University of Zurich[UZH]

Department of Psychology

Approximation of Kolmogorov Complexity for Short Strings via Algorithmic Probability

An Objective Measure of Randomness and Complexity – Introducing the R package `acss`

Henrik Singmann
Nicolas Gauvrit
Fernando Soler-Toscano
Hector Zenil

Introductory Example

- Two sequences of coin flips:
  1. `HTHTHTHT`
  2. `HTHHTHTT`

- What do we know about their difference?
  - P. believe **2** more likely than **1** (e.g., Kahneman & Tversky, 1972).
  - Actual probability of occurence is identical = $(\frac{1}{2})^8$
  - Shannon entropy (i.e., $-\sum p_i \log_2(p_i)$) is identical = 1

- Can we formalize intuition that **1** is more regular than **2**?

Foundational Notion

A string is random if it is hard to describe.
A string is not random if it is easy to describe.

Introductory Example

- Two sequences of coin flips:
  1. HTHTHTHT
  2. HTHHTHTT

- What do we know about their difference?
  - P. believe **2** more likely than **1** (e.g., Kahneman & Tversky, 1972).
  - Actual probability of occurence is identical = $(1/2)^8$
  - Shannon entropy (i.e., $-\sum p_i \log_2(p_i)$) is identical = 1

- Can we formalize intuition that **1** is more regular than **2**?

Foundational Notion

A string is random if it is hard to describe.
A string is not random if it is easy to describe.

Introductory Example

- Two sequences of coin flips:
  1. HTHTHTHT
  2. HTHHTHTT

- What do we know about their difference?
  - P. believe **2** more likely than **1** (e.g., Kahneman & Tversky, 1972).
  - Actual probability of occurence is identical = $(1/2)^8$
  - Shannon entropy (i.e., $-\sum p_i \log_2(p_i)$) is identical = 1

- Can we formalize intuition that **1** is more regular than **2**?

Foundational Notion

A string is random if it is hard to describe.
A string is not random if it is easy to describe.

Introductory Example

- Two sequences of coin flips:
  1. `HTHTHTHT`
  2. `HTHHTHTT`

- What do we know about their difference?
  - P. believe **2** more likely than **1** (e.g., Kahneman & Tversky, 1972).
  - Actual probability of occurence is identical = $(½)^8$
  - Shannon entropy (i.e., $-\sum p_i \log_2(p_i)$) is identical = 1

- Can we formalize intuition that **1** is more regular than **2**?

Foundational Notion

A string is random if it is hard to describe.
A string is not random if it is easy to describe.

Algorithmic Complexity

Definition

[Kolmogorov(1965), Chaitin(1966)]

$$K_U(s) = \min\{|p| : U(p) = s\}$$

- Algorithmic complexity $K_U(s)$ of a string s is length of shortest program p that produces s running on a universal Turing machine U.
- Two problems:
  1. $K_U(s)$ is *uncomputable*, but can be approximated from above (i.e., *upper semi-computable*).
  2. $K_U(s)$ depends on choice of Turing machine U.
- Small impact of U for long strings, but strong impact for short strings.

Algorithmic Complexity

Definition

[Kolmogorov(1965), Chaitin(1966)]

$$K_U(s) = \min\{|p| : U(p) = s\}$$

- Algorithmic complexity $K_U(s)$ of a string s is length of shortest program p that produces s running on a universal Turing machine U.

- Two problems:
  1. $K_U(s)$ is *uncomputable*, but can be approximated from above (i.e., *upper semi-computable*).
  2. $K_U(s)$ depends on choice of Turing machine U.

- Small impact of U for long strings, but strong impact for short strings.

Algorithmic Probability

A measure that describes the expected output of a random program running on a universal Turing machine U:

Definition

$$m(s) = \sum_{p:U(p)=s} 1/2^{|p|}$$

i.e., sum over all programs for which U with p outputs string s and halts (Levin, 1977).

- probability that randomly selected deterministic program produces s.
- *Algorithmic coding theorem* (Levin, 1974) shows: $K(s) = -\log_2 m(s) + O(1)$, with $O(1)$ independent of s.

