Turing and Randomness

Rod Downey*

School of Mathematics, Statistics and Operations Research,
Victoria University of Wellington,
PO Box 600, Wellington,
New Zealand.
rod.downey@vuw.ac.nz

Abstract. In an unpublished manuscript, Turing anticipated the basic ideas behind the theory of algorithmic randomness. He did so by nearly 30 years. Turing used a computationally constrained version of "measure theory" to answer a question of Borel in number theory. This question concerned constructing what are called "absolutely normal" numbers. In this article, I will try to explain what these mysterious terms mean, and what Turing did.

1 Borel, number theory and normality

1.1 Repeated decimals in fractions

Mathematicians have always been fascinated with patterns in numbers. At a very early stage of our education, we learn about the special nature of about decimal expansions of rational numbers. Recall that a real number is *rational* if it is a fraction: it can be expressed as $\frac{p}{q}$ for some integers p, q. The reader might remember from school, or maybe first year university, that numbers like $\sqrt{2}$ are not rational, and it can be shown that "most" numbers (in a precise mathematical sense) are irrational. Long ago, the Greeks showed that a real number between 0 and 1 is *rational* if and only if it has a finite decimal expansion, or a decimal expansion which repeats from some point onwards. For example, $\frac{1}{4} = 0 \cdot 25$, and $\frac{3}{7} = \cdot 428571428571428571....$ The astute reader will notice that we need a bit of care with the $\frac{1}{4}$ case as we could also think of it as $0 \cdot .2499999999999999999.....$ On the other hand, this alternative expansion does fit the bill about repeating from some point onwards. For simplicity, we will ignore such ambiguities. The reader might also remember that we can count using different bases[1]. *Binary* is a standard such base, where each place in the representation represents a power of 2. For example, $7 = 2^2 + 2^1 + 1$, and hence in binary, 7 would be written as 111. In base 3 we only use $0, 1, 2$, and express using powers of 3. In base 3, 7 becomes 21 which is $2 \times 3^1 + 1$. Note that bases can be bigger than 10, and we would have to invent symbols to represent the larger "digits". You may have

* Supported by the Marsden Fund of New Zealand.

[1] In fact using base 10 is a relatively recent convention.

noticed the ISBN code on a book. The ISBN code on a book is base 11, and uses x to represent 10[2].

The Greek result about repeats in the decimal representation of rationals remains true if we change the base from base 10 to any base. For instance, in base 3, $\frac{1}{4} = \cdot 02020202...$ In the discussion below, we will henceforth drop the decimal point and be concerned with infinite sequences.

1.2 Borel and normality

In 1909, Émil Borel [8] was interested in sequences which satisfied the *law of large numbers*. This law says that if you repeat an experiment many times, then the average value should be the expected value. If you toss a fair coin many times you expect as many heads half of the time. In base 10, this law says that frequency of choosing a digit between 0 and 9 is exactly what you would expect in the limit, namely $\frac{1}{10}$. What do we mean by "the limit"? Base 2 corresponds to tossing a coin. Over time, you would expect as many heads as tails. But this is only the *eventual long term behaviour*. If I toss a head, the next toss is independent of this toss. So with probability $\frac{1}{2}$ I would again get a head. But we expect that if the coin was fair, the law of large numbers states that it will all "even out in the long run".

Take such a sequence X representing an infinite collection of coin tosses. After s coin tosses, we can see how we are going so far by looking at how many heads we have seen in the first s tosses compared to the total number of tosses so far. We could examine the *ratio* at step s:

$$\frac{|\{X(k) = 1 \mid k \leq s\}|}{s}.$$

This denotes the number places of X we have heads ($X(k) = 1$) after s tosses, divided by s. If this was a fair coin this frequency should get closer and closer to $\frac{1}{2}$. Mathematically, we would say that the "*limit*" as $s \to \infty$ is $\frac{1}{2}$.

[2] Base 11 is chosen as it is prime, and this allows for certain error correcting aspects of the ISBN code. The code works by adding the first 9 numbers multiplied by their position from 1 to 9. Then we put a "check digit" at the end, which is the number needed to give 0 remainder after multiplying by 10 and dividing the sum by 11. For example, if the first 9 numbers were 019825080, they $1 \cdot 0 + 2 \cdot 1 + 3 \cdot 9 + 4 \cdot 8 + 5 \cdot 2 + 6 \cdot 5 + 7 \cdot 0 + 8 \cdot 8 + 9 \cdot 0 = 165$, which has remainder 0 when divided by 11, so the last digit would be 0. The system imagines someone ordering a book, say, and making an error. Using this check digit allows us to detect any single error such as writing 3 in place of 2 in this number, since the weighted sum of first nine numbers would be 170. The next multiple of 11 is $16 \times 11 = 176$, so if the number written on the order was correct the last digit would have been 6, but it was not, it was 0. ISBN codes will also detect any error caused by the transposition of two numbers, like switching 9 and 5. This only works as 11 is prime. The whole modern apparatus of error correcting codes enabling us to send information across noisy channels and figuring out what was sent *relies* on more high powered versions of arithmetic of primes to powers. For example, the Internet, and digital television, CD's, DVD's, MP3's would all be impossible without such mathematics.

More generally, we say that a sequence X is *Borel normal to base n* if it is exactly the same except we use an "n-sided coin". Using the notation above, if we represent X in base n, then for any digit between 0 and $n-1$, we need

$$\lim_s \frac{|\{X(k) = i \mid k \leq s\}|}{s} = \frac{1}{n}.$$

Borel defined a real number (an infinite sequence) to be *absolutely normal* if it was normal when written in any base.

