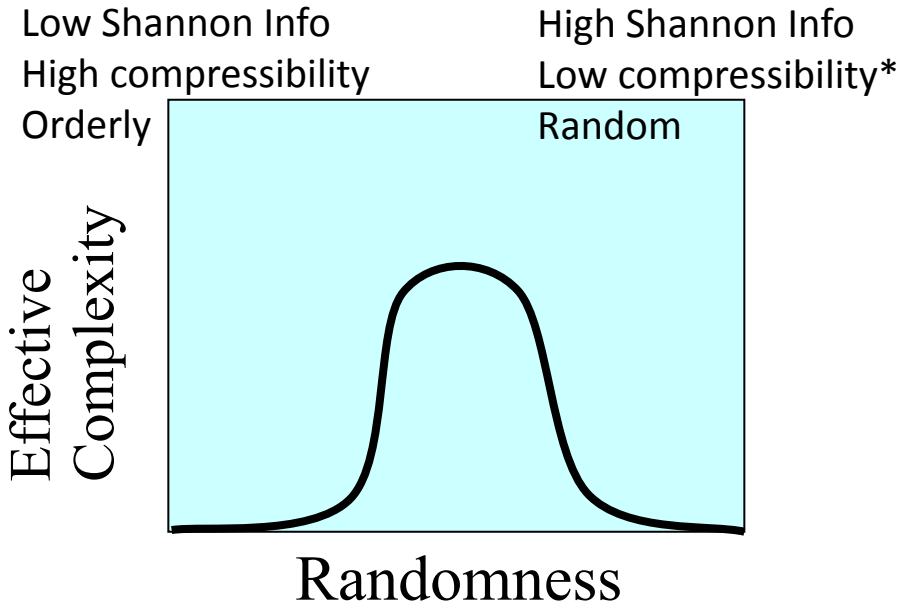
Measures of Complexity

- Entropy
- AIC (Kolmogorov Complexity)
- EFFECTIVE COMPLEXITY
- Total Information
- Logical Depth
- Statistical Complexity
- Size
- Fractal Dimension