Algorithmic Probability

A measure that describes the expected output of a random program running on a universal Turing machine U:

Definition

$$m(s) = \sum_{p:U(p)=s} 1/2^{|p|}$$

i.e., sum over all programs for which U with p outputs string s and halts (Levin, 1977).

- probability that randomly selected deterministic program produces s.
- *Algorithmic coding theorem* (Levin, 1974) shows: $K(s) = -\log_2 m(s) + O(1)$, with $O(1)$ independent of s.

Algorithmic Complexity for Short Strings: ACSS

We approximated $m(s)$ by running (huge samples of random) Turing machines and saved the resulting strings.

Computations performed to approximate $m(s)$

(n, m)	Steps	Machines	Runtime	Strings
(5,2)	500	9 658 153 742 336 (= all)	450 days	all $\|s\| \leq 11$
(4,4)	2000	3.34×10^{11}	62 days	all $\|s\| \leq 11$
(4,5)	2000	2.14×10^{11}	44 days	all $\|s\| \leq 10$
(4,6)	2000	1.8×10^{11}	41 days	all $\|s\| \leq 10$
(4,9)	4000	2×10^{11}	75 days	all $\|s\| \leq 10$

n number of states in Turing machine
m number of symbols

Resulting distributions $m(s)$ available via R package `acss`:
http://cran.r-project.org/package=acss

Applying ACSS

s	K_2	K_4	K_5	K_6	K_9
1: HTHTHTHT	19.84	22.76	23.93	25.08	28.08
2: HTHHTHTT	21.58	24.45	25.62	26.77	29.85

Why do Participants judge **1** as less probable than **2**?

- "presumably because the former appears less random" (Kahneman & Tversky, 1972, p. 432).
- Probability of *any* string produced by random process $P(s|R) = (1/2)^8$
- Perhaps Participants judge converse: $P(R|s)$

Applying ACSS

s	K_2	K_4	K_5	K_6	K_9
1: HTHTHTHT	19.84	22.76	23.93	25.08	28.08
2: HTHHTHTT	21.58	24.45	25.62	26.77	29.85

Why do Participants judge **1** as less probable than **2**?

- "presumably because the former appears less random" (Kahneman & Tversky, 1972, p. 432).
- Probability of *any* string produced by random process $P(s|R) = (1/2)^8$
- Perhaps Participants judge converse: $P(R|s)$

A Bayesian Approach

Bayes Theorem

$$P(R|s) = \frac{P(s|R)P(R)}{P(s|R)P(R) + P(s|D)P(D)},$$

R: random process, *D*: deterministic process.

- $P(s|R)$ is trivial (e.g., $(½)^8$).
- $m(s)$ can be used to approximate $P(s|D)$; normalize across all s with same length.
- Given subjectivity of priors, $P(D)$ and $P(R)$: $\frac{P(R|s)}{P(D|s)} = \frac{P(s|R)}{P(s|D)} \times \frac{P(R)}{P(D)}$
  1. HTHTHTHT: .036
  2. HTHHTHTT: .124

A Bayesian Approach

Bayes Theorem

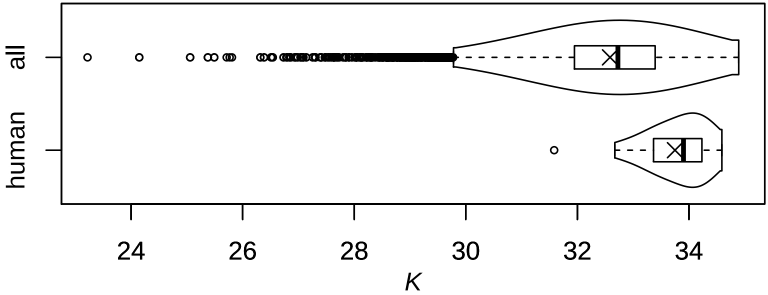
$$P(R|s) = \frac{P(s|R)P(R)}{P(s|R)P(R) + P(s|D)P(D)},$$

R: random process, D: deterministic process.