In his 1909 paper, Borel observed that "almost every" real number is absolutely normal. Mathematically, this can be expressed by saying that the collection of absolutely normal numbers has Lebesgue measure 1, which corresponds to saying that if you threw a dart at the real line, with probability 1 it would hit an absolutely normal number.

Turing's work was motivated by the following two questions asked by Borel:

1. Can there be an irrational number normal to one base and not another?
2. Can you give an *explicit construction* of an absolutely normal number?

We are more concerned here with question 2. The reader might guess that numbers like $\sqrt{2}, \pi, \log 2, e$ etc are all absolutely normal. They are certainly explicit. $\sqrt{2}$ and $\log 2$ are what are called *algebraic irrational numbers*, in that they correspond to solutions of "algebraic" equations which only use simple arithmetic operations. For instance $\sqrt{2}$ is a solution to $x^2-2=0$. It is conjectured that *every algebraic irrational number* is absolutely normal. Our attempts to establish this conjecture are as pitiful as they could possibly be. *No explicit example has ever been proven to be normal even to a single base!*. Bailey and Crandall [2] in 2001, suggested an unproven "dynamical" hypothesis arising from "experimental mathematics". Assuming their hypothesis, we can prove that, for example, e and π, as well as every algebraic irrational number, are all absolutely normal.

Another way of constructing normal numbers was discovered by Turing's friend the economist Champernowne (who later held chairs in both Cambridge and Oxford) in 1933 as an undergraduate. What you do is simply write the digits in increasing order. This to base to, the Champernowne number is $C_{10} = 0.12345678910111213141516\ldots$. This can be done to any base. These are easily proven to be normal in the base that they are written, but we have no idea as to how to show that they are normal when expressed in terms of another base, which we think they are. In place of all of the integers, 1,2,3,... we could take some set $a_1 < a_2 < \ldots$ and use this set to also define a similar sequence. For example, if we use the primes we could write $C_P = 0.2357111317\ldots$. Remarkably, this sequence is known to be normal in base 10, as shown by Copeland and Erdös in 1946 in [17]. Unfortunately, it is not known if the primes written in base 2 result in such a normal sequence to base 2. The Copeland-Erdös base 10 proof relies on the "density" of primes in the relevant base.

As with many problems in number theory, it is easy to make conjectures, but hard to prove things!

2 Turing's approach

So failing to have *natural* examples of absolutely normal numbers, we might ask what would be an *explicit* example.

In a manuscript [46], thought to have been written around 1938, and never published in Turing's lifetime, Turing suggested that the an absolutely normal number n would be *explicitly* constructed if the number was *computable*; meaning that we could build some computer which could compute the expansion of n with any desired precision. Any irrational number met in "normal mathematics" will have this property, as all have rapidly converging approximations. For example, $e = \frac{1}{1!} + \frac{2}{2!} + \ldots$, and hence calculating $e = 2.718281828459045235\ldots$ is computable. This method of computing e was known to Euler as early as 1748. Similar series are known which rapidly converge to π, and this is true of any other "natural" constant.

As we will see, Turing's approach anticipates a line of research going back to the early 20th century, but really only realized in the early 1960's. Before we look at Turing's work we will briefly describe this body of work. This analysis will enable us to say exactly what Turing did, and how he did it.

3 von Mises and Ville

The late 1920s and early 1930s saw an adequate foundation for the theory of probability being built. This theory culminated in the celebrated work of Kolmogorov [25]. The foundation is based on the idea of the expected behaviour of some event in a probability space; and hence based around measure theory. I will try to explain these terms. The foundation is more or less what you would learn in school. The underlying concept is that we don't give any meaning to an explicit string (finite sequence of coin tosses, say) being random, but look at *expected behaviour* of *events*. Tossing a coin n times takes place in a "space of possibilities" (in this case, the collection of all binary strings of length n), and we would assign any sequence of length n probability 2^{-n} of occurring. (Here for the time being we will concern ourselves with base 2, heads or tails.) For example, tossing 4 times results in the space of possibilities $\{0000, 0001, 0010, 0011, \ldots, 1111\}$, each of which has probability 2^{-4} of occurring.

In the infinite case we look at the event of a sequence having a certain string as its initial segment. Such and event might be beginning with 101. The probability that we begin a sequence of coin tosses with head, tail, head is $2^{-3} = \frac{1}{8}$. The mathematical way to express this is that the *uniform Lebesgue measure* of the collection of all sequences extending 101 is 2^{-3}, or, more generally, extending some strings of length n is 2^{-n}. Using this idea, we are reduced to understanding similar *measures* on *probability spaces*. There is a huge and deep theory of these objects way beyond the scope of this article, but for our purposes it suffices to keep the coin tossing example in mind.

Less commonly known is that Kolmogorov's foundation came well after an earlier attempt to try to give meaning to the notion of randomness for an *individual sequence*. This is completely contrary to Kolmogorov's approach since

in that approach all sequences are equally likely. However, none but the most contrary would suggest that the sequence $S = 010101010101010...$ was random. The question is: "How to differentiate between such a sequence and the kind of sequence claimed to come from, say, quantum effects?". As we mentioned above, the first thing you learn in elementary probability theory at school is that any two sequences of coin flips of length n with an unbiased coin are equally likely, and have probability 2^{-n}.