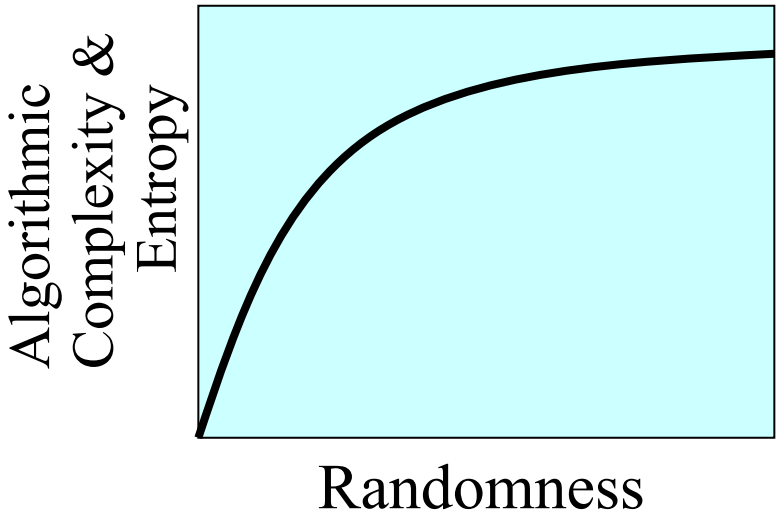
Gell-Mann's Effective Complexity

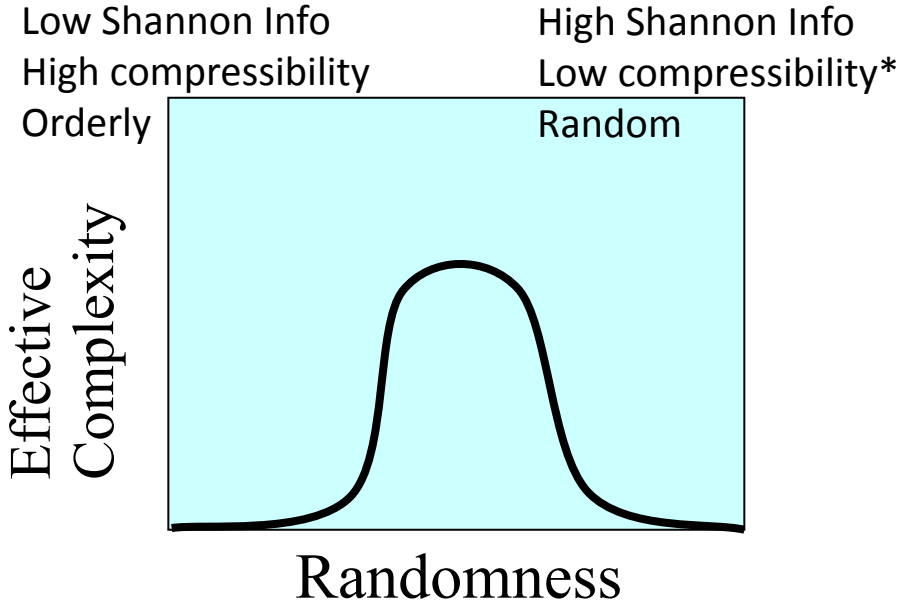


Gell-Mann's Effective Complexity

- The length of the **shortest description of** a set's **regularities**
- EC(x) = K(r) where r is the set of regularities in x

  and Kolmogorov Complexity (or AIC) or K(r), is the length of the most concise description of a set
- Highest for entities that are not strictly regular or random





Algorithmic Complexity

(also known as Kolmogorov-Chaitin complexity or Algorithmic Information Content - AIC)

- *Kolomogorov* complexity or Algorithmic Information Content (AIC), written *K(x),* is the length, in bits, of the smallest program that when run on a Universal Turing Machine outputs (prints) *x* and then halts.
- Example: What is *K(x)*
  - where *x* is the first 10 even natural numbers?
  - where *x* is the first 5 million even natural numbers?

- Possible representations where *n* is the length of *x*
  - *0, 2, 4, 6, 8, 10, 12, 14, 16, 18, … (2n - 2)*
  - for (j = 0; j < n: j++) printf("%d\n", j * 2)

Algorithmic Complexity

(also known as Kolmogorov-Chaitin complexity or Algorithmic Information Content - AIC)

- *Kolomogorov* complexity or Algorithmic Information Content , written *K(x),* is the length, in bits, of the smallest program that when run on a Universal Turing Machine outputs (prints) *x* and then halts.
- Example: What is *K(x)*
  - where *x* is the first 10 even natural numbers?
  - where *x* is the first 5 million even natural numbers?

- Possible representations where *n* is the length of *x*
  - *0, 2, 4, 6, 8, 10, 12, 14, 16, 18, … (2n - 2)* *K(x) = O(n log n) bits*
  - for (j = 0; j < n: j++) printf("%d\n", j * 2) *K(x) = O(log n) bits **

**Complexity of program size (length), not program running time*

Algorithmic Complexity

(also known as Kolmogorov-Chaitin complexity or Algorithmic Information Content - AIC)

- Kolmogorov complexity formalizes what it means for a set of numbers to be compressible
  - Data that are redundant can be compressed and have lower complexity
  - Random strings are incompressible, therefore contain no regularities to compress - they have HIGH Kolmogorov complexity
- Implication: The more random a system, the greater its AIC (and greater entropy)
- Contrast with Statistical simplicity

  Random strings are simple because you can approximate them statistically
  - Coin toss, random walks, Gaussian (normal distributions)
  - You can compress random numbers with statistical descriptions and only a few parameters

- s1 = 1111111111111111
  for i:=1:16
  print '1'

  K(s) is O(log n) where n = length(s1)
  The dominant term is the number "16" whose representation in bits will require $\log_2 16 = 4$ bits

- S2 is a string of 1's, length(s2) = 1 billion
  for i:=1:1,000,000,000
  print '1'

K(s) *if s is a repeated string with no uncertainty, then* K(s) is the number of bits required to represent the length of the string: K(s) is $O(\log_2 n)$

K(s1) is $O(\log_2 16)$ so K(s1) = 4 bits
K(s2) is $O(\log_2 10^9)$ so K(s2) = 30 bits

Constant terms such as the representation of the print statement are ignored

The ***minimum*** K(x), given length(x) = n, is log(n).

