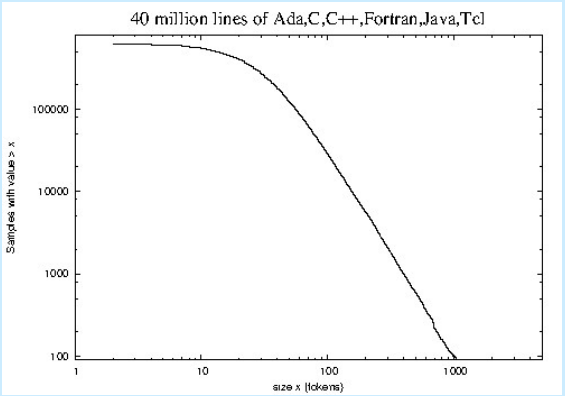
“Conservation of Information in Computing and Biology: Nature’s Hidden Clockwork ?"

Les Hatton and Greg Warr*

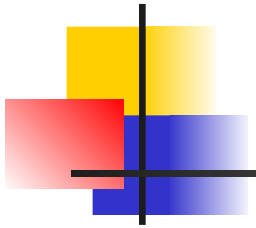
www.leshatton.org

Version 1.1: 27/FEB/2013

*Medical University of North Carolina

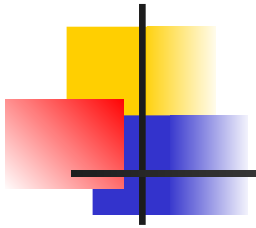


Overview



- Defects in software systems
- A hidden clockwork
- Analogues with the genome
- Defects and the genome
- Conclusions

So what can we say about defect ?



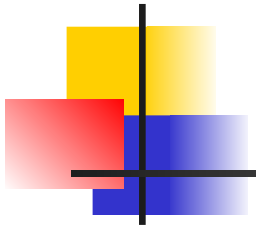
- On quantification
  - Computer scientists have researched the average *density* of defect in code extensively
  - Where we have been much less successful is in quantifying the *effects* of such defect on numerical results.

They occur in interesting places …

26th February 2007

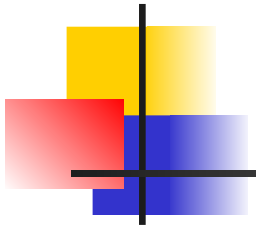
6 F-22 Raptors were left without major systems when the systems crashed after crossing the International Date Line on route from Okinawa to Hawaii. “It was a software glitch – somebody made an error in a couple of lines of code out of millions.”

They are surprisingly common …



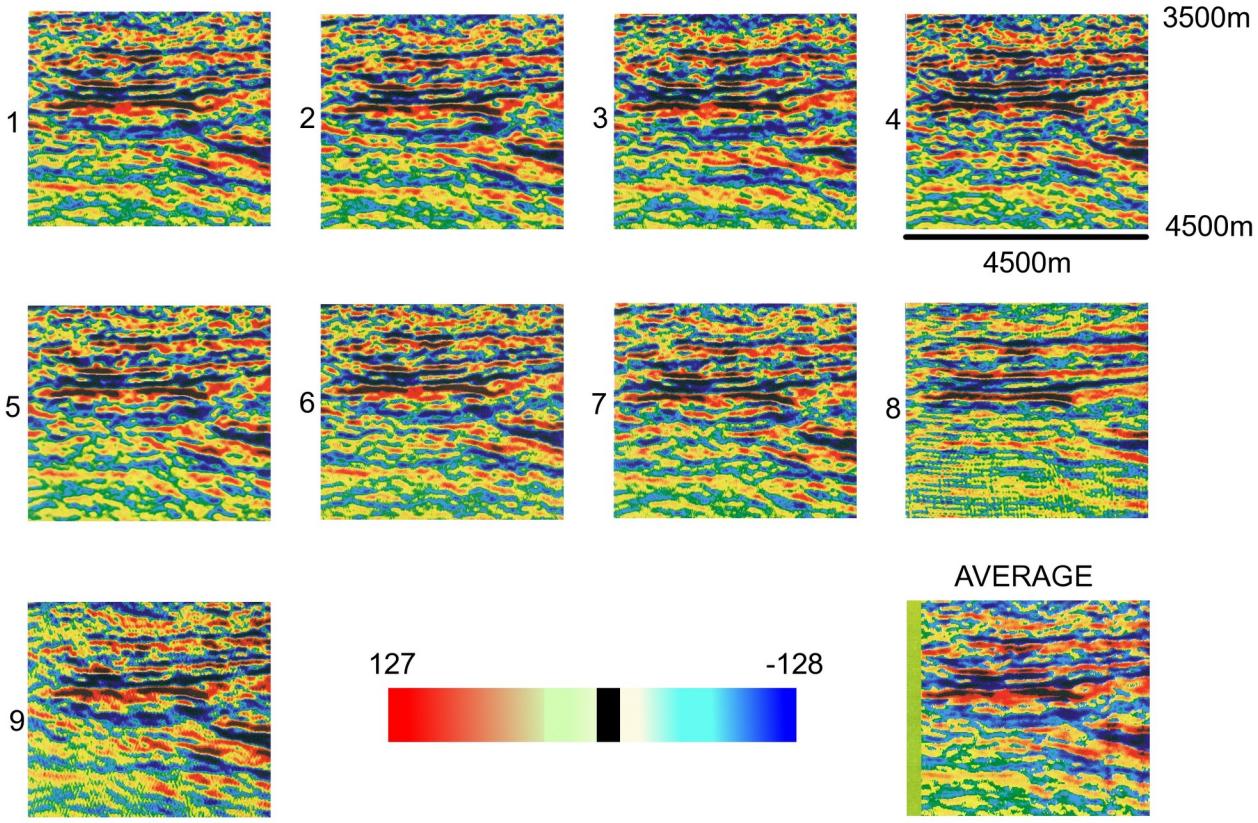
- Small recent selection from scmagazineuk.com …
  - 22-25 Jun 2012 "*Routine* software update stops two of Britain's biggest banks (NatWest and RBS) processing payments."
  - 16-Jul-2012 "Symantec fixes blue screen of death bug"
  - 21-Aug-2012 "McAfee's defective update causes user chaos"
  - 03-Sep-2012 "Oracle issue emergency patch for Java but more flaws are found"
  - 27-Sep-2012 "Brucon: Major flaws in eID web applications revealed"
  - 10-Oct-2012 "Microsoft issues one critical patch and seven important on October's Patch Tuesday"

