A Measure of Randomness

Gary Geng

May 2020

Contents

1 Introduction

We will follow A.N. Kolmogorov's paper *On Tables of Random Numbers* [4], first published in English in 1963. In this paper, Kolmogorov discusses a test of randomness for binary strings of finite length. This had come after previously unsuccessful attempts at characterizing the randomness of infinite binary strings, and would prove to be a step in the right direction for the development of the theory of randomness in computability theory.

Previously, in 1919, Richard von Mises [7] had attempted to test for the randomness of an infinite sequence using frequency analysis. However, without the tools of computability theory, von Mises' approach was imprecise. Church [1] later attempted to add to von Mises' results to resolve some ambiguities, but as we will see, their approaches would prove to be flawed. Building on these ideas, the notion of Kolmogorov complexity would be crucial for later developments of tests for the randomness of infinite binary sequences. The results of the paper we are examining could be considered as a precursor the notion of Kolmogorov complexity.

Before we begin, we will clarify the idea of randomness. There are three main approaches to classifying the randomness of a binary sequence. Here we list these in the words of Downey and Hirschfeldt [3]:

1. *Statistician's approach:* random sequences should not have statistically rare properties. In the language of measure theory, the set of sequences satisfying any sort of statistical regularity should belong to an effectively null set.

2. *Gambler's approach:* random sequences should be unpredictable. Namely, a gambler betting double-or-nothing on successive bits of a sequence should not be able to get unboundedly rich.

3. *Coder's approach:* we should not be able to compress random sequences while retaining all its information.

We will be examining the last approach in this paper. We will also see that von Mises attempts to something alike the *statistician's approach*, but as he lacked the tool of computability theory, his attempt would fall short. The central importance of computability theory here can be intuitively understood with simple examples. In one case, if one flips a coin 10,000 times and it landed heads each time, one would certainly not consider that random. However, one could also argue that such is one of the 2^{10000} variations possible, and in any case one of these variations should occur. In another case, suppose as in the gambler's approach, someone is betting on the outcomes of 10,000 coin flips. If he knew these outcomes in advance, then clearly he would bet exactly on the flips which he knows to be heads. But this defeats the point, as the gambler, in essence, *cannot* know of the outcome of a flip before he placed the bet. The ideas in these two cases can be summarized as, *one can only determine the randomness of a sequence as it unfolds, but not after he learns it.* Then, it is important to define what is *computable* with the existing information that we have, as it is not mathematically rigorous enough to merely state that "a gambler is not able to consistently profit off of this sequence, and therefore it is random".

2 Background

We give a very brief overview of some basic concepts in computability theory. We owe some background and notations to the book by Weber [**?**].

The motivation for computability theory is a good mathematical definition for a procedure, or function, that can be computed by a program. For this, we only consider functions that take on natural number values. We first explore the concept of Turing machines.

2.1 Turing machines

Definition 2.1 (Turing machine)**.** A *Turing machine* (TM), as formulated by Alan Turing, is the concept of a computer with an infinitely long tape. The tape is partitioned horizontally into squares, in which are symbols. The possible symbols are 0, 1, and blank. A Turing machine has a pointer called the **head**, which starts off pointing at a square on the tape. The tape could be understood as the input of the TM, and the head as first located at the initial read location of the input.

TM may be specified using sequences of quadruples, known as *instructions*. An instruction tuple

$$\langle a, b, c, d \rangle$$

is decoded like so:

- a is the current state of the TM
- b is the symbol that the head of the TM currently reads
- c is an instruction for the head to write, or to move (left/right)
- d is the state of the TM *after* this instruction is executed

At any given step, an instruction is executed if a matches the current state, and b matches the current symbol read. The behavior of the instruction depends on c and d.

We will give a brief example on the operations of a TM. Suppose we are given a tape

...	B	B	B	**1**	0	0	1	1	1	0	B	B	...

Blanks are denoted by "B". The head is initially at the bolded "1", or the first square that is not a blank. Usually, blanks are used to specify the start and end of an input.

We will write a Turing machine that overwrites all the 0's to 1's. Here are its instructions:

$\langle q_0, 1, R, q_0 \rangle$

$\langle q_0, 0, 1, q_0 \rangle$

$\langle q_0, B, B, q_1 \rangle$

We have initial state q_0. Since we are reading 1, the first instruction executes: R indicates that we move right, and the last tuple entry "q_0" indicates that we stay in state q_0. Now we are reading a 0. Then the second instruction executes: 1 indicates that we write "1" (overwriting 0), and we again stay in q_0. This would be repeated until we reach the right end of the tape. At that point we

would read B, and the third instruction is executed: we write B (do nothing), and enter q_1. After this, since there is no state q_1, the TM terminates.