Clearly we could try to apply *tests* to a sequence to test its apparent randomness. A random sequence should pass the law of large numbers and hence in the limit should the number of 0's and 1's should have the same frequency.

In 1919, Richard von Mises [48] attempted to give a basis for randomness based upon a generalization of the law of large numbers which used arbitrary chosen subsequences. Von Mises' idea was to consider any possible *selection* of a subsequence and ask that the selected subsequence also be normal. The point here is that the sequence $S = 010101010101010...$ is certainly not random but is does obey the law of large numbers as there are equal numbers of 0's and 1's in the limit. But clearly if we *selected* every second bit we would get all 1's. It is not reasonable that selecting every second bit of a random number should result in all 1's, so S *fails this randomness test.*

Von Mises generalized this idea as follows. Let f be a *place selection function.* That is, $f(i)$ is the i-th place selected. In the law of large numbers $f(i) = i$. In the proof that S is not random, $f(i) = 2i$. f should be increasing. Von Mises asked that we replace counting

$$\frac{|\{X(k) = 1 \mid k \leq s\}|}{s}$$

(which we used in the law of large numbers) by

$$\frac{|\{X(f(k)) = 1 \mid k \leq s\}|}{s},$$

the ratio of the number of *selected places which give heads* divided by the number of selected places.

When should X be regarded as random? We would perhaps say that X is random if and only if it is random for *all possible selections.* There is a big problem with this idea. No sequence X can be random for *all selection functions.* Since any nontrivial X has infinitely many 0's, we could simply have an f which chose the positions of the zeroes of X in increasing order. But surely this counterexample is spiritually unfair: we are trying to capture the notion that we should not be able to *predict* the next bit, and this f is chosen after defining X. It is always easy to predict the answer if you know it in advance! The question is "What kinds of selections should be allowed to capture the notion of *prediction*?" Our intuition is that prediction is somehow a computational process, and hence from a modern perspective we would guess we should obey all laws generated by *computable selections.*

However, von Mises work predated the work in the 30s clarifying the notion of computable function, famously culminating in Turing's [43]. Thus von Mises had

no canonical choice for "acceptable selection rules". Wald [49, 50] showed that for any *countably infinite*[3] collection of selection functions, there is a sequence that is random in the sense of von Mises. [15] proposed restricting f to (partial) computable increasing functions. A sequence X which defeats all such f is now called *computable stochasticity*, and *partial computable stochasticity*, in the case of partial computable selections.

Notice in these discussions we are abandoning the idea of *absolute randomness* in some metaphysical sense. Such a notion of "absolute randomness" is used, for instance, by physicists in an imprecise way[4]. These discussions lead to *algorithmic randomness*, where we will use tools from Turing's computation theory to quantify what randomness means. More precisely, since we have abandoned the notion of absolute randomness to "levels of randomness", we use these tools to calibrate by the computational strictness of the tests the randoms must pass. The harder the test, the more random we regard the sequence to be.

Does von Mises-Wald-Church computable stochasticity capture the notion of algorithmic randomness? Alas, the answer is no. A savage blow was dealt to von Mises' programme by Ville [47] who proved that for any countable collection of selection functions, there is a sequence X which is random relative to all of them, *but for each n, there are always more $0's$ in $X \upharpoonright n$, the first n bits of X, than there are 1's.* I think if you walked into a casino and were told that there would always be more tails than heads, you would believe it to be a biased coin, and could probably win some money! Ville suggested that perhaps we could add von Mises versions of another law (the law of iterated logarithms) to the list. Perhaps Von Mises's laws together with the additional laws, captured the notion of algorithmic randomness. This all seems very *ad hoc.* The immediate natural question is : "Is there yet another such Ville-like counterexample for this set of laws?" Again the answer is yes, as discussed in Downey-Hirschfeldt [20]. So again we fail to capture algorithmic randomness.

4 Martin-Löf

The above was how matters stood until 1966 and the work of Per Martin-Löf [33]. Martin-Löf effectivized the notion of a *null set* from classical measure theory, and gave a definition of algorithmic randomness based on this idea. Let me explain.

Recall that we are working in the space of infinite sequences of 0's and 1's. As discussed above *Lebesgue measure* of the collection of all sequences beginning with some string σ is $2^{-|\sigma|}$, where $|\sigma|$ denotes σ's length. Recall that this is the

[3] For example, all machine programmes can be thought of as being represented "digitally" as sequences of 0's and 1's. Hence we can think of programmes/algorithms as simply being numbers, and this is a countable (countably infinite) collection as we can "count" the numbers $1, 2, 3, \ldots$.

[4] It is ongoing research to determine what kind of randomness is actually necessary for parts of physics like the Copenhagen interpretation of quantum mechanics. Deep questions in physics concern whether the universe can physically generate any of the following: randomness, non-randomness, computability, incomputability!

probability that a sequence begins with σ. Our example was the measure of sequences beginning with 101 being $\frac{1}{8}$. This measure being the probability that a sequence begins with head, tail, head.

Martin-Löf's idea is that tests of randomness like the law of large numbers are computable ways of narrowing down the possible X's which could be considered as random. More concretely, consider the example $S = 01010101....$ We *don't* want this to be random. So we would first *hope* a random X avoided the collection of all sequences starting with a 0, namely the "cone" $[0] = \{0Y : Y \text{ is an infinite sequence of } 0's \text{ and } 1's\}$. This cone of sequences has measure $\frac{1}{2}$, since half of all sequences begin with a 0 and half with a 1. Failing to pass this first test would entail the sequence being of the form $X = 0Y$. We pass the test if the sequence began with $1Y$. Suppose we fail the test and so $X = 0Y$. We would then hope that X avoided [010], the set of all sequences starting with 010, being of measure $\frac{1}{8}$. If X does *not* begin with 010, we are happy as it passes the test. If we fail and $X = 010Z$, we would move on to test with [01010] having even lower measure (probability). We continue this for each such test. We are happy if X eventually passes the test, as it fails to be in one of these cones.