How big must a defect be before we notice it ?



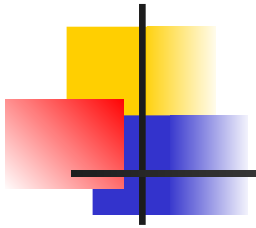
- 11 October 2012
  - *"French Woman stunned after phone company sends her **EUR 11.8 quadrillion** phone bill"*
  - *This comfortably beats the previous record of …*
- 10 April 2006
  - *"Malaysian man gets **USD 218 trillion** phone bill"*
- In both cases, the victim had difficulty convincing the phone company they had made a mistake.

How small must a defect be before we miss it ?



Comparison of 9 commercial packages using the same algorithms on the same data in the same programming language, (Hatton and Roberts (1994), IEEE TSE)

Effects on scientific progress involving computation

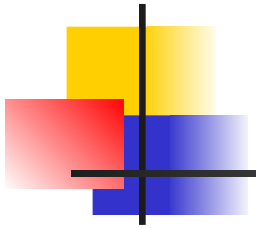


Scientific computation and reproducibility*

- Reproducibility is at the heart of the scientific method
- Without source code it is impossible to reproduce scientific results. (Even with it, there are considerable problems)
- Scientific work without accompanying data, source code and the means to reproduce is **not** science.

*See the discussion in Darrel Ince, Les Hatton and John Graham-Cumming (2012) "The case for open computer programs", Nature 482, p. 485-488, 23 Feb 2012.

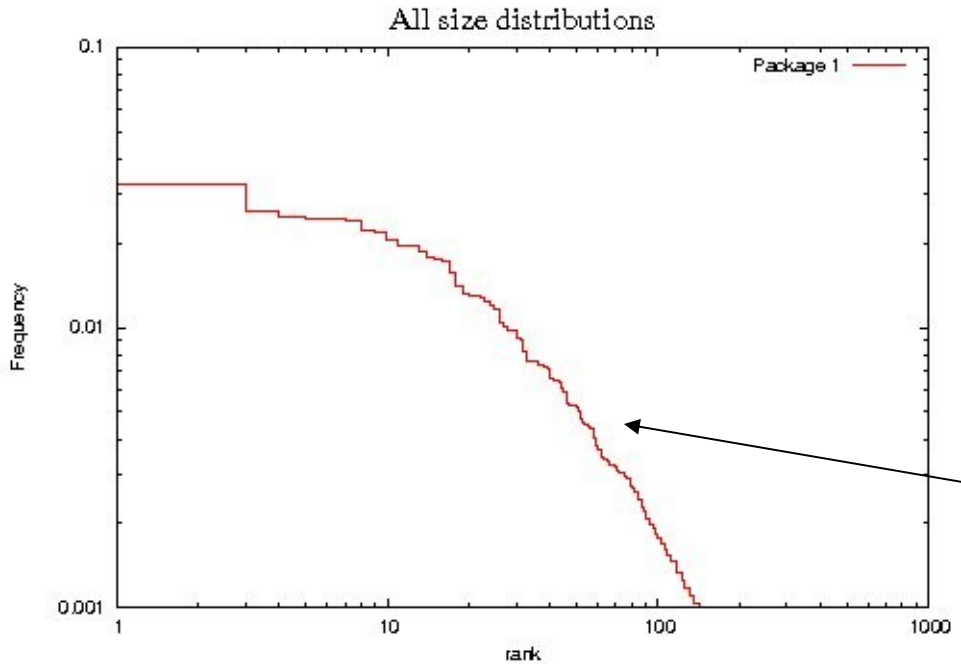
Overview



- Defects in software systems
- A hidden clockwork
- Analogues with the genome
- Defects and the genome
- Conclusions

Software size distributions appear power-law **in LOC**

Systems appear astonishingly similar in their component size distributions …



Smoothed (cdf) data for 21 systems, C, Tcl/Tk and Fortran, combining 603,559 lines of code distributed across 6,803 components, (Hatton 2009, IEEE TSE)