What is the AIC for the following strings?
s3 = 0101010101010101
s4 = 011011011011011
s5 = 0001000010001000010001
s6 = 01110111011000110110101

s3 = 0101010101010101
for i:=1:8
print ‘01’
2 bits are needed to represent the pattern, 3 bits to represent the 8 repetitions
K(s3) = O(log(n))

s4 = 011011011011011
for i:=1:5
print ‘011’
3 bits to represent the repeating pattern, log(5) bits to represent the number of repetitions
K(s4) = O(log(n))

s5 = 000100001000100001
for c = 1:2
print ‘000100001’

8 bits to represent the repeating pattern, 1 bit to represent the repetitions. Here the length of the pattern is ½ n, so
K(s5) = O(n)

s6 = 011101110110001101101101

K(s6), if s6 is a random string, there is no way to compress it. Then K(s6)= length(s6) = 24 bits

There are two problems with AIC (and therefore, with Effective Complexity)

- Calculation of K(x) depends on the machine we have available (e.g., what if we have a machine with an instruction “print the first 10 even natural numbers”?)
  - COMPLEXITY DEPENDS ON CONTEXT
- Determining K(*x*) for arbitrary *x* is uncomputable

Problem 1: K(x) depends on the programming language

Resolution: optimal specification functions can be defined so that

"The complexity of an object *x* is invariant (up to an additive constant independent of *x*) under transition from one optimal specification function to another."

(Li & Vitanyi "An Introduction Kolmogorov Complexity and its Applications" 2008)

http://jeremykun.com/2012/04/21/kolmogorov-complexity-a-primer/

Problem 2: Identifying the shortest program to print the string is uncomputable

Resolution: None

It is not possible to determine the amount of randomness in any arbitrary string

Regularities can be difficult to identify

- s6 = 011101110110001101101 01
- It looks like a random string, so K(s6) = length(s6)

But its not !

- What is the shortest program that would produce this string?

Fibonacci Series

0	1	1	10	11	101	1000	1101	10101
0	1	1	2	3	5	8	13	21

A short computer program (of length (L_{fb}) that produces the Fibonacci series can generate s6, so K(s6) can be reduced to max (L_{fb}, log(n) where n specifies the length of the Fibonacci series to be printed.

What About

s7=00100100001111110110101010001000100001011010001
100001000110100...?

This is a statistically random string.

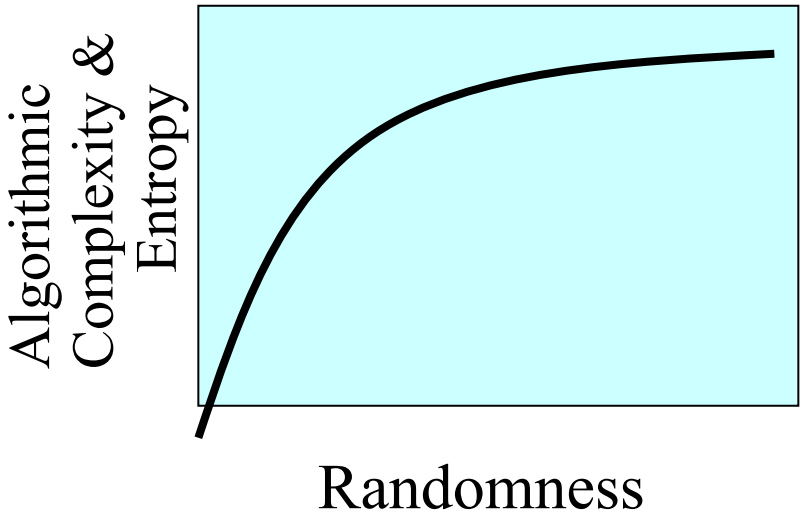
It is also the binary representation of the first decimals of pi.