- $P(s|R)$ is trivial (e.g., $(½)^8$).
- $m(s)$ can be used to approximate $P(s|D)$; normalize across all s with same length.
- Given subjectivity of priors, $P(D)$ and $P(R)$: $\frac{P(R|s)}{P(D|s)} = \frac{P(s|R)}{P(s|D)} \times \frac{P(R)}{P(D)}$
  1. `HTHTHTHT`: .036
  2. `HTHHTHTT`: .124

Applications I: Restricted Human Randomness

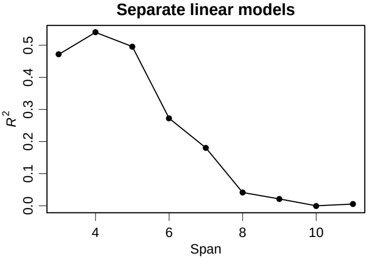
- 34 participants asked to produce a series of 10 symbols using "A", "B", "C", and "D" that would
- "look as random as possible, so that if someone else saw the sequence, she would believe it to be a truly random one"

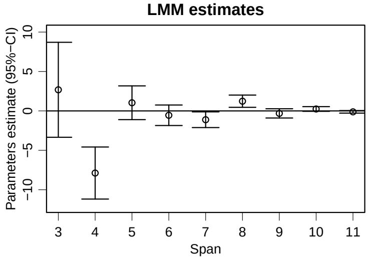


- 0.5% (= 220) more random strings exist; e.g., `ABCDCBBDAC`

Applications II: Local Complexity

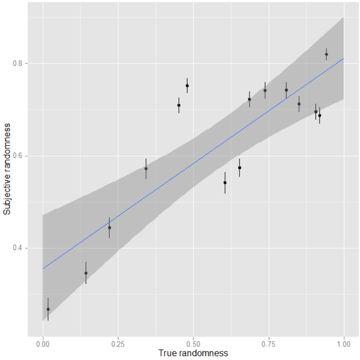
- Hahn (2014) suggested that due to working memory limitations longer sequences are evaluated piecewise.
- Consider “aaabcbad” (8-characters, 4-symbols), *local complexity* with `span` = 6 will return K_4(aaabcb), K_4(aabcba), and K_4(abcbad), which equals (18.6, 19.4, 19.7).
- Matthews (2013, Experiment 1) asked participants to rate binary strings of length 21 on 6-point scale ranging from “definitely random” to “definitely not random”.





Applications III: Conspiracy Theories (CT)
“Nothing happens by accident”

- Dieguez, Wagner-Egger, & Gauvrit (in press, Psych. Sci.) tested if belief in CT correlates with decreased perception of randomness.
- In each of 3 experiments ($N \approx 500$) different measures of CT correlated with each other.
- Ratings of randomness for binary strings correlated strongly with actual randomness (r = [.5, .8]) **but not with CT.**
- Belief in CT is *not* associated with a low-level sensory deficit.

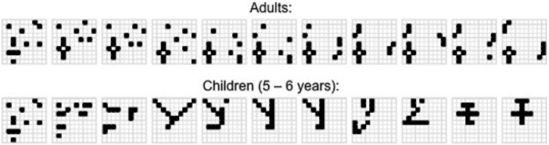


Summary

- We provide an approximation to algorithmic probability $m(s)$
- $m(s)$ approximates *the* objective measure of complexity and randomness for short strings: $K(s)$

Future Plans:

- Extend the supported length using *Block Decomposition Method*.
- Extend the support to two dimensional strings (e.g., Kempe, Gauvrit, & Forsyth, 2015, Cognition):



Thank you for your attention.

Nicolas Gauvrit
(Université de Paris)

Fernando Soler-Toscano
(Universidad de Sevilla)

Hector Zenil
(Karolinska Institute, Stockholm)

I thank my collaborators in the **Algorithmic Nature Group**.

