms for open-ended evolution in dynamical systems. *Scientific reports*, 7(1):1–15, 2017.

[24] Sean D Devine. *Algorithmic Information Theory for Physicists and Natural Scientists*. IOP Publishing, 2020.

[25] Paul MB Vitányi. Similarity and denoising. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20120091, 2013.

[26] M. Li and P.M.B. Vitanyi. *An introduction to Kolmogorov complexity and its applications*. Springer-Verlag New York Inc, 2008.

[27] R. Cilibrasi and P.M.B. Vitányi. Clustering by compression. *Information Theory, IEEE Transactions on*, 51(4):1523–1545, 2005.

[28] L.A. Levin. Laws of information conservation (nongrowth) and aspects of the foundation of probability theory. *Problemy Peredachi Informatsii*, 10(3):30–35, 1974.

[29] Kamaludin Dingle, Chico Q Camargo, and Ard A Louis. Input–output maps are strongly biased towards simple outputs. *Nature communications*, 9(1):761, 2018.

[30] Mark Buchanan. A natural bias for simplicity. *Nature Physics*, 14:1154, 2018.

[31] Kamaludin Dingle, Guillermo Valle Pérez, and Ard A Louis. Generic predictions of output probability based on complexities of inputs and outputs. *Scientific reports*, 10(1):1–9, 2020.

[32] A. Lempel and J. Ziv. On the complexity of finite sequences.

Information Theory, IEEE Transactions on, 22(1):75–81, 1976.

[33] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977.

[34] J.P. Delahaye and H. Zenil. Numerical evaluation of algorithmic complexity for short strings: A glance into the innermost structure of algorithmic randomness. *Appl. Math. Comput.*, 219:63–77, 2012.

[35] Fernando Soler-Toscano, Hector Zenil, Jean-Paul Delahaye, and Nicolas Gauvrit. Calculating Kolmogorov complexity from the output frequency distributions of small Turing machines. *PloS one*, 9(5):e96223, 2014.

[36] Douglas Lind and Brian Marcus. *An introduction to symbolic dynamics and coding*. Cambridge university press, 1995.

[37] Ali Kanso and Nejib Smaoui. Logistic chaotic maps for binary numbers generations. *Chaos, Solitons & Fractals*, 40(5):2557–2568, 2009.

[38] Arno Berger. *Chaos and Chance: An Introduction to Stochastic Apects of Dynamics*. Walter de Gruyter, 2001.

[39] Boris Hasselblatt and Anatole Katok. *A first course in dynamics: with a panorama of recent developments*. Cambridge University Press, 2003.

[40] Peter Grunwald and Paul Vitányi. Shannon information and Kolmogorov complexity. *arXiv preprint cs/0410002*, 2004.

[41] C.S. Calude. *Information and randomness: An algorithmic perspective*. Springer, 2002.

[42] P. Gács. *Lecture notes on descriptional complexity and randomness*. Boston University, Graduate School of Arts and Sciences, Computer Science Department, 1988.

[43] Boris Ryabko, Jaakko Astola, and Mikhail Malyutov. *Compression-based methods of statistical analysis and prediction of time series*. Springer, 2016.

[44] Konstantin Chirikhin and Boris Ryabko. Compression-based methods of time series forecasting. *Mathematics*, 9(3):284, 2021.

[45] Anthony S Tay and Kenneth F Wallis. Density forecasting: a survey. *Journal of forecasting*, 19(4):235–254, 2000.

[46] Paul MB Vitányi. Conditional kolmogorov complexity and universal probability. *Theoretical Computer Science*, 501:93–100, 2013.

[47] Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul MB Vitányi. The similarity metric. *IEEE transactions on Information Theory*, 50(12):3250–3264, 2004.

[48] Seth Lloyd. Measures of complexity: a nonexhaustive list. *IEEE Control Systems Magazine*, 21(4):7–8, 2001.

[49] Melanie Mitchell. *Complexity: a guided tour*. Oxford University Press, 2009.

[50] William Bialek, Ilya Nemenman, and Naftali Tishby. Complexity through nonextensivity. *Physica A: Statistical Mechanics and its Applications*, 302(1-4):89–99, 2001.

[51] William Bialek, Ilya Nemenman, and Naftali Tishby. Predictability, complexity, and learning. *Neural computation*, 13(11):2409–2463, 2001.

[52] Paul Vitányi. How incomputable is kolmogorov complexity? *Entropy*, 22(4):408, 2020.

[53] Jason Teutsch and Marius Zimand. A brief on short descriptions. *SIGACT News*, 47(1):42–67, March 2016.