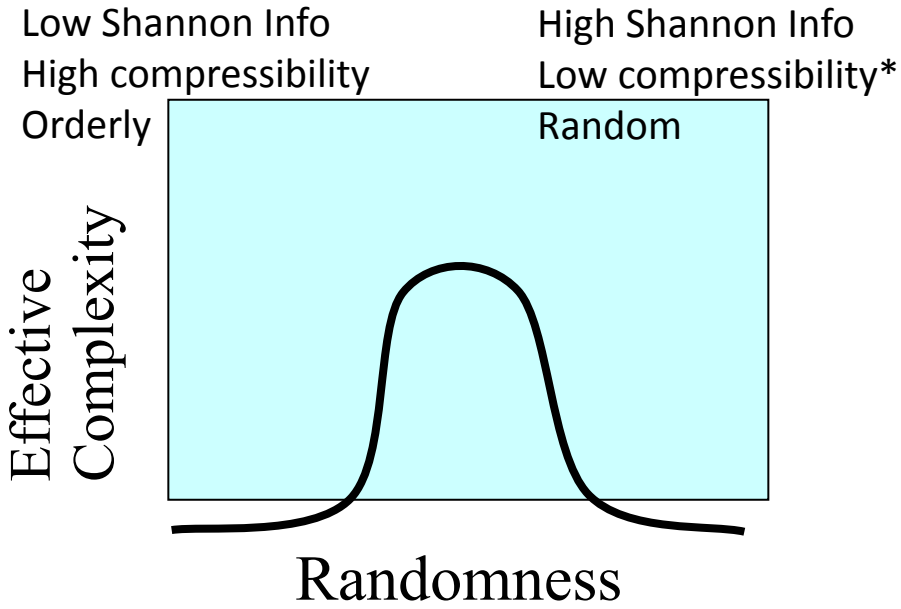
A short program could generate this “random” string.

Revisting Gell-Mann's Effective Complexity

- Effective Complexity (EC) is **NOT** the same as AIC.
- It is the length of the AIC (the shortest description) **of a set's regularities**
- EC(x) = K(r) where r is the set of regularities in x

  and Kolmogorov Complexity (or AIC), K(r), is the length of a concise description of a set
- Highest for entities that are not strictly regular or random





What's a regularity?

To calculate Effective Complexity, 3 steps (know these steps)

1. Identify the regularities, r
2. Calculate K(r), the AIC of the regularities
3. Effective complexity = K(r)

Calculating regularities is hard, Gell Mann suggests one formal way

(FYI, this is not testable material but is a way to think about regularities and predictability

- Determine mutual Kolmogorov complexity between parts of the string
- If x = [x1,x2]
- K(x1,x2) = K(x1) + K(x2) – K (x)
- The sum of the complexity of the parts – the complexity of the whole
- Regularities exist when K(x1,x2) > 0

Eg. 10010 10011 10010 10101 10010 10101 10010 - the whole has more regularity than the sum of the regularities in the parts, so calculate K(x) of the whole, not the parts K(x1)

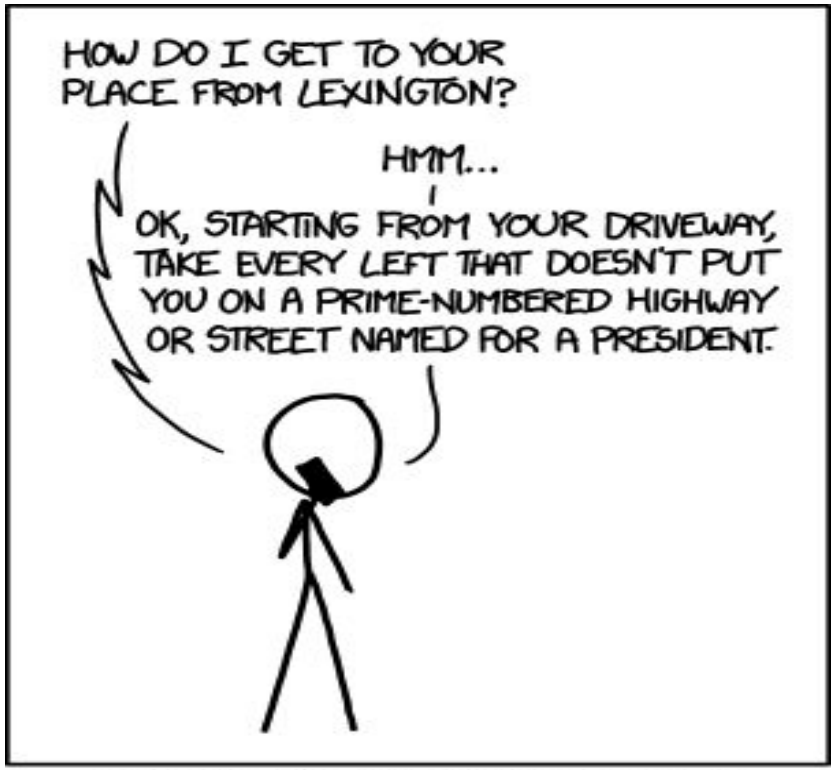
- Here the regularity r = 10010
- AIC of 10010 which is O(5)
- Effective Complexity is O(5)
- Ignore the randomness to calculate EC