This is a very simple example with only two states, but one can see how a Turing machine may be able to perform complex procedures using many states, similar to a computer executing a program.

2.2 Church-Turing thesis

The **Church-Turing thesis** is a hypothesis about the class of computable functions, or roughly functions that could be computed by a human following some procedure. The Church-Turing thesis claims that *a function is computable if and only if it can be computed by a Turing machine.* This may seem like a very general notion, and indeed there is no proof for this hypothesis. But nevertheless this is what most people mean now when they say a function is "computable".

2.3 Coding

Coding is an important technique in computability theory. It allows us to encode certain sets of objects to the set of natural numbers, the domain of computable functions. Additionally it allows us to have an enumeration of all Turing machines.

For a 2-tuple $\langle x, y\rangle$, we have the following **Cantor pairing function**

$$\pi(\langle x, y\rangle) : \mathbb{N}^2 \to \mathbb{N} = \frac{1}{2}(x + y)(x + y + 1) + y$$

which is a bijection from $\mathbb{N}^2$ to $\mathbb{N}$. Note that from this we may encode tuples of any length inductively, by way of

$$\langle x, y, z\rangle := \pi(\langle \pi(\langle x, y\rangle), z\rangle)$$

Therefore we can allow for arbitrarily many arguments for our computable functions by encoding tuples into natural numbers.

A very important consequence here is that we may map the Turing machines to $\mathbb{N}$, by first encoding their instruction tuples and then encoding the tuple of its instructions. The code of a Turing machine is known as its **index**, and a TM with index j is called the jth Turing machine.

2.4 Universal Turing Machine

Denote by φ_j the jth Turing machine.

Definition 2.2 (Universal Turing machine)**.** A Turing machine U is said to be a *universal Turing machine* if we have

$$U(\langle j, x\rangle) = \varphi_j(x)$$

This machine is capable of emulating all other Turing machines. We know that U exists because, theoretically, we may write out a set of instructions for U to unwrap the input into instructions (the reverse bijection from $\mathbb{N}$ to the set of sequences of instructions), and then execute these instructions on the input. That is, we can set out a blueprint that works *uniformly* for all input. Turing gives a detailed description of that universal TM [**?**].

The enumeration of Turing machines and the notion of the universal Turing machine will be required in later sections.

3 Mises-Church Randomness

Richard von Mises [7] was one of the first to attempt to give a rigorous definition of randomness. Mises observed that a random sequence should follow the law of large numbers, which essentially states that the occurrences of an event as a fraction of n, the total number of trials, should approach its expected value as $n \to \infty$.

The idea here is that, as in flipping a coin, one has equal chances of getting heads and getting tails. So by the law of large numbers, the frequency of heads (or tails) should approach $1/2$.

Let $\sigma[j]$ denote the jth digit of a binary sequence σ. A test based on the above intuitions may be devised, namely: a sequence $\sigma \in 2^{<\omega}$ is random if

$$\limsup_{n\to\infty} \frac{|\{m < n : \sigma[m] = 1\}|}{n} = \frac{1}{2} \tag{1}$$

holds.

This would not work, however. For example, it would accept the obviously nonrandom sequence $101010\ldots$ Von Mises then devised the notion of a *selection function*. A selection function f is a strictly increasing function that maps $\mathbb{N}$ to a set of integer indices. We then have the below definition.

Definition 3.1 (von Mises)**.** An infinite binary sequence σ is Mises-random if for *every* selection function f, the following holds

$$\limsup_{n\to\infty} \frac{|\{m < n : \sigma[f(m)] = 0\}|}{n} = \frac{1}{2}$$

This would imply that every subsequence of σ satisfies the law of large numbers.

There is yet another problem with the above definition. Consider the selection function that outputs the subsequence of σ consisting only of 1's. Then the test would fail for all σ. Then notion of a "computable" function is required, i.e. a function that, when deciding whether to select element $\sigma[n]$, has only information about the elements $\sigma[1] \dots \sigma[n-1]$.