[54] Hector Zenil, Liliana Badillo, Santiago Hernández-Orozco, and Francisco Hernández-Quiroz. Coding-theorem like behaviour and emergence of the universal distribution from resource-bounded algorithmic probability. *International Journal of Parallel, Emergent and Distributed Systems*, 34(2):161–180, 2019.

[55] Cristian S Calude, Kai Salomaa, and Tania K Roblot. Finite state complexity. *Theoretical Computer Science*, 412(41):5668–5677, 2011.

[56] Markus Müller. Stationary algorithmic probability. *Theoretical Computer Science*, 411(1):113–130, 2010.

[57] Scott Aaronson. Why philosophers should care about computational complexity. *Computability: Turing, Gödel, Church, and Beyond*, 261:327, 2013.

[58] TM Cover. Universal gambling schemes and the complexity measures of kolmogorov and chaitin. rep. no. 12, statistics dep, 1974.

[59] GLEN Langdon. A note on the ziv-lempel model for compressing individual sequences (corresp.). *IEEE Transactions on Information Theory*, 29(2):284–287, 1983.

[60] Jorma Rissanen. Universal coding, information, prediction, and estimation. *IEEE Transactions on Information theory*, 30(4):629–636, 1984.

[61] Meir Feder, Neri Merhav, and Michael Gutman. Universal prediction of individual sequences. *IEEE transactions on Information Theory*, 38(4):1258–1270, 1992.

[62] TM Cover and J.A. Thomas. *Elements of information theory*. John Wiley and Sons, 2006.

[63] Meir Feder. Gambling using a finite state machine. *IEEE Transactions on Information Theory*, 37(5):1459–1465, 1991.

[64] Neri Merhav and Asaf Cohen. Universal randomized guessing with application to asynchronous decentralized brute–force attacks. *IEEE Transactions on Information Theory*, 66(1):114–129, 2019.

[65] Neri Merhav and Meir Feder. Universal prediction. *IEEE Transactions on Information Theory*, 44(6):2124–2147, 1998.

[66] Eli Plotnik, Marcelo J Weinberger, and Jacob Ziv. Upper bounds on the probability of sequences emitted by finite-state sources and on the redundancy of the lempel-ziv algorithm. *IEEE transactions on information theory*, 38(1):66–72, 1992.

[67] Samuel Epstein. An extended coding theorem with application to quantum complexities. *Information and Computation*, 275:104660, 2020.

[68] S. Wolfram. *A New Kind of Science*. Wolfram Media, 2002.

[69] JB Coe, SE Ahnert, and TMA Fink. When are cellular automata random? *EPL (Europhysics Letters)*, 84(5):50005, 2008.

[70] Boumediene Hamzi and Houman Owhadi. Learning dynamical systems from data: A simple cross-validation perspective, part i: Parametric kernel flows. *Physica D: Nonlinear Phenomena*, 421:132817, 2021.

[71] Jürgen Schmidhuber. Discovering solutions with low kolmogorov complexity and high generalization capability. In *Machine Learning Proceedings 1995*, pages 488–496. Elsevier, 1995.

[72] Jürgen Schmidhuber, Jieyu Zhao, and Marco Wiering. Shifting inductive bias with success-story algorithm, adaptive levin search, and incremental self-improvement. *Machine Learning*, 28(1):105–130, 1997.

[73] Guillermo Valle-Pérez, Chico Q Camargo, and Ard A Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. *arXiv preprint arXiv:1805.08522*, 2018.

[74] Santiago Hernández-Orozco, Hector Zenil, Jürgen Riedel, Adam Uccello, Narsis A Kiani, and Jesper Tegnér. Algorithmic probability-guided machine learning on non-differentiable spaces. *Frontiers in artificial intelligence*, 3, 2020.

[75] Marcus Hutter. On universal prediction and bayesian confirmation. *Theoretical Computer Science*, 384(1):33–48, 2007.

[76] Marcus Hutter. *Universal artificial intelligence: Sequential decisions based on algorithmic probability.* Springer Science & Business Media, 2004.