What are frozen accidents?

Gell-Mann:
In Evolution “Frozen Accidents” cause regularities

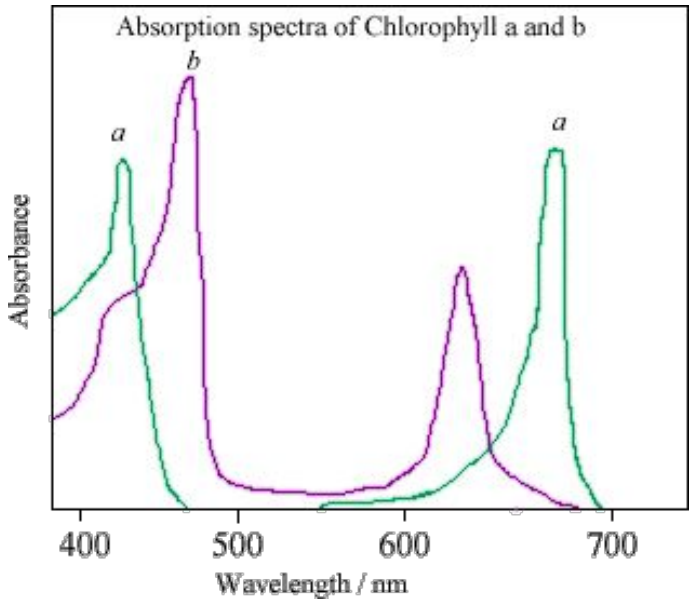
Identify “frozen accidents” (regularities) in genomes

Calculate Effective Complexity as the AIC of the regularities



http://c2.com/cgi/wiki?KolmogorovComplexity

Why are land plants green?



How complex is it?

- Other considerations when defining complexity
- How hard is it to describe?
- How hard is it to create?
- What is its degree of organization?

Lloyd 2001, 40 measures of complexity

One alternative to EC: Logical Depth

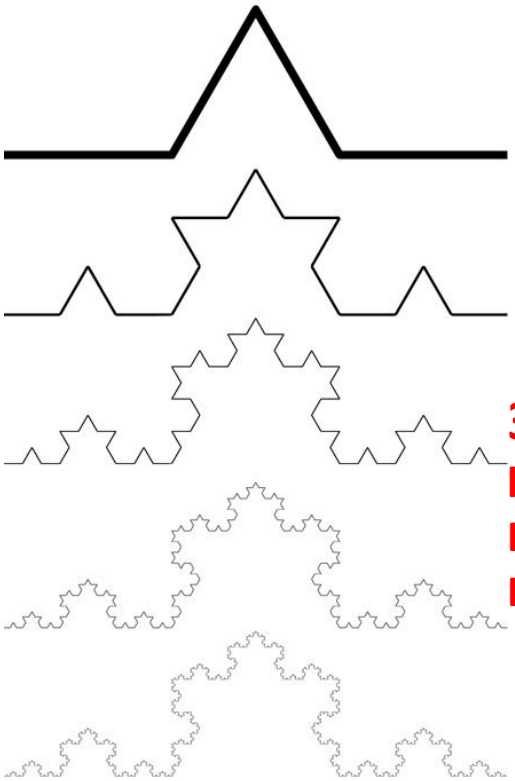
- Bennett 1986;1990:
  - The *Logical depth* of *x* is the run time of the shortest program that will cause a UTM to produce *x* and then halt.
  - Logical depth is not a measure of randomness; it is small both for trivially ordered and random strings.
- Drawbacks:
  - Uncomputable.
  - Loses the ability to distinguish between systems that can be described by computational models less powerful than Turing Machines (e.g., finite-state machines).