Power-law is linear on log-log plot

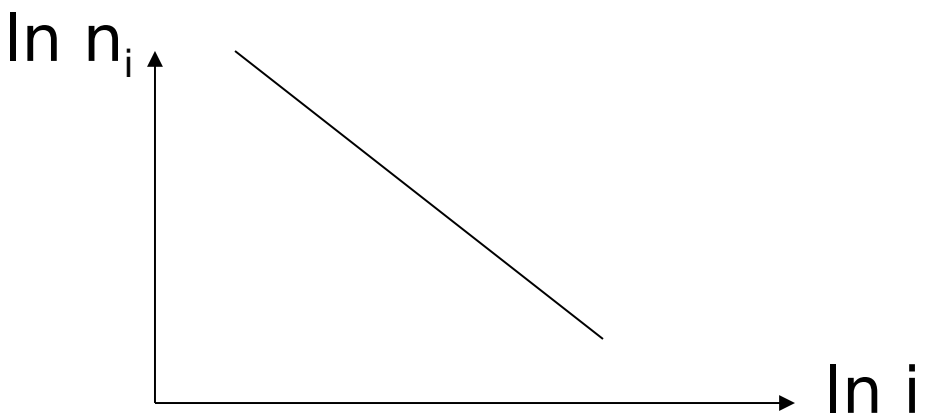
What is power-law behaviour ?

Frequency of occurrence n_i given by $n_i = \frac{nc}{i^p}$

This is usually shown as

$$\ln n_i = \ln(nc) - p \ln i$$

which looks like



A model for emergent power-law size behaviour using tokens

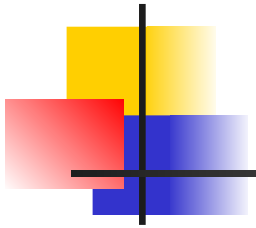
Suppose component i in a discrete system has t_i tokens in all constructed from an alphabet of a_i unique tokens.

First we note that

$$a_i = a_f + a_v(i)$$

(Programming Languages)	Fixed tokens of a language, **{ } [ ] ; while ...**	Variable tokens, (id names and constants)
Genome	ACTG	Methylation

How do we build components from tiny pieces (tokens) ?



Take an example from C:

Fixed (18)

void int ( ) [ ] { ,
; for = >= -- <=
++ if > -

+

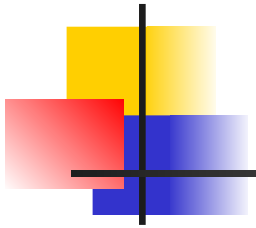
Variable (8)

bubble a N i j t 1
2

```
void bubble( int a[], int N)
{
 int i, j, t;
 for( i = N; i >= 1; i--)
 {
  for( j = 2; j <= i; j++)
  {
   if ( a[j-1] > a[j] )
   {
    t = a[j-1]; a[j-1] = a[j]; a[j] = t;
   }
  }
 }
}
```

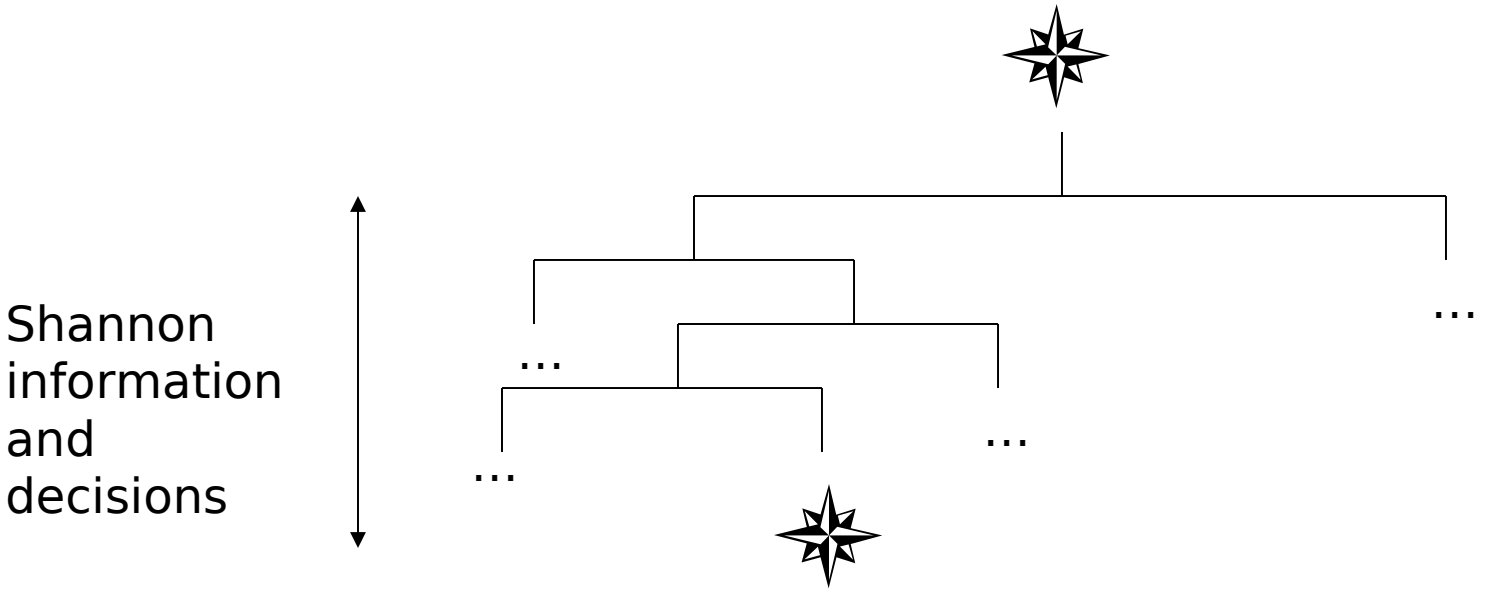
Total (94)