How to generalize this to all computable tests[5]? Martin-Löf's idea is to have *levels* of tests, $T_1, T_2, \ldots$. Each computer (predictor) is allowed to generate its own collection of tests. Fix one such computer M and one such sequence. T_1 is the easiest to pass. Since M is generating T_1, this can be imagined a collection of strings $s_1, s_2, \ldots$ where we are saying we don't want X to begin with s_1, nor s_2, nor s_3, etc. Of course, we need to be fair since we could never pass the test saying that that X cannot start with either a 0 or a 1!. So the tests will need to get smaller in measure. Standardly, T_1 is a test of measure $\leq \frac{1}{2}$, and T_k has measure $\leq 2^{-k}$. Since these are computable tests, we imagine the computer as generating this list of predictions, $T_1 = \{[s_1], [s_2], \ldots\}$. For example, suppose we did not want any sequence with more 0's than 1's. Then T_1 could be the collection $\{[0], [100], [010], \ldots\}$, of overall measure $\frac{1}{2}$. These are M's predictions of initial segments of X. M is wrong if *none of them are initial segments of* X. If X extends one of these initial coin tosses, it *fails* test number T_1, since we predicted an initial segment of X. In the example, X extends 100, say, it fails T_1. As remarked above, it is important for fairness that the tests get smaller as we are only interested in the *limiting* behaviour. So we could imagine T_2 as being similarly generated by the same computer M as $[r_1]$, $[r_2] \ldots$ but now we would ask that the total measure in T_2 is $\leq \frac{1}{4}$, and so on. For example, for testing "more 0's than 1's" the next test could be $\{[001], [000], [10001], [10010], \ldots\}$, which *refines* the first test. In general, the total measure of T_k is at most 2^{-k}, and hence in the limit the measure is 0, so the property the machine tests is "computably rare" or, equivalently, "computably statistically negligible". X passes this test if at some stage k, X does not extend *any member of* T_k; none of machine M's "level k" predictions about the initial segments of X are correct.

[5] The following paragraph is a little technical, but the reader should try to get the gist of what is being done as it is pretty important. It is summarized by the subsequent paragraph.

For example, if X began with 10011, it would pass T_1 but fail T_2 and hence fails the Martin-Löf test that this M represents. We say that X is Martin-Löf random if and only if it pass all such tests. Almost all sequences are Martin-Löf random, and this notion of randomness has been shown to provide a robust basis for algorithmic randomness.

In summary, Martin-Löf re-formulated all the laws at an abstract level based upon effectivizing classical measure theory. The measure of a set of sequences is the mathematical version of its probability. Martin-Löf randomness says that we regard X as random if and only if it passes all computable tests of computable measure 0. That is, X avoids all computable tests of non-randomness of computable probability 0.

5 Solomonoff, Kolmogorov, Levin, Chaitin, and Schnorr

There were other approaches to a definition of algorithmic randomness. For (finite) strings, a suitable definition was formulated by Kolmogorov, who argued that if a string had identifiable regularities, then we would be able to *compress* the string and hence it could not be random. Here we think of a machine M as a descriptional process, an algorithm τ being processed by M to give an output σ. Then τ is a description of σ; a programme that M will use to print σ. The length of τ is a measure of its length as a description, and random σ should be hard to describe.

As an illustration, consider the sequence $\sigma = 010101010\ldots$ (1000 times). A short description τ of σ is "print 01 1000 times". For the purposes of this discussion we will regard this description as length 16 by simply counting the number of symbols. The point is that this length 16 description is producing an output σ of length 2000. We are exploiting the *regularities* of this particular σ to compress the description. Kolmogorov's intuition was that for a *random* sequence there would be *no regularities* and then the only way to describe σ was to essentially use σ itself.

More precisely, a string of length n should be random relative to some descriptional process M is and only if the only way to describe it would be to use a string of the same length. Like white noise, a random string is *incompressible*. You can test this with children. Suppose you had a maze which was shaped as a binary tree of height 6 with boxes in the end. That is, there are 2^6 possible routes to get to the boxes. One of the boxes has money in it, and this box is known to the child. A child is to tell us which box has the money. Now if the box is the leftmost one all they have to say is always turn left. If the box was found by say, left-right-left etc, again this is easy to describe. If the place of the prize is determined randomly, the child would need to tell us the whole sequence[6]. This compressibility approach gives rise to the what is now called *Kolmogorov complexity.*

[6] In the text Li-Vitanyi [31], there is a report of this experiment being done on communication of ants, of the "food-finding" ants describing the location of the food box in a similar set-up to the "food-gatherer" ants.

The point here is that there are many ways to describe some string. Suppose I gave you a very long string σ whose bits were the 100th bits of π in its binary expansion. This string is far from random, but such non-randomness would be hard to perceive, until this fact was discovered. At the point we discover this hidden pattern, the the descriptional complexity σ might go from appearing to be quite random (using σ to describe σ) to be very easy. Mathematically, the complexity of this string of length n would go from complexity like n, the length of σ, to about $\log n + O(1)$. This latter one being the number of bits needed to describe n in binary, plus the $O(1)$ for the fixed program printing out that many 100th bits of π.