Von Mises attempted to precisely define such a function, but he lacked the tools of computability theory. Later, Alonzo Church would present a more rigorous definition [1]:

Definition 3.2. (Church) An infinite binary sequence σ is Mises-Church random if the following two conditions hold:

1. The equation (1), proposed by von Mises, is satisfied.
2. Let f be any computable function, and let $\sigma \upharpoonright m$ denote the subsequence obtained by taking the first m elements of σ. If $f(x) = 1$ for infinitely many input x, then

$$\limsup_{n\to\infty} \frac{|\{m < n : f(\sigma \upharpoonright (m-1)) = 1, \sigma[m] = 1\}|}{n} = \frac{1}{2}$$

There are a few things worth noting of this definition. First is the role of f. It is useful now to think of the selection function f as a function that accepts the first $m-1$ elements of σ, and then decides if it wants to include mth element in the test subsequence. Here we make the arbitrary choice to include an element if f outputs 1, and exclude it otherwise. We only want infinite subsequences of σ, so we restrict to functions that output 1 for infinitely many inputs (clearly a function that never outputs 1 is useless to us).

The other thing worth noting is the coding of $\sigma \upharpoonright m$ to a natural number. One may consider directly interpreting it as a binary number. However, in order to further distinguish between sequences like 0 and 00, one needs to add an additional length argument. Church defines a more elegant coding:

$$b_1 = 1,\ b_{n+1} = 2b_n + \sigma[n]$$

so that $\sigma \upharpoonright m$ may be coded to b_{m+1}. Regardless, see that for a Turing machine, one can construct a tape of 0's and 1's and insert blanks to mark its end.

Church thus made von Mises' notion of randomness precise. However, in 1939, Jean Ville had already proposed a counterexample to the von Mises test for randomness!

Recall that $\sigma \upharpoonright m$ denotes the subsequence containing the first m elements of σ. Ville shows the following theorem.

Theorem 1 (Ville (1939))**.** *For any selection function, there is a sequence σ that passes the von Mise test for randomness using that selection function, and that $\sigma \upharpoonright m$ contains more 1's than 0's for every m.*

Define *prefix* of a sequence σ to be a subsequence $\sigma \upharpoonright m$ for any m. Clearly if every prefix of σ contains more 1's than 0's, σ should not be considered random.

Ville himself proposed tests that could handle his specific example. But the question arises: what if there are still other counterexamples that would bypass the existing test? How can we be sure that our test is impervious to such counterexamples of irregularity? Then it is necessary to consider a (possibly infinitely enumerable) set of selection functions. This leads us to Kolmogorov's paper.

4 Kolmogorov's Randomness Measure

4.1 Motivation and Background

Kolmogorov begins the paper by expressing his previous view that frequency analysis, as championed by von Mises, is problematic in the following ways: 1) frequency analysis of an infinite sequence does not contribute to real-world applications, where the trials are always infinite; and 2) frequency analysis of a finite sequence is not rigorous enough to be used "within the framework of formal mathematics".

Indeed, a frequency analysis of the finite decimal digits $\pi = 3.1415926\ldots$ would fail to find a pattern within them. However we know, intuitively, that π contains relatively little information, since one can easily derive it to arbitrary precision using a short computer program. We also recall that previously, Ville was able to give a counterexample to the formulations of Church-Mises randomness, which was based on the frequency analysis of infinite sequences.

However, Kolmogorov claims that he had come to change his latter view of frequency analysis on finite sequences. To show that in fact a rigorous characterization of the randomness in a finite sequence is possible, Kolmogorov would make use of the fact that "there cannot be a *very large number of simple algorithms*". We shall see the meaning of this soon.

4.2 Admissible Selection Algorithms

Suppose we are now given a large, finite binary sequence $\sigma \in 2^{<\omega}$ (Kolmogorov refers to them as "tables"). It is clearly not enough to examine the frequency of digits in the entire sequence alone. As in the last section, we seek to extract subsequences of σ and apply tests to them as well. But as we have seen before, the step of selecting these subsequences can prove to be problematic. Kolmogorov gives a definition of the set of what he calls to be "admissible selection algorithms".

Let $N = |\sigma|$, namely the length of σ. Consider a sequence of three partial

computable functions, F_j, G_j, H_j, as below. Note that F_0, G_0, H_0 take no arguments, so we may consider them as (possibly undefined) constants.