Building systems using tokens



- Some notes on Shannon information theory
  - Based on assembling messages from non-divisible pieces, such as bits.
  - Suppose we have an N-bit stream of 1s and 0s. There are 2^N possible ways of arranging these.
  - Shannon was interested in representing the complexity or *information content* of these and realised following Hartley in 1928 that $\log_2 (2^N) = N$ was related to the number of choices need to find a particular combination

Building systems



Depth in tree is essentially the log of the total number of destinations

Adding in Hartley-Shannon information

For an alphabet a_i the Hartley-Shannon information content for component i is given by

$$I_i = \log(a_i a_i ... a_i) = \log(a_i^{t_i}) = t_i \log(a_i)$$

How do we build systems from components ?

So, consider a general software system of T tokens divided into M pieces each with t_i tokens, *in which size and choice (Hartley-Shannon information I) is conserved.*

1	2	3	….			
			t_i, I_i			
					…	M

$$T = \sum_{i=1}^{M} t_i$$

$$I = \sum_{i=1}^{M} I_i$$

General mathematical treatment and the Clockwork Theorem

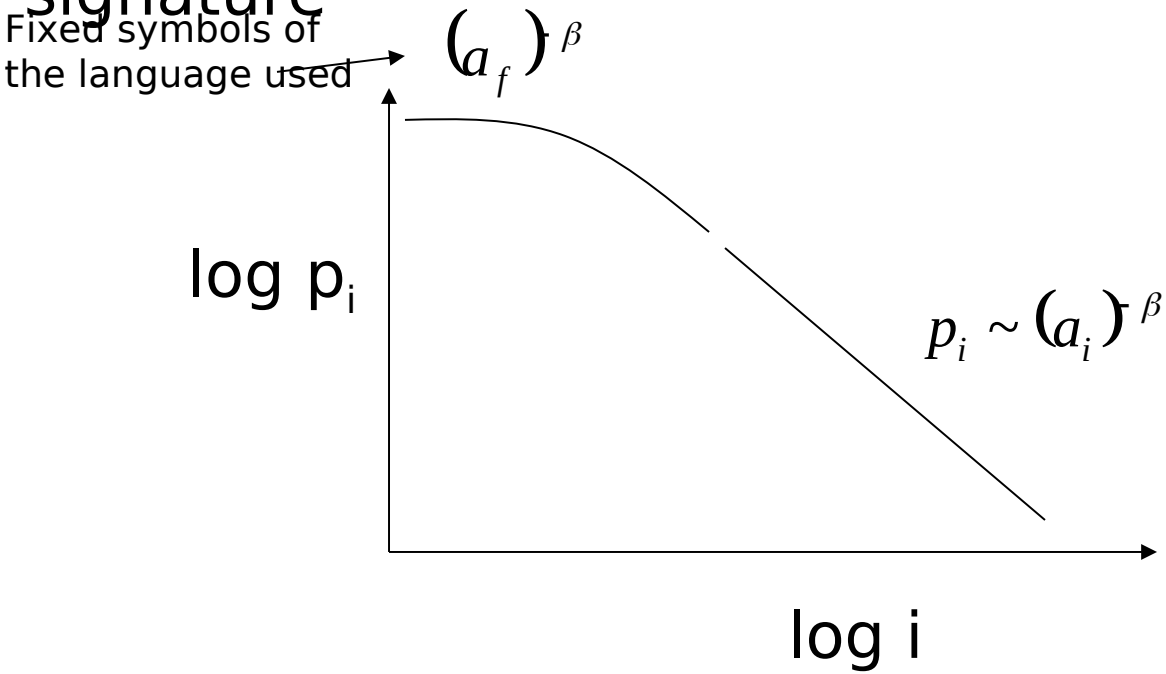
It can be shown

$$p_i \sim (a_i)^{-\beta}$$

This states that in any discrete system, conservation of size and information (i.e. choice) is overwhelmingly likely to produce a power-law alphabet distribution. (Hatton (2011), IFIP, Boulder Colorado).