A natural guess would be that an (infinite) sequence X is random if and only if for all n, the first n bits of X are incompressible in the sense outlined above. Without going into details, the classical (plain) Kolmogorov complexity is not quite the correct notion for infinite sequences[7]. The reason for this is that plain complexity above fails to capture the *intentional meaning* of the information of the bits of τ coding the information of σ. There are at least three ways around this and all rely on restricting the kinds of devices M allowable. Using prefix-free codes[8] by Levin, Chaitin and Schnorr, and in a certain sense even earlier by Solomonoff [42], there is a natural notion called *prefix-free Kolmogorov complexity*, K such that X is Martin-Löf random if and only if the the prefix-free complexity of the first n bits of X is n or more (up to a constant factor).

Finally, as developed by Schnorr there is a gambler's version of randomness, spiritually close to von Mises. A *martingale* is a betting function on strings satisfying the expected fairness condition,

$$f(\sigma) = \frac{f(\sigma 0) + f(\sigma 1)}{2}.$$

That is, the capital you have accumulated at σ has to be shared over the bets for $\sigma 0$ and $\sigma 1$. A martingale *succeeds* on a sequence X if and only if using f to make bets we win an infinite amount of money. Schnorr showed that X is Martin-Löf random if and only if no (appropriately) algorithmic martingale succeeds on X. So von Mises's intuition about prediction has a version that works after all!

---

[7] The reason is that in the formulation above, a description τ into M gives more than simply the *bits* of τ to generate σ, but also the *length* of τ, and hence $|\tau| + \log |\tau|$ (since we know τ is the length of the program) much information. We know not only that this is a program, but that the program stops. Using this fact, we can show that for long strings, the complexity of $X \upharpoonright n$ must always go below n for some lengths n.

[8] That is, the descriptions are like telephone numbers, if τ and ρ are input descriptions to M and both give outputs, then τ is not a prefix of ρ. No telephone number should be a prefix of another! The point here is that in plain complexity both 100 and 1001 could be codes of programs (descriptions). Implicitly, if we are using 100 and not 1001, we need a stopping signal after the second 0. This means we are using the fact that 100 is a program, and we are *not* using any extension of it. Telephone numbers avoid this extra information implicitly being used.

In summary, there are three basic approaches to defining random sequences. They are the statistician's approach, that a random sequence should have no computably rare properties; the coder's approach, that a random sequence should have no regularities which allow for compression; and the gambler's approach, that we should not be able to predict the bits of a random sequence. The three approaches towards defining the notion of an algorithmically random sequence all naturally give the same random sequences.

6 Turing's work

The above describes the main lines or development of algorithmic randomness up to the point that Martin-Löf gave a suitable definition. Recall that Turing was interested in absolute Borel normality. Clearly, a random sequence will be absolutely normal. As we have already seen, if the frequency of 1's is less than that of the 0's you could easily have a betting strategy to allow you to make infinite capitol in the limit. Take an infinite binary sequence. Suppose it is not normal in base 10, and in base 10 the frequency of 5's is lower than expected infinitely often. In base 10, you could formulate a betting strategy to exploit this fact. But this will translate back to a strategy which could be used about the binary expansion, since there is a very tight computable relationship between the expansions of the two bases. This kind of consideration means that randomness to any base implies randomness to any other base.

In an unpublished manuscript, Turing attacked the question of an explicit construction of an absolutely normal number by interpreting "explicit" to mean *computable*. His manuscript entitled *"A note on normal numbers"*, presumably written in 1938, presents the best answer to date to Borel's second question: an algorithm that produces absolutely normal numbers. This early proof of existence of computable normal numbers remained largely unknown because Turing's manuscript was only published in 1997 in his Collected Works, edited by J. L. Britton [46]. The editorial notes say that the proof given by Turing is inadequate and speculate that the theorem could be false. In [4] Becher, Figueira and Picci reconstructed and completed Turing's manuscript, trying to preserve his ideas as accurately as possible and correcting minor errors.

As Becher [3] remarks, the very first examples of normal numbers were independently given by Henri Lebesgue and Waclaw Sierpiński[9] in 1917 [27, 41]. They can also be modified to give computable instances by giving a computable reformulation of the original constructions of 1917, as shown by Becher and Figueira [5]. Until recently, as we see below, together with Turing's algorithm these are the only known constructions of computable normal numbers. It is clear that Turing was unaware of the limiting constructions given in [27, 41]. In any case what is interesting is the way that Turing solved Borel's question; and how it relates to Martin-Löf approaches to randomness.

What does Turing's construction do? His paper says the following:

[9] Both published their works in the same journal issue, but Lebesgue's dates back to 1909, immediately after Borel's question.

> Although it is known that almost all numbers are [absolutely] normal no example of [an absolutely] normal number has ever been given. I propose to show how [absolutely] normal numbers may be constructed and to prove that almost all numbers are [absolutely] normal constructively.

Turing's idea is to first give an extension of the law of large numbers to "blocks" of numbers. Spiritually, the reader can see that not just single digits, but fixed blocks of numbers should occur with the desired frequency in a random sequence, else you could bet on the knowledge that they don't. Why blocks? The idea is that translating between bases results in correlations between blocks of integers in one base and blocks in another. Realizing this, key to Turing's construction is the following: Turing regards blocks as generating "tests" with small measure, and hence if we effectivize the notion of null set appropriately, and avoid such tests, we should have an an absolutely normal number.