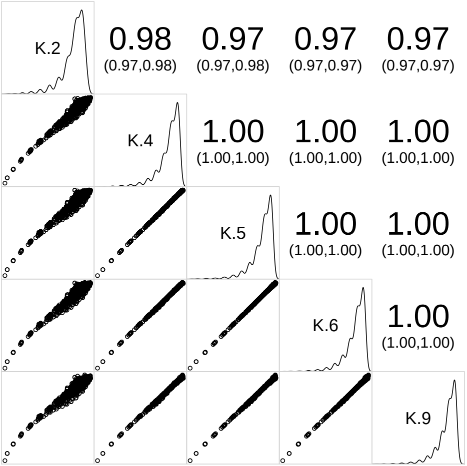
Gauvrit, N., Singmann, H., Soler-Toscano, F., & Zenil, H. (in press). Algorithmic complexity for psychology: a user-friendly implementation of the coding theorem method. *Behavior Research Methods*.

http://algorithmicnature.org/
http://singmann.org/

Validity I

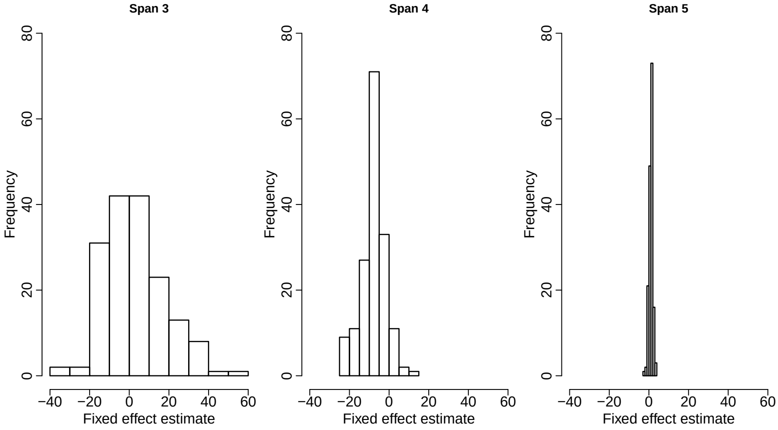
- Statistical evidence that extending or reducing Turing machine sample space does not impact order (Zenil et al., 2012; Soler-Toscano et al., 2013, 2014)
- Correlation in output distribution using very different computational formalisms, e.g. cellular automata and Post tag systems (Zenil and Delahaye, 2010).
- $-\log_2(m(s))$ produces results compatible with compression methods to $K(s)$ and strongly correlates to direct $K(s)$ calculation (i.e., length of first shortest Turing machine found producing s; Soler-Toscano et al., 2013, PLOS ONE)

Validity II



(all 2047 binary strings)

Matthews 2013: Individual Differences



Algorithmic Complexity

Definition

[Kolmogorov(1965), Chaitin(1966)]

$$K_U(s) = \min\{|p| : U(p) = s\}$$

- Algorithmic complexity $K(s)$ of a string s is length of shortest program p that produces s running on a universal Turing machine U.
- The formula conveys: a string with low algorithmic complexity is highly compressible, the information it contains can be encoded in a program much shorter in length than the length of the string itself.
- Although $K_U(s)$ is *uncomputable*, it is *upper semi-computable*. It can be approximated from above.
- For long strings $K(s)$ can be approximated via lossless compression and impact of U is small (*invariance theorem*).
- For short strings (e.g., length < 100), impact of U is considerable (e.g., $K_{U'}(s_1) < K_{U'}(s_2)$ whereas $K_{U''}(s_1) > K_{U''}(s_2)$) and lossless compression no option.

Algorithmic Complexity

Definition

[Kolmogorov(1965), Chaitin(1966)]

$$K_U(s) = \min\{|p| : U(p) = s\}$$

- Algorithmic complexity $K(s)$ of a string s is length of shortest program p that produces s running on a universal Turing machine U.
- The formula conveys: a string with low algorithmic complexity is highly compressible, the information it contains can be encoded in a program much shorter in length than the length of the string itself.
- Although $K_U(s)$ is *uncomputable*, it is *upper semi-computable*. It can be approximated from above.
- For long strings $K(s)$ can be approximated via lossless compression and impact of U is small (*invariance theorem*).
- For short strings (e.g., length < 100), impact of U is considerable (e.g., $K_{U'}(s_1) < K_{U'}(s_2)$ whereas $K_{U''}(s_1) > K_{U''}(s_2)$) and lossless compression no option.