Application to software systems

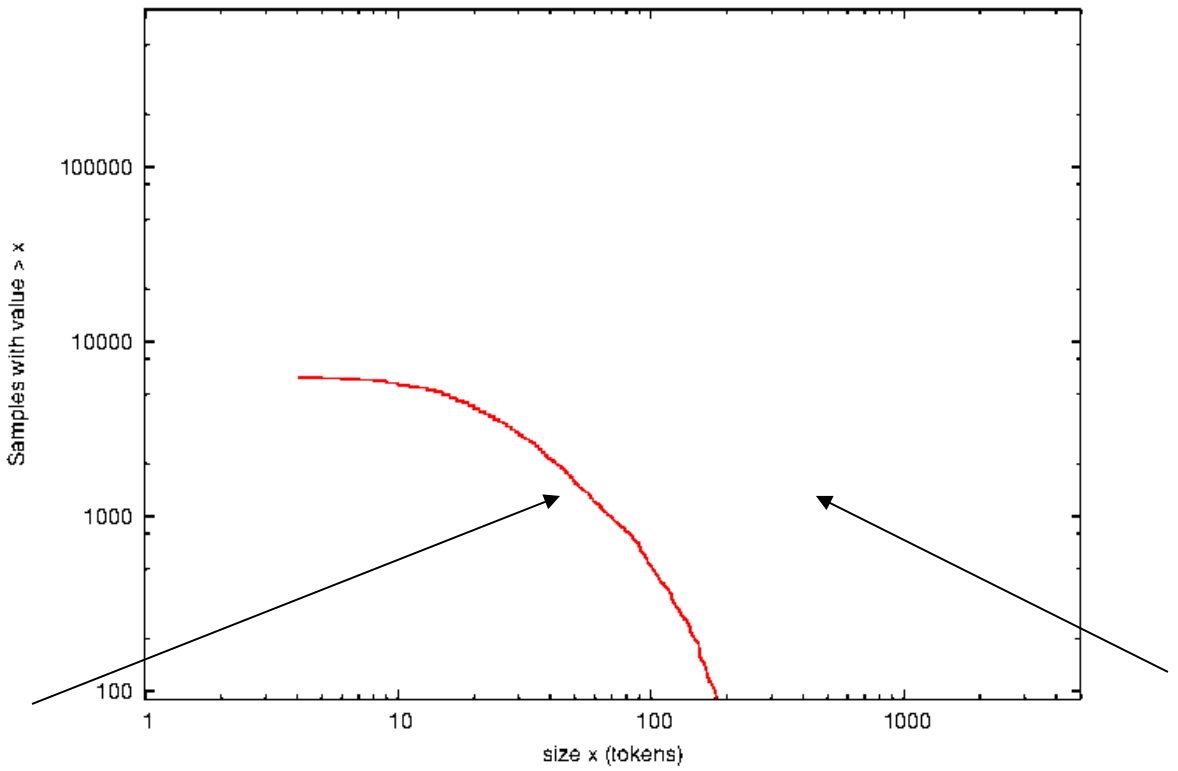
Verification: We are looking for the following signature



This requires writing a compiler front end (lexical analyser) for each programming language analysed.

Equilibriation to the clockwork theorem in ~ 500,000 line chunks

Model fits to better than p = 1.0 10^{-16}



Equilibriation

55 million lines of Ada, C, C++, Fortran, Java, Tcl-Tk from 90+ systems

Scale-free behaviour

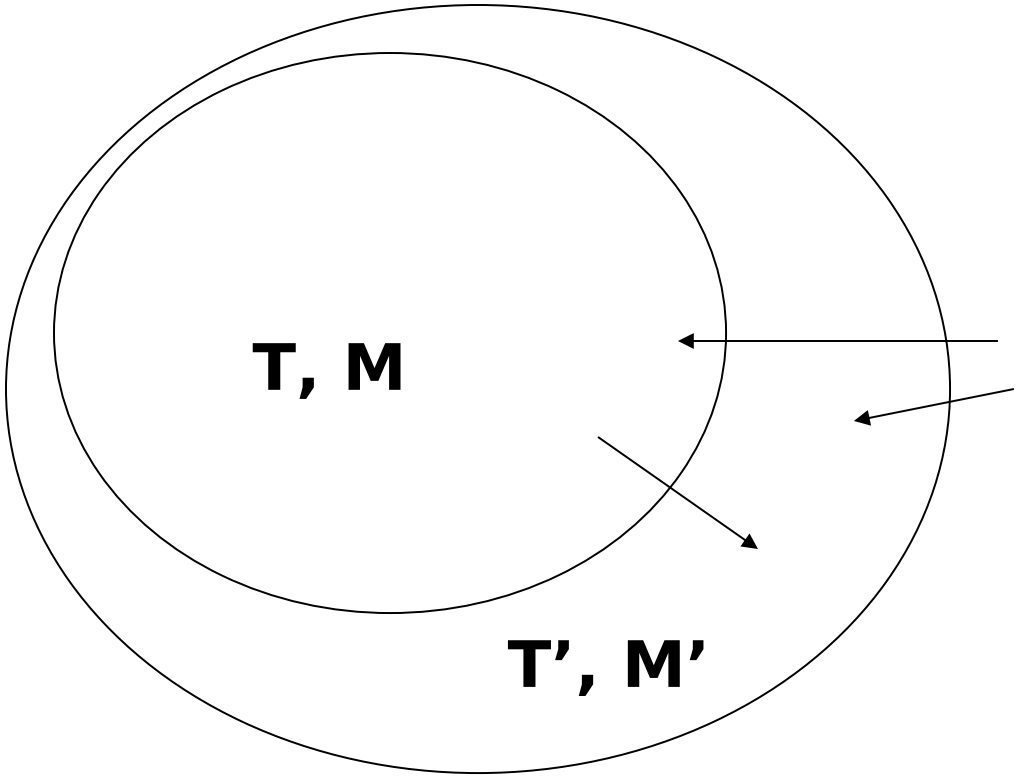
Why would information be conserved ?

- Physical Systems (Noether (1915))
  - Conservation of Energy / Time symmetry
  - Conservation of Linear Momentum / Displacement symmetry
  - Conservation of Angular Momentum / Rotational symmetry
  - Conservation of Charge / Phase symmetry
- Discrete token-based systems
  - Conservation of Information / ***Scale***.

  is independent of T and M.