In the unpublished manuscript, Turing then *develops a computable version of measure theory, and proves that the sequences which are* not *absolutely normal have* computable measure 0. *Therefore, Turing concludes, there must be a computable sequence which is absolutely normal.*

Here is what Jack Lutz said of this in a lecture at the conference *Computability, Complexity and Randomness, 2012* at Cambridge:

> "Placing computability constraints on a nonconstructive theory like Lebesgue measure seems a priori to weaken the theory, but it may strengthen the theory for some purposes. This vision is crucial for present-day investigations of
> – individual random sequences,
> – dimensions of individual sequences,
> – measure and category in complexity classes, etc."

What is fascinating here is the clarity here of Turing's intuition: To construct absolutely normal numbers, take the classical proof that almost all numbers are, and re-do this as a computable version. This is essentially the celebrated approach of Martin-Löf, except that we must be more delicate, since no computable sequence can be Martin-Löf random. This happened almost 30 years before Martin-Löf's work. Turing's analog of Martin-Löf tests are sensitive enough to exclude absolutely normal numbers, but insensitive enough to allow suitable computable sequences to pass them.

It is interesting to speculate as to why Turing did not develop the whole apparatus of algorithmic randomness since he seemed to have many of the ideas needed for this development. To the best of my understanding, whilst he did have the notion of a pseudo-random number generator, Turing himself thought that randomness was a physical phenomenon. Turing certainly recognized the noncomputable nature of generating random strings. For example, from Turing [44], we have the following quote.

> " An interesting variant on the idea of a digital computer is a "digital computer with a random element." These have instructions involving the

throwing of a die or some equivalent electronic process; one such instruction might for instance be, "Throw the die and put the-resulting number into store 1000." Sometimes such a machine is described as having free will (though I would not use this phrase myself)."

Interestingly, in the sentences after the quote he recognizes the difficulty of perception of randomness:

"It is not normally possible to determine from observing a machine whether it has a random element, for a similar effect can be produced by such devices as making choices depend on the digits of the decimal for π."

We know that Turing used algorithms to generate *pseudo-random* strings, strings that seemed sufficiently random that they worked the way we would expect a random source to behave. Perhaps Turing was only concerned with pseudo-random strings in relation to algorithms in relation to, for example, cryptography and artificial intelligence as outlined below. Certainly shortly after 1938, the war intervened and, after the war Turing did not return to the topic of normality. He did not mention randomness except in relation to efficiency and efficacy of f algorithms. I don't think he ever considered the problem of *defining an algorithmically random sequence.*

7 Turing and randomness as a resource

From my readings of Turing's works, it is clear that Turing regarded randomness as a computational resource. For example, in artificial intelligence Turing consider learning algorithms. Turing says in [44]

"It is probably wise to include a random element in a learning machine.... A random element is rather useful when searching for the solution of some problem."

Turing then gives an example of search for the solution to some numerical problem, pointing out that if we do this systematically, we will often have a lot of overhead corresponding to our previous searches. However, if the problem has solutions reasonably densely in the sample space random methods should succeed.

Form a modern perspective, we know of many algorithms which work well randomly. A simple illustration is what is called polynomial identity testing (PIT). PIT the following problem: Given a polynomial $P(X_1, \ldots, X_n)$, is that polynomial identically zero (i.e. results in zero no matter what numbers we substitute for the variables)? For example, $X_1X_2 - X_2X_1$ is identically zero. Now, this might seem a very strange and specialized problem, but it turns out that many problems can be computationally restated as instances of PIT, so that we

can solve them by solving an algorithm for PIT[10]. There are very efficient randomized algorithms for PIT. In fact, the dumbest algorithm one could imagine works. Take *any random numerical substitutions* for X_1, $X_2 \ldots X_n$, and evaluate the polynomial at those values. If you don't get zero, the answer is certainly "no". If you do evaluate to zero say "yes" . You will be correct with high probability, roughly speaking because such polynomials are zero quite rarely if they are not identically zero[11].

A major theme in modern algorithm design is to use randomness as a resource like this. It is fair to say that most computational learning algorithms, like those that monitor you on the Internet and try to see you things based on learning your preferences, or those that model physical phenomena, all use randomness as a resource. This use is anticipated by Turing in his writings. In [45], Turing even speculates that randomness is somehow necessary for intelligence. To my knowledge, Turing did not develop this theme mathematically.

This lack of mathematical development is perhaps unfortunate, as there is an implicit recognition of algorithm *complexity* in Turing's writings; such recognition coming from *cryptography* where large search spaces are used to hide information, and *practical computing* where an answer must be found in real time. Even Turing's writing about intelligence and the limitations of machines are full of calculations as to numbers of possible state in computers, etc. Perhaps Turing could have developed the mathematical theory of computational complexity (only really developed in the 1960's and 70's) had he developed his ideas further. But this is complete speculation. It is clear that he had an intuitive understanding of the time and space complexity of computation.

We remark that these complexity issues are very deep. It is a very longstanding open question whether PIT can be solved by a non-randomized deterministic algorithm in polynomial time. Although we have no idea how to construct such an algorithm, is thought that such an algorithm exists. This is because of the work of Impagliazzo and Wigderson [24] whose work proved a certain hardness/randomness tradeoff[12]. For a long time primality testing, determining whether a number is prime, had the same status. That is, primality testing has randomized algorithms which work very quickly, and known for many years. In a celebrated result first announced in 2002 (published in [1]) Agrawal, Kayal, and

[10] Strictly speaking this computational restatement usually happens for polynomials with coefficients not in the rationals, but over some large finite *field*, which is a special kind of mathematical structure whose arithmetic "resembles" the rationals.