Appendix A: Example outputs

The highest probability output string, and lowest probability output string, for each complexity value in Figure 2 are given here. The strings were n =25 bits long.

```
t= 0.0

k= 0.0  maxP= 0.499709  string= 0000000000000000000000000
k= 0.0  minP= 0.139038   string= 1111111111111111111111111

k= 4.41  maxP= 0.088398  string= 0111111111111111111111111
k= 4.41  minP= 1.9e-05     string= 1111111111111111111111110

k= 5.88  maxP= 0.044289  string= 0011111111111111111111111
k= 5.88  minP= 1e-06         string= 1111011101110111011101111

k= 7.35  maxP= 0.007191  string= 1110101010101010101010101
k= 7.35  minP= 1e-06         string= 1101010101010101010111111

k= 8.82  maxP= 0.002899  string= 0111010101010101010101010
k= 8.82  minP= 1e-06         string= 0011011011011011011011010

k= 10.29  maxP= 0.00058  string= 1010111011101110111011101
k= 10.29  minP= 1e-06       string= 0101101101101101101111110

k= 11.76  maxP= 0.000175  string= 1110101010101110111011101
k= 11.76  minP= 1e-06         string= 0100110001000100010001000

k= 13.24  maxP= 9.9e-05  string= 1011101110111010101110101
k= 13.24  minP= 1e-06      string= 1111101110101111011110101

k= 14.71  maxP= 5.5e-05  string= 1010101011101110101011101
k= 14.71  minP= 1e-06      string= 1110101101101001001001011

k= 16.18  maxP= 2.1e-05  string= 1011101011101110101011101
k= 16.18  minP= 1e-06      string= 0110010110111111111011011

k= 17.65  maxP= 1.4e-05  string= 1011111011101011101111101
k= 17.65  minP= 1e-06      string= 0001010110101010110111101

k= 19.12  maxP= 4e-06  string= 1011110111110101010111101
k= 19.12  minP= 1e-06   string= 1101110101110111011001001

k= 20.59  maxP= 3e-06  string= 1011010111010101110101101
k= 20.59  minP= 1e-06   string= 1011010010011101110110010

k= 22.06  maxP= 2e-06  string= 1001010011111010010010110
k= 22.06  minP= 1e-06   string= 0101110001001100111110010

k= 23.53  maxP= 1e-06  string= 1011011100111101010011101
k= 23.53  minP= 1e-06   string= 1011011100111101010011101

k= 25.0  maxP= 1e-06  string= 1011010111001111001000010
k= 25.0  minP= 1e-06   string= 1011010111001111001000010
```

For $t = 0.0$, the simplest strings are the string of 0s and string of 1s. There is a four-fold difference in their probabilities. For some of the medium complexity values, there is a larger range in the probabilities. For the maximum complexity of 25 bits, the highest and lowest probability output is the same.

```
t= 3.0

k= 0.0  maxP= 0.141318  string= 111111111111111111111111
k= 0.0  minP= 0.141318   string= 111111111111111111111111

k= 4.41  maxP= 0.062287  string= 011111111111111111111111
k= 4.41  minP= 8.5e-05      string= 111111111111111111111110

k= 5.88  maxP= 0.041925  string= 101010101010101010101010
k= 5.88  minP= 2e-06         string= 000000000001111111111111

k= 7.35  maxP= 0.02823  string= 111010101010101010101010
k= 7.35  minP= 1e-06       string= 111111111111111011101111

k= 8.82  maxP= 0.011337  string= 011101010101010101010101
k= 8.82  minP= 1e-06         string= 111111000100010001001000

k= 10.29  maxP= 0.002157  string= 101011101110111011011101
k= 10.29  minP= 1e-06         string= 111111111111010101111110

k= 11.76  maxP= 0.000733  string= 111010101010111011011101
k= 11.76  minP= 1e-06         string= 111110101101101101011111

k= 13.24  maxP= 0.000447  string= 101110111011101010111010 1
k= 13.24  minP= 1e-06         string= 001111010110110110110110 1

k= 14.71  maxP= 0.000181  string= 101010101110111010111010 1
k= 14.71  minP= 1e-06         string= 001010100100101100101111 1

k= 16.18  maxP= 8.4e-05  string= 101110101110111010111010 1
k= 16.18  minP= 1e-06      string= 000101111111101111110101 1

k= 17.65  maxP= 3.1e-05  string= 101110101010111111011101
k= 17.65  minP= 1e-06      string= 101111110111011010111101 0

k= 19.12  maxP= 1.2e-05  string= 101110101011111110111110 1
k= 19.12  minP= 1e-06      string= 111111100101101110100111 0

k= 20.59  maxP= 4e-06  string= 101101010101111101110110 1
k= 20.59  minP= 1e-06   string= 111001011110100101010110 1

k= 22.06  maxP= 3e-06  string= 101101010011101011011001 0
k= 22.06  minP= 1e-06   string= 010001011001111101010011 0

k= 23.53  maxP= 2e-06  string= 101111001011011010111011 0
k= 23.53  minP= 1e-06   string= 100101111010110110001001 0

k= 25.0  maxP= 1e-06  string= 101100111000101011110100 1
k= 25.0  minP= 1e-06   string= 101100111000101011110100 1
```