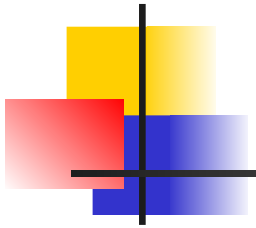
$$p_i \sim (a_i)^{-\beta}$$

Conservation under evolution



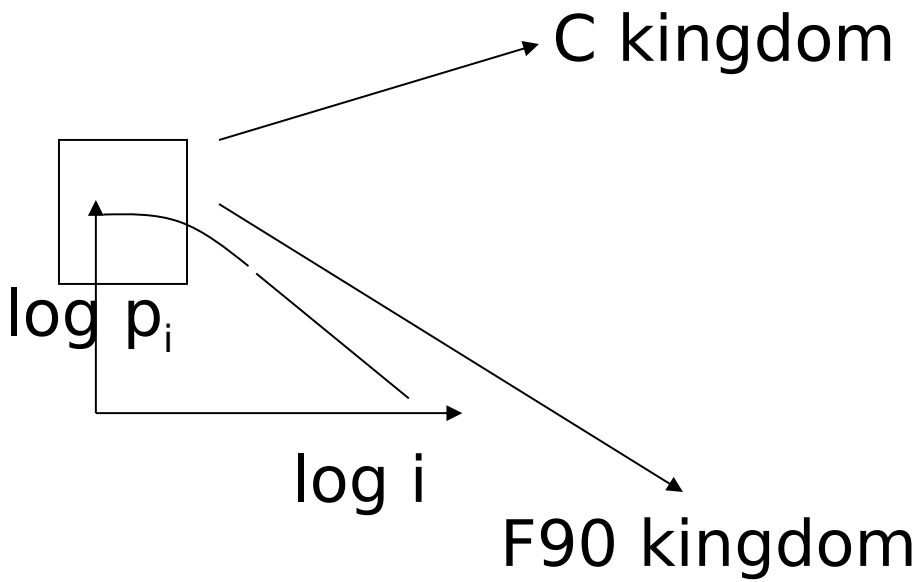
$$p_i \sim (a_i)^{\beta}$$

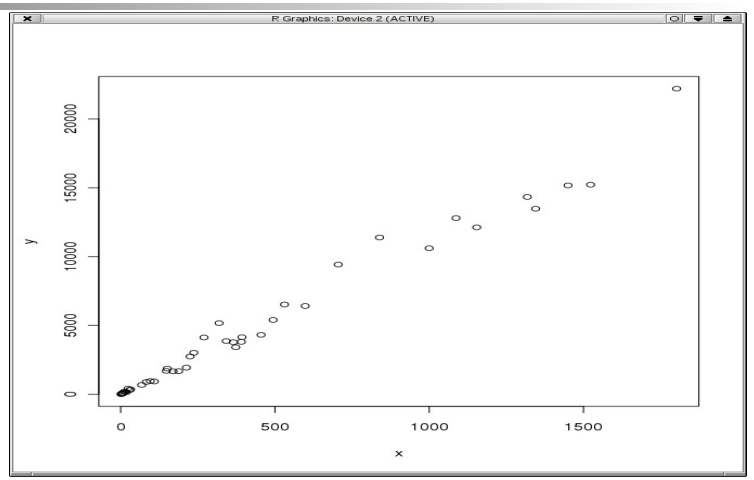
Overview

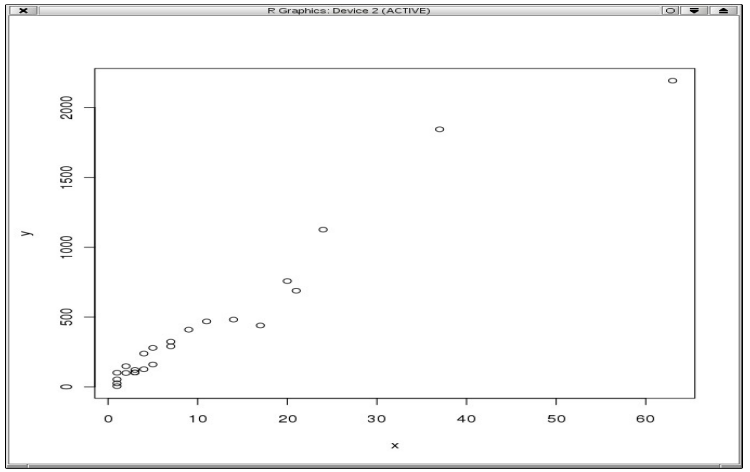


- Defects in software systems
- A hidden clockwork
- Analogues with the genome
- Defects and the genome
- Conclusions

An observation on software systems







If we confine ourselves to small software components, component size should be uniformly distributed => same average length

Genetic example of uniform length distribution

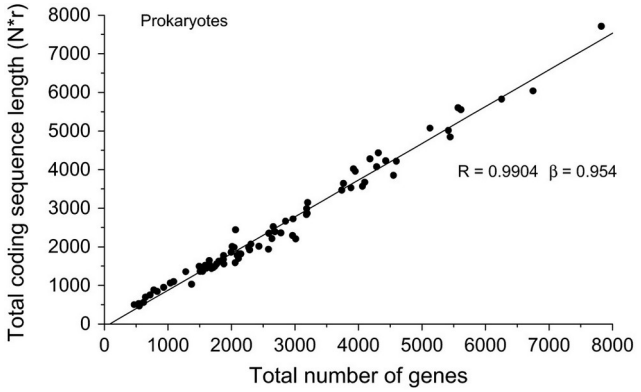
- To first order, genes have a simple *constant* alphabet, $a_i = 4$, (ACTG)
- The Clockwork Theorem then implies that the random variable L representing gene length is *uniformly distributed*, giving

$$E(L) = k.\frac{T}{M}$$

T is the total length of the genomic sequence and M the number of genes. k is a constant.

