Preserving Software: challenges and opportunities for reproductibility of Science

Roberto Di Cosmo
INRIA and IRIF

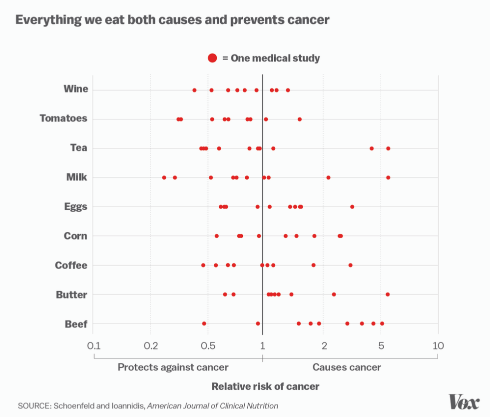
roberto@dicosmo.org

September 16, 2016

Software Heritage

What causes cancer?

Is everything we eat associated with cancer?
Schoenfeld and Ioannidis, *Amer. Jour. of