[11] This is very easy to see for polynomials of one variable. A nonzero cubic polynomial $P(X) = aX^3 + bX^2 + cX + d$ can have at most 3 roots. Hence $P(X)$ can evaluate to zero in at most 3 places. It is highly unlikely a random choice would be unlucky enough to pick one of those 3 points! The case with many variables is proven by induction.

[12] They prove a truly remarkable result which says roughly that if certain problems are as hard as we think they are, then 'BPP=P", which means that all problems like PIT with efficient randomized polynomial time algorithms have polynomial time derandomized versions.

Saxena gave a deterministic (i.e. non-randomized) polynomial time algorithm for primality testing.

8 Where does this all go?

8.1 Normality

There's a lot of ongoing work on (absolute) normality, since it is a natural number-theoretical notion. In terms of its relationship with randomness, it is possible to show that normality really is a *precise calibration of randomness*. Normality corresponds to a kind of randomness property called *finite state Hausdorff dimension*, where we look at the behaviour of betting strategies (martingales) which are controlled by *finite state gamblers*[13]. Schnorr and Stimm [39] proved that a sequence X to base b is normal to base b if and only if the finite state Hausdorff dimension of X is 1, which is a statement about the Kolmogorov complexity[14] of initial segments of X. Mayordomo and Lutz (in preparation, the theorem first posted in [34]), and independently, Becher, Heiber and Slaman [6] have shown that it is possible to construct such sequences which are quite simple, we can compute the n-th bit of X in polynomial time, somewhere near $O(n^2)$.

8.2 Randomness

There has been a huge amount of work on algorithmic randomness and its applications to, for example, algorithm design, understanding analysis in mathematics, Brownian motion, ergodic theory, Markov chain theory, physics, etc. (e.g. [7, 12, 13, 21, 16, 22, 23]) I won't even try to present any of this here. For general introductions to the area of randomness and its relation to computability, I refer to Downey and Hirschfeldt [20], Nies [32] and for a general reference to Kolmogorov complexity you can look at Li-Vitanyi [31]. For a more informal discussion as to the general theory of randomness, I refer to Zenil [51], which has essays by leading experts in the area. For more on the mathematical, and particularly, logical, developments stemming from Turing's work, I refer to the forthcoming volume [19].

---

[13] If the reader is unfamiliar with this notion, widely used in computer science, they should think of them as memoryless machines which plod from one internal state to the next, processing the bits of the input, purely on the basis of the bit that is being read and the internal state. The are the most basic of all machines.

[14] Specifically, if KF denoted finite state Kolmogorov complexity, then we would ask that for any finite state machine, $\liminf_{n\to\infty} \frac{KF(X\upharpoonright n)}{n} = 1$. We refer the reader to Dai et al. [18], Bourke et. al. [10], and to Jack Lutz's home page for much more on this topic, and its relationship to things like DNA self-assembly, modeling the processes of molecular biology.

References

1. M. Agrawal, N. Kayal, N. Saxena. PRIMES is in P *Annals of Math.* Volume 160 (2004), 781-793.
2. D. Bailey and R. Crandall. On the Random Character of Fundamental Constant Expansions. *Experimental Mathematics,* Vol. 10 (2001), No. 2
3. Veronica Becher. Turing's normal numbers: towards randomness. In S.B. Cooper, A.Dawar, B. Löwe (eds.), CiE 2012, Lecture Notes in Computer Science 7318: 35-45 Springer, Heidelberg, 2012.
4. V. Becher, S. Figueira, R. Picchi. Turing's unpublished algorithm for normal numbers. *Theoretical Computer Science* 377 (2007), 126–138.
5. V. Becher and S. Figueira. An example of a computable absolutely normal number. *Theoretical Computer Science* 270 (2002), 947–958.
6. V. Becher, P. Heiber, and T. Slaman. A polynomial time algorithm for computing absolutely normal numbers. *Proceedings Amer. Math. Soc.*, in press.
7. L. Bienvenu, A. Day, M. Hoyrup, I. Mezhirov, and A. Shen. Ergodic-type characterizations of algorithmic randomness. To appear in *Information and Computation.*
8. E. Borel. Les probabilités dénombrables et leurs applications arithmétiques. *Rendiconti del Circolo Matematico di Palermo* 27 (1909), 247–271.
9. E. Borel. *Leçons sur la thèorie des fonctions.* Gauthier Villars, 2nd ed. 1914.
10. C. Bourke, J. Hitchcock, N. Vinodchandran, Entropy rates and finite-state dimension. *Theoretical Computer Science* 349(3) (2005), 392–406.
11. Y. Bugeaud, Nombres de Liouville et nombres normaux, *Comptes Rendus de l'Académie des Sciences de Paris* 335 (2002), 117–120.
12. V. Brattka, J. Miller, and A. Nies, *Randomness and differentiability*, to appear.
13. M. Braverman and M. Yampolsky, Non-Computable Julia Sets. *Journ. Amer. Math. Soc.*, Vol. 19(3), 2006
14. G. Chaitin. A theory of program size formally identical to information theory. *Journal of the ACM,* Vol. 22 (1975), 329–340.
15. A. Church. On the concept of a random sequence. *Bulletin of the American Mathematical Society,* Vol. 46 (1940), 130–135.
16. R. Cilibrasi, P.M.B. Vitanyi, and R. de Wolf. Algorithmic clustering of music based on string compression. *Computer Music J.,* Vol. 28 (2004), 49-67.
17. A. Copeland and P. Erdös. Note on normal numbers. *Bull. Amer. Math. Soc.* Vol. 52 (10) (1946), 857860.
18. L. Dai, J. Lutz, E. Mayordomo. Finite-state dimension. *Theoretical Computer Science* 310 (2004), 1–33.
19. R. Downey (editor). *Turing's Legacy.* To appear, Cambridge University Press.
20. R. Downey and D. Hirschfeldt, *Algorithmic Randomness and Complexity*, Springer-Verlag, 2010.
21. W. Fouche. The descriptive complexity of Brownian motion. *Advances in Mathematics*, Vol. 155 (2000), 317–343.
22. H. Fuchs and C. Schnorr. Monte Carlo methods and patternless sequences. In *Operations Research Verfahren*, Vol XXV, Symp. Heidelberg, 1977, 443-450.
23. M. Hochman and T. Meyerovitch. A characterization of the entropies of multi-dimensional shifts of finite type. *Annals of Mathematics*, Vol. 171 (2010), No. 3, 2011-2038
24. R. Impagliazzo and A. Wigderson. P = BPP if E requires exponential circuits: derandomizing the XOR lemma. In *Proceeding STOC '97 Proceedings of the twenty-ninth annual ACM symposium on Theory of computing* 220-229.