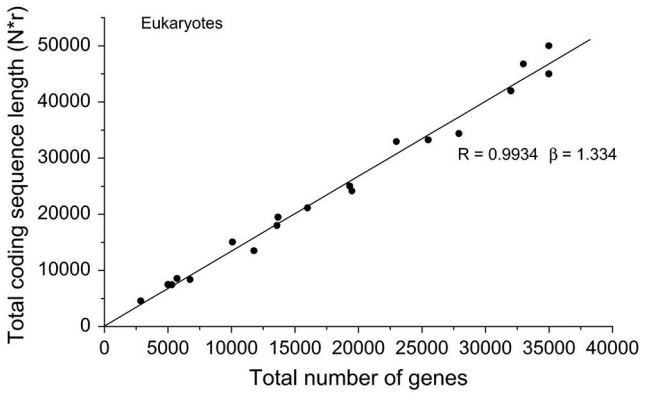
Genetic example of alphabetic power-law theorem

Prokaryotes



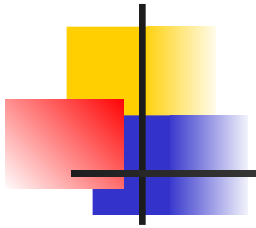
This is indeed observed :-

Eukaryotes



Xu et al. (2006) “Average gene length is highly conserved in Prokaryotes and Eukaryotes and diverges only between the two kingdoms”, Mol Biol Evol (June 2006) 23 (6), p. 1107-1108

Overview



- Defects in software systems
- A hidden clockwork
- Analogues with the genome
- Defects and the genome
- Conclusions

So, what can we say about defects ?

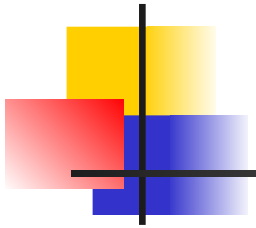
- However, systems evolve such that the total number of defects D is conserved, (since they are finite), (Hatton, (2009) IEEE TSE, 35(4), p. 566-572), giving

$$D = \sum_{i=1}^{M} d_i \Rightarrow p_i = \frac{1}{Q(\gamma)} e^{-\gamma \frac{d_i}{t_i}}$$

- Combining this with the clockwork theorem

$$p_i \sim (a_i)^{-\beta}$$

The tloga theorem

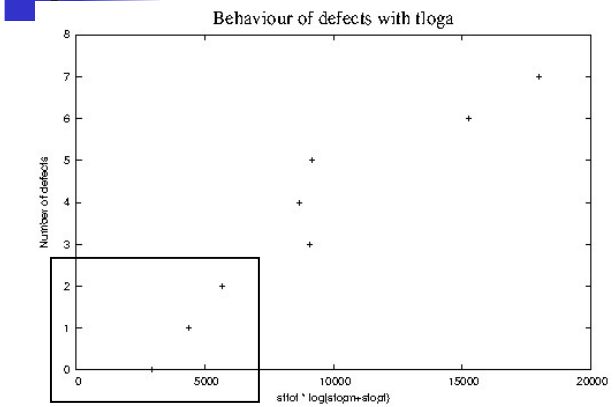


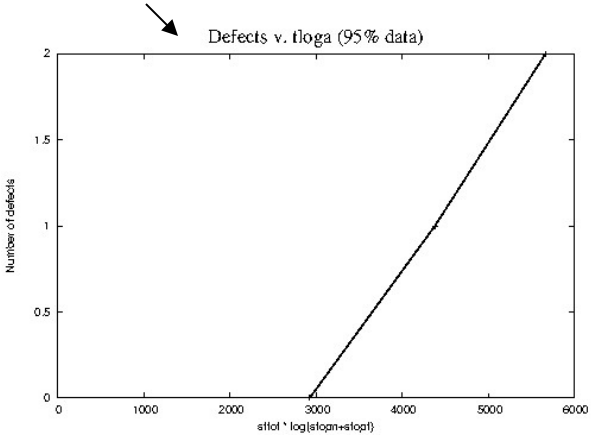
- Predicts the following asymptotic defect distribution

$$d_i \approx t_i \log a_i$$

Can we test this ?