- Ay et al 2008, proposed proof that strings with high effective complexity also have high logical depth, and low effective complexity have small logical depth.

Another Alternative: Total Information

- Alternative approach in Gell-Mann & Lloyd 1998
- EC(x) = K(E) where E is the set of entities of which x is a typical member
- Then K(x) is the length of the shortest program required to specify the members of a the set of which x is a typical member
- Effective complexity measures knowledge—the extent to which the entity is nonrandom and predictable
- Total Information is Effective complexity, K(E), + the Shannon Information of the peculiarities (remaining randomness)
- TI(x) = K(E) + H(x)
- There is a tradeoff between the effective complexity (the completeness of a description of the regularities) and the remaining randomness
- Ex: 10010 10011 10010 10011 10010 10011

Fractal Dimension



$3^D = 4,$

$\log(3^D) = \log(4)$

$D = \log(4)/\log(3)$

$D = 1.26$

$\log(3^{dimension}) = dimension \times log(3) = \log(4)$. Thus $dimension$

$= \log(4)/\log(3) \approx 1.26.$

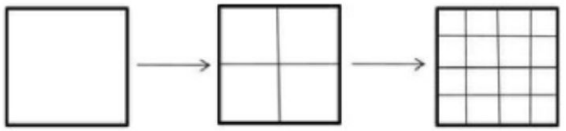
Start with our familiar line segment. Bisect it (i.e., cut it in half). Then bisect the resulting line segments, continuing at each level to bisect each line segment:

$2^1 = 2,$

$D = 1$

Each level is made up of two half-sized copies of the previous level.

Now start with a square. Bisect each side. Then bisect the sides of the resulting squares, continuing at each level to bisect every side:

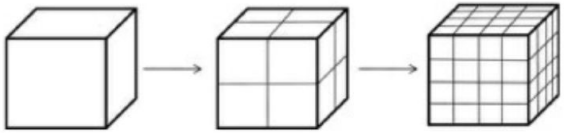


$2^2 = 4,$

$D = 2$

Each level is made up of four one-quarter-sized copies of the previous level.

Now, you guessed it, take a cube and bisect all the sides. Keep bisecting the sides of the resulting cubes:



$2^3 = 8,$

$D = 3$

Each level is made up of eight one-eighth-sized copies of the previous level.

This sequence gives a meaning of the term *dimension*. In general, each level is made up of smaller copies of the previous level, where the number of copies is 2 raised to the power of the dimension ($2^{dimension}$). For the line, we get $2^1 = 2$ copies at each level; for the square we get $2^2 = 4$ copies at each level, and for the cube we get $2^3 = 8$ copies at each level. Similarly, if you trisect instead of bisect the lengths of the line segments at each level, then each level is made up of $3^{dimension}$ copies of the previous level. I'll state this as a general formula:

Create a geometric structure from an original object by repeatedly dividing the length of its sides by a number x. Then each level is made up of $x^{dimension}$ copies of the previous level.

Complexity Measures

- Information-theoretic methods:
  - Shannon Entropy
  - Algorithmic complexity
  - Mutual information between a system and its environment
- Effective Complexity:
  - Neither regular nor random entities have high Effective Complexity
- Total Information: Effective Complexity + Entropy
    - AIC of regularities + entropy of what remains

- Computational complexity:
  - How many resources does it take to compute a function?
- The language/machine hierarchy:
  - How complex a machine is needed to compute a function?
- Logical depth:
  - Run-time of the shortest program that generates the phenomena and halts.
- Fractal Dimension
- Asymptotic behavior of dynamical systems:
  - Fixed points, limit cycles, chaos