25. A. N. Kolmogorov. *Grundbegriffe der Wahrscheinlichkeitrechnung, Ergebnisse Der Mathematik.* (1933) translated as *Foundations of Probability*, New York: Chelsea Publishing Company, 1950.
26. A. N. Kolmogorov, *Three approaches to the quantitative definition of information*, Problems of Information Transmission, 1 (1965), 1–7.
27. H. Lebesgue. Sur certaines démonstrations d'existence. *Bulletin de la Société Mathématique de France* 45 (1917), 132–144.
28. L. Levin. *Some theorems on the algorithmic approach to probability theory and information theory*, Dissertation in Mathematics Moscow University, 1971.
29. L. Levin. Laws of information conservation (non-growth) and aspects of the foundation of probability theory. *Problems of Information Transmission,* Vol. 10 (1974), 206–210.
30. M. Levin. On absolutely normal numbers. English translation in *Moscow University Mathematics Bulletin* 34:32–39.
31. M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*, 2nd edition, Springer-Verlag, 1997.
32. A. Nies, *Computability and Randomness,* Oxford University Press, 2009.
33. P. Martin-Löf, *The definition of random sequences*, Information and Control, 9 (1966) 602–619.
34. E. Mayordomo. Construction of an absolutely normal real number in polynomial time. Preprint, November 2012.
35. C. Schnorr. A unified approach to the definition of a random sequence. *Mathematical Systems Theory,* Vol. 5 (1971), 246–258.
36. C. Schnorr, *Zufälligkeit und Wahrscheinlichkeit. Eine algorithmische Begründung der Wahrscheinlichkeitstheorie*, volume 218 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin–New York, 1971.
37. C. Schnorr and H. Stimm. Endliche Automaten und Zufallsfolgen. *Acta Informatica* 1 (1972), 345–359.
38. W. Schmidt. On normal numbers. *Pacific Journal of Math.* 10 (1960), 61–672.
39. W. Sierpiński. Démonstration élémentaire du théorème de M. Borel sur les nombres absolument normaux et détermination effective d'un tel nombre. *Bulletin de la Société Mathématique de France* 45(1917), 127–132.
40. R. Solomonoff. A Formal Theory of Inductive Inference, Part I. *Information and Control,* Vol 7, No. 1 (1964), 1-22.
41. A. Turing. On computable numbers with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society,* Vol. 42 (1936), 230–265, 1936. Correction in *Proceedings of the London Mathematical Society,* Vol. 43 (1937), 544–546.
42. A. Turing. Computing machinery and intelligence. *Mind,* Vol. 59 (1950), 433-460.
43. A. Turing. Intelligent Machinery, A Heretical Theory 1951, but first published in *The Essential Turing*, J. Copeland ed., Claredon Press, Oxford, 2004.
44. A. Turing. A note on normal numbers. In J.L.Britton, editor *Collected Works of A.M. Turing: Pure Mathematics.* North Holland, Amsterdam, 1992, 117–119, with notes of the editor in 263–265. Reprinted in *Alan Turing - his work and impact*, S B. Cooper and J. van Leeuwen editors, Elsevier, 2012.
45. J. Ville, *Étude Critique de la Notion de Collectif*, Gauthier-Villars, 1939.
46. R. von Mises. Grundlagen der Wahrscheinlichkeitsrechnung. *Math. Z.*, Vol. 5 (1919), 52–99.
47. A. Wald. Sur le notion de collectif dans la calcul des probabilitiés. *Comptes Rendes des Seances de l'Académie des Sciences,* Vol. 202 (1936), 1080–1083.

48. A. Wald. Die Weiderspruchsfreiheit des Kollektivbegriffes der Wahrscheinlichkeitsrechnung. *Ergebnisse eines mathematischen Kolloquiums,* 8 (1937), 38–72.
49. H. Zenil, *Randomness Through Computation: Some Answers, More Questions,* (Hector Zenil editor), World Scientific, Singapore, 2011.