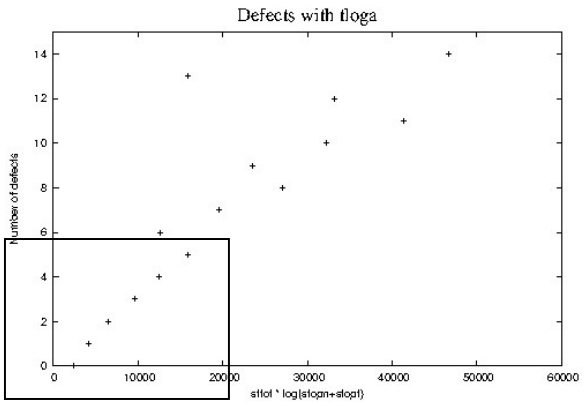
The tloga theorem on software systems

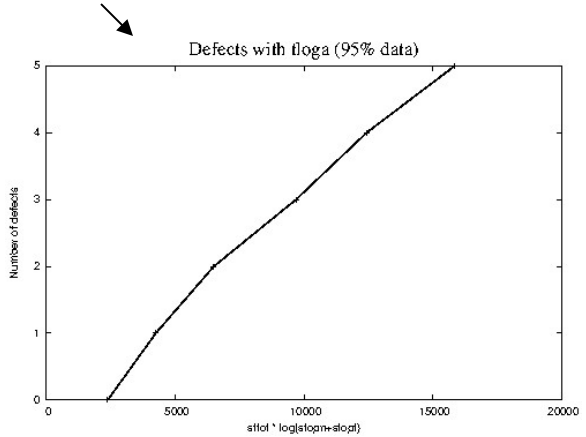




NAG Fortran Library

95%



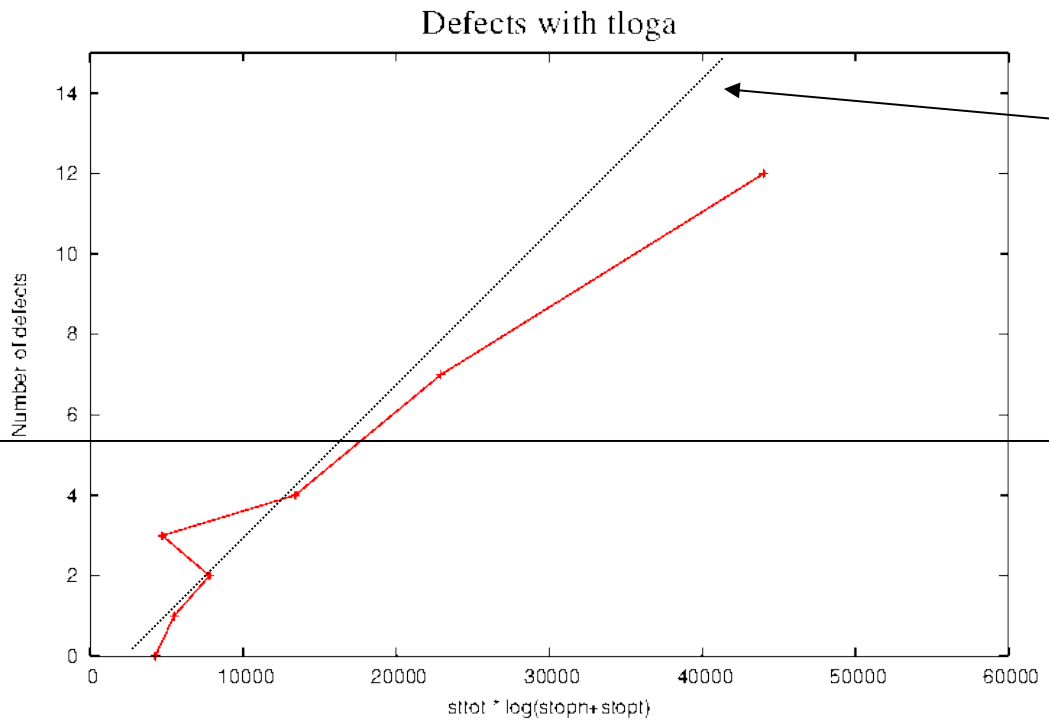


Eclipse IDE (Java)

Equilibriation to tloga in the Eclipse IDE*

data sparse

95% data



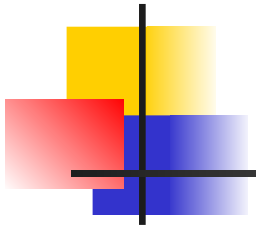
Regression line

*With grateful thanks to Andreas Zeller et. al. (2007) for extracting the defect data and making it openly available. http://www.st.cs.uni-sb.de/softevo. The data comes from releases 2.0,2.1 and 3.0. There are 10,613 components in the release 3.0.

Where next with the genome ?

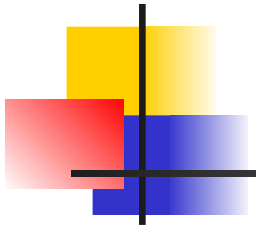
- Alphabet extension and methylation
  - *Conservation of information has some interesting implications which are currently being checked*
- Defects
  - *The tloga theorem implies that gene error rate is basically linearly proportional to gene length slightly modulated by methylation. This is also currently being checked.*

Overview



- Defects in software systems
- A hidden clockwork
- Analogues with the genome
- Defects and the genome
- Conclusions

Conclusions



Discrete tokens, (ACTG, language elements)

Evolution (Natural Selection or selection of the useful)

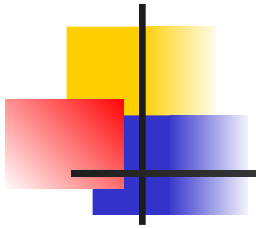
Conservation of information

Conclusions

$$p_i \sim (a_i)^{-\beta} \qquad d_i \sim t_i \log a_i$$

- Software component distributions appear to evolve the same way however we build them and whatever they do guided by the unseen hand of information conservation. (Equivalent to energy conservation in physical systems.)
- The genome also appears to conserve information leading (inter alia) to constant average gene length in kingdoms
- The smallest components of software systems are distributed similarly to genes because they have a nearly constant alphabet
- From an information point of view, the evolution of the gene and the evolution of software systems appear very similar.

References



My writing site:-

http://www.leshatton.org/

Specifically,

http://arxiv.org/abs/1207.5027

http://arxiv.org/1209.1652

For comments on reproducibility in computational science,

http://www.nature.com/nature/journal/v482/n7386/full/nature10836.html

Thanks for your attention.