Defining Complexity
Suggested References

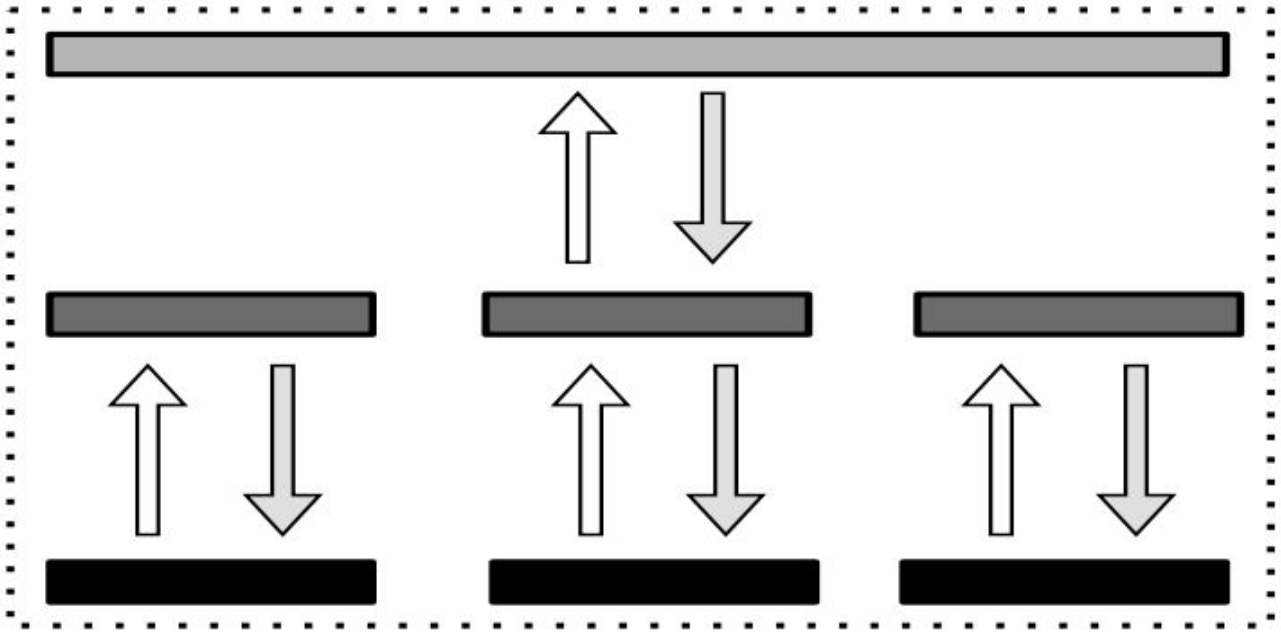
- *Computational Complexity* by Papadimitriou. Addison-Wesley (1994).
- *Elements of Information Theory* by Cover and Thomas. Wiley (1991).
- Kaufmann, *At Home in the Universe* (1996) and *Investigations* (2002).
- Per Bak, *How Nature Works: The Science of Self-Organized Criticality* (1988)
- Gell-Mann, *The Quark and the Jaguar* (1994)
- Ay, Muller & Szkola, Effective Complexity and its Relation to Logical Depth, ArXiv (2008)

- Kolmogorov Complexity With Python https://www.youtube.com/watch?v=KyB13PD-UME
- http://en.wikipedia.org/wiki/Kolmogorov_complexity
- http://www.neilconway.org/talks/kolmogorov.pdf
- http://people.cs.uchicago.edu/~fortnow/papers/quaderni.pdf
- http://c2.com/cgi/wiki?KolmogorovComplexity

What else is missing from these definitions of Complexity?

Feedback
between hierarchical levels of organization

Complex systems are not static strings. They are dynamic systems with adaptive feedbacks.



What are frozen accidents in our computational infrastructure?

- Are they ‘accidents’ or there for a reason?
- Are they beneficial or detrimental or neither?
- If detrimental, can they be changed or overcome?
- How do they influence or constrain communication, computation and their future trajectories or evolution?

Discussion:COVID-19 as a CAS

- What agents or organizations are adapting?
- How do levels of organization affect each other? (consider populations of viruses & humans, individuals, genomes & societies)
- In what ways is the pandemic a frozen accident?
- How should an effective response react to the complexity of the pandemic?