For $t = 3.0$, the simplest string was the all 1s string. The all 0s did not appear in the sampling. For the second lowest complexity value, there is a 1000-fold difference in the probability value between the highest and lowest, despite the fact that the strings are reverses of each other.

```
t= 3.57

k= 0.0  maxP= 1e-06  string= 111111111111111111111111
k= 0.0  minP= 1e-06  string= 111111111111111111111111

k= 4.41  maxP= 3e-06  string= 011111111111111111111111
k= 4.41  minP= 1e-06  string= 111111111111111111111110
```

```
k= 5.88  maxP= 0.00011  string= 10101010101010101010101
k= 5.88  minP= 1e-06  string= 00011111111111111111111111

k= 7.35  maxP= 0.000553  string= 10110110110110110110110
k= 7.35  minP= 1e-06  string= 11101101111111111111111

k= 8.82  maxP= 0.001202  string= 10010010010010010010010
k= 8.82  minP= 1e-06  string= 11110111011101111110111

k= 10.29  maxP= 0.000463  string= 10111011101110111010111
k= 10.29  minP= 1e-06  string= 11111101110111111111110

k= 11.76  maxP= 0.000689  string= 10110010010010010010010
k= 11.76  minP= 1e-06  string= 01101010110101101101010

k= 13.24  maxP= 0.000601  string= 10111010101110111101101
k= 13.24  minP= 1e-06  string= 11110111101101010010111

k= 14.71  maxP= 0.000328  string= 10111011101010101101101
k= 14.71  minP= 1e-06  string= 11101011010010011011011

k= 16.18  maxP= 0.000182  string= 10111010111010101110111
k= 16.18  minP= 1e-06  string= 00101010110010111111101

k= 17.65  maxP= 6.6e-05  string= 10111110111010111101111
k= 17.65  minP= 1e-06  string= 01001111111110011111001

k= 19.12  maxP= 1.9e-05  string= 10111010101111111101111
k= 19.12  minP= 1e-06  string= 10111101111110011110010

k= 20.59  maxP= 8e-06  string= 10110101111111011110101
k= 20.59  minP= 1e-06  string= 01101100101010011111101

k= 22.06  maxP= 3e-06  string= 10010111101110101101001
k= 22.06  minP= 1e-06  string= 10100110001111000010011

k= 23.53  maxP= 2e-06  string= 10110101001111110110011
k= 23.53  minP= 1e-06  string= 01001111101100010111110

k= 25.0  maxP= 1e-06  string= 10010111100001110100011
k= 25.0  minP= 1e-06  string= 10010111100001110100011
```

And finally the outputs for $t = 4.0$

```
t= 4.0

k= 0.0  maxP= 2e-06  string= 11111111111111111000000
k= 0.0  minP= 1e-06  string= 11111111000000000000000

k= 1.92  maxP= 1e-06  string= 01110111011101110111111
k= 1.92  minP= 1e-06  string= 01110111011101110111111

k= 3.85  maxP= 2e-06  string= 11111100000011111111000
k= 3.85  minP= 1e-06  string= 00100100100100100110111

k= 5.77  maxP= 2e-06  string= 00101010101010010100100
k= 5.77  minP= 1e-06  string= 10100100100100100111111

k= 7.69  maxP= 2e-06  string= 11111101000100010001011
k= 7.69  minP= 1e-06  string= 01111111111111111010011

k= 9.62  maxP= 3e-06  string= 10000110000000000000110
```

```
k= 9.62  minP= 1e-06  string= 11011011111110110111011111

k= 11.54  maxP= 3e-06  string= 11000000100001000000010111
k= 11.54  minP= 1e-06  string= 11011001101010101010001

k= 13.46  maxP= 4e-06  string= 11000100011111011001111100
k= 13.46  minP= 1e-06  string= 11101110111011011010010010

k= 15.38  maxP= 3e-06  string= 11101010111011100101111101
k= 15.38  minP= 1e-06  string= 11110001110010100000000010

k= 17.31  maxP= 4e-06  string= 10010100101111101011010000
k= 17.31  minP= 1e-06  string= 10001111101011100000001100

k= 19.23  maxP= 4e-06  string= 11100101111011101111100010
k= 19.23  minP= 1e-06  string= 01001000001111111010001100

k= 21.15  maxP= 3e-06  string= 10110000111010011011110011
k= 21.15  minP= 1e-06  string= 11001110101101100000111001

k= 23.08  maxP= 2e-06  string= 10100100011010111110011101
k= 23.08  minP= 1e-06  string= 00110101100011100101111101

k= 25.0  maxP= 1e-06  string= 10111100101000011101011101
k= 25.0  minP= 1e-06  string= 10111100101000011101011101
```

The simplest strings to appear were not the all 0s or all 1s strings, which did not appear at all in sampling. For $t = 4.0$, we see that the probabilities of strings is not related to complexity at all, and the distribution is essential uniform with each string appearing with frequency ≈ 1.