F_0	G_0	H_0
$F_1(\chi_1, \tau_1)$	$G_1(\chi_1, \tau_1)$	$H_1(\chi_1, \tau_1)$
$F_2(\chi_1, \tau_1; \chi_2, \tau_2)$	$G_2(\chi_1, \tau_1; \chi_2, \tau_2)$	$H_2(\chi_1, \tau_1; \chi_2, \tau_2)$
$\dots$	$\dots$	$\dots$
$F_{N-1}(\chi_1, \tau_1; \dots ; \chi_{N-1}, \tau_{N-1})$	$G_{N-1}(\chi_1, \tau_1; \dots ; \chi_{N-1}, \tau_{N-1})$	$H_{N-1}(\chi_1, \tau_1; \dots ; \chi_{N-1}, \tau_{N-1})$

The argument χ_j stores the position, or *index* of the jth element of σ previously viewed, and so it takes on integer values in $[1, N]$.

The argument τ_j stores whether the element denoted by χ_j is selected. If that element is selected, $\tau_j = 1$; otherwise $\tau = 0$.

Below are the roles of each of the three functions:

1. F_j is the *explorer function*. Given the sequence of previous examined and selected elements, it decides which index it would like to view next. And thus, it takes on an integer value in $[1, N]$. We additional impose on F_j the natural limitation that $F_k(\chi_1, \tau_1; \dots ; \chi_k, \tau_k) \neq \chi_i$ for all $i = 1 \dots k$, to prevent viewing the same element twice. Worth noting that unlike Church's selection function which is increasing, this function allows us to jump back and forth in the sequence.

2. G_j is the *selector function*. Again, given the current sequence $1 \dots j$, it decides if it would like to select the $j+1$th element, which F_j selects. Crucially, G_j does not know the output of F_j, and vice versa. It outputs 1 or 0 to indicate selection or discard, i.e. we have $\tau_{j+1} = G_j(\chi_1, \tau_1; \dots ; \chi_j, \tau_j)$

3. H_j is the *terminator function*. Given the current sequence, it decides if it would like to terminate the procedure, i.e. stop selecting by outputting 1 (for termination) or 0 (keep going). If H_j does not equal 1 for $j = 0 \dots N$, the procedure automatically terminates after exhausting the entire sequence.

As Kolmogorov shows, to select a subsequence we would form the following sequence

$$
\begin{aligned}
x_1 &= F_0 \\
x_2 &= F_1(x_1, t_{x_1}) \\
x_3 &= F_2(x_1, t_{x_1}; x_2, t_{x_2}) \\
\dots &= \dots \\
x_s &= F_{s-1}(x_1, t_{x_1}; \dots ; x_{s-1}, t_{s-1})
\end{aligned}
$$

where as before, the x_j's are the indices, and t_{x_j} indicates whether x_j is selected, i.e. $t_{x_j} = G_{j-1}(x_1, t_{x_1}; \ldots; x_{j-1}, t_{j-1})$. s is the smallest j for which $H_j(\ldots) = 1$ (or N otherwise).

So now we have a well-defined and valid selection procedure. Let us denote by R a function that encapsulates the sequences of functions F_j, G_j, and H_j. Namely, R is a function that accepts a finite binary string σ and outputs a subsequence of σ using the procedures described above. Let such an R be defined as a **selection function**.

We will now consider how a set of such selection functions can be used to provide a measure of randomness.

(n, ϵ)-Randomness

Fix N as the length of the input sequence σ, and let $\mathscr{R}_N$ be a set of selection functions. Kolmogorov refers to such a set as a **system**.

Definition 4.1 (Kolmogorov)**.** A finite binary sequence σ is called (n, ϵ)-random with respect to the system $\mathscr{R}_N$, if there exists a constant p, such that for any subsequence

$$\alpha = R(\sigma), R \in \mathscr{R}_N$$

with length

$$|\alpha| \geq n$$

and with *frequency* π defined in the conventional sense as

$$\pi(\alpha) = \frac{1}{|\alpha|} \sum_{k \in \alpha} \alpha[k]$$

the below inequality is satisfied

$$|\pi(\alpha) - p| \leq \epsilon$$

We will from now on refer to this as (n, ϵ, p)-randomness, with the understanding that p is a fixed probability.

From a statistical standpoint, one may understand n as the sample size, and ϵ as the level of significance of the test, with the result being more statistically significant smaller ϵ.

And now we have the central theorem of the paper:

Theorem 2. *If the number of elements of the system $\mathscr{R}_N$, denoted by $|\mathscr{R}_N|$ does not exceed the value*

$$\frac{1}{2} e^{2n\epsilon^2(1-\epsilon)}$$

then for any p, $0 \leq p \leq 1$, there exists a sequence σ of length N that is (n, ϵ, p)-random with respect to $\mathscr{R}_N$.

By taking the binary logarithm of the estimate, the theorem becomes more intuitive. Define

$$\lambda(\mathscr{R}_N) = \log_2 |\mathscr{R}_N|$$

The the condition of the theorem becomes

$$\lambda(\mathscr{R}_N) \leq 2(\log_2 e)n\epsilon^2(1-\epsilon) - 1$$

Now we are able interpret Kolmogorov's earlier statement on the simplicity of algorithms. Kolmogorov himself describes the quantity $\lambda(\mathscr{R}_N)$ as "the quantity of information, which is necessary for choosing an individual element R from $\mathscr{R}_N$". To understand this, recall our previous notion of the enumeration of computable functions (Turing machines). Namely, there is a bijection of the set of computable functions to the natural numbers. Then, we may encode any computable function φ_j to its index j, regardless of how complicated the function is itself. What Kolmogorov is indicating, then, is that for a large system of functions $\mathscr{R}_N$, encoding any member R of that system requires more "information", or bits. Roughly, the higher number of bits required to encode, the more complicated, and less "simple", a function is. The number of bits needed for encoding, or determining, any such R in $\mathscr{R}_N$ would be $\lambda(\mathscr{R}_N)$. Thus, given n, ϵ, and arbitrary p, in order to find a sequence σ that is (n, ϵ, p)-random with respect to a system $\mathscr{R}_N$, we need to place an upper bound on the quantity $\lambda(\mathscr{R}_N)$. With such an upper bound, we are essentially making the set of selection functions simpler.

Hence, Kolmogorov points out that by Theorem 2, for *any* system $\mathscr{R}_N$, as long as $\lambda(\mathscr{R}_N)/n$ is sufficiently small, there are guaranteed to be sequences that are (n, ϵ, p)-random with respect to $\mathscr{R}_N$. The implications of this result, and its relation to the Kolmogorov complexity will be discussed in section 5.

4.3 Proof of Theorem 2

The prove theorem 2, Kolmogorov claims that the following theorem on the Bernoulli scheme can be proven:

Theorem 3. *Consider any infinite sequence of independent Bernoulli trials, each with probability of success p. Denote the number of successes in the first k trials by random variable μ_k. Then for any positive n, ϵ we have*

$$P\left(\sup_{k \geq n} \left| \frac{\mu_k}{k} - p \right| \geq \epsilon \right) \leq 2e^{-2n\epsilon^2(1-\epsilon)} \tag{2}$$

From this theorem we easily obtain the following corollary

Corollary 3.1. *Let the conditional probability*

$$P(\chi_k = 1 \,|\, k \leq v, \chi_1 = x_1, \ldots, \chi_{k-1} = x_{k-1}) = p$$

where $\chi_1, \chi_2, \ldots, \chi_v$ is a sequence of a random number of random variables, and $x_1, x_2, \ldots, x_{-1}$ are the current quantities assigned to $\chi_1, \chi_2, \ldots, \chi_{k-1}$. Then

$$P\left(v \geq n, \left|\frac{\mu_v}{v} - p\right| \geq \epsilon\right) < 2e^{-2n\epsilon^2(1-\epsilon)}$$

This is true because we have

$$P\left(v \geq n, \left|\frac{\mu_v}{v} - p\right| \geq \epsilon\right) \leq P\left(\sup_{k \geq n} \left|\frac{\mu_k}{k} - p\right| \geq \epsilon\right)$$

and the rest follows directly from (2). □

Now consider a system $\mathscr{R}_N$, and define $\rho = |\mathscr{R}_N|$. Suppose we have $\sigma \in 2^{<\omega}$ with the probability for $\sigma[j] = 1$ be p independently of the other elements. Then if we take any $R \in \mathscr{R}_N$, and let $s = R(\sigma)$ be the subsequence extracted by R. Then the probability that $|s| \geq n$ and satisfying the inequality $|\pi(\sigma) - p| \geq \epsilon$ is at most $2e^{-2n\epsilon^2(1-\epsilon)}$. This comes directly from the corollary.

Now then if $\rho = |\mathscr{R}_N| \leq \frac{1}{2}e^{2n\epsilon^2(1-\epsilon)}$ (the hypothesis of theorem 2), we have an estimate of the probability that

$$|\pi(s) - p| \leq \epsilon \tag{3}$$

holds for all $s = R(\sigma), R \in \mathscr{R}_N$. This is the complement of the probability that (3) fails for at least one R. If we sum these probabilities for all R such that $|R(\sigma)| \geq n$, we see that the sum is less than $\rho \cdot 2e^{-2n\epsilon^2(1-\epsilon)} \leq 1$. Thus the probability for a sequence σ to be (n, ϵ, p)-random with respect to $\mathscr{R}_N$ is greater than zero, and there must exist such a desired sequence. Notably, this is true independent of p.

4.4 Some Details

Kolmogorov concludes the paper by exploring some possibilities of sharpening the hypothesis in theorem 2. For this, some cases of the asymptotic behaviors of n and ϵ are considered, but no conclusive results are reached.

The reader may also note that we have not given a proof for theorem 3. Kolmogorov wrote that he would prove it "in another paper", but in fact he never did. Much later in 1993, Turner, Young, and Seaman [8] would finally provide a proof. The proof is not long, but it requires a good amount of statistical background, so we will not go over it in this paper.

5 Further Remarks

The idea of (n, ϵ, p)-randomness would evolve and be transformed into a more elegant and concise test in Kolmogorov's seminal paper *Three Approaches to*

the Quantitative Definition of Information [5]. In it, he would introduce the concept of Kolmogorov complexity, intuitively understood as follows:

For any binary sequence σ and Turing machine M, let the τ be the shortest input such that $M(\tau) = \sigma$. The Kolmogorov complexity $C_M(\sigma)$ is then the length of τ.

But now note that there is a universal Turing machine U that emulates all other machines. Then we can pass to U a concatenation $j\sigma$ such that j is the index of the Turing machine to emulate, and σ is the input string. Then suppose for M that reproduces σ with shortest input τ, U may reproduce σ with input length $|\tau| + \rho_M$. We call $C(\sigma) = C_U(\sigma)$ the (plain) Kolmogorov complexity of σ, and it is only ρ_M larger than $C_M(\sigma)$ for any M. The idea then, is that σ is random if $C(\sigma)$ is not much shorter than the length of σ. This is the "compression" intuition we described in the *coder's approach* in the introduction.

One can very clearly see the similarities in ideas between (n, ϵ, p)-randomness and Kolmogorov complexity. In both cases, the complexity the algorithm (or length, the number of bits in the description) is the part of the consideration for the randomness of a sequence σ. Suppose that we want to maliciously construct a function M so that σ fails to be (n, ϵ, p)-random relative to it. Since we do not know σ, a sure way to do it is to enumerate over possible "regularities" in σ. The large $|\mathscr{R}_N|$ is, the more such regularities may be exploited. For σ with low Kolmogorov complexity, such as $1010101010\ldots$, a very short function can be constructed to exploit the regularity of the even/odd indices, by selecting a subsequence with even indices. Therefore only for *very* simple Turing machines (i.e. very small $|\mathscr{R}_N|$) can σ possibly pass the tests. Then σ is not considered random in either definitions.

As Li [6] puts it in his book, Kolmogorov's approach to defining randomness, that is, to show that no algorithm that computes a sequence can be much shorter than the sequence itself, is a definite shift from von Mises' approach to attempt to predict the next element of the sequence. This shift toward characterizing the system with which we test the sequence would yield fruitful results.

References

[1] A. Church. On the Concept of a Random Sequence. *Bull. Amer. Math. Soc.* **46** (1940), 130-135

[2] R.G. Downey and D. Hirschfeldt. *Algorithmic Randomness and Complexity.* Springer Science+Business Media, 2010

[3] R.G. Downey and D. Hirschfeldt. Computability and Randomness. *Notices Amer. Math. Soc.* **66** (2019), 1001-1011

[4] A. N. Kolmogorov. On Tables of Random Numbers. *Theoretical Computer Science*, **207** (1998) 387-395

[5] A.N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems Inform. Transmission*, **1** (1965), 1-7.

[6] M. Li, P. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*, 3rd Ed. Texts in Computer Science, Springer, New York, 2008.

[7] R. von Mises. Grundlagen der Wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift*, **5** (1919) 52–99

[8] D. W. Turner, D. M. Young, J. W. Seaman Jr. A Kolmogorov inequality for the sum of independent Bernoulli random variables with unequal means. *Statistics & Probability Letters*, **23** (1995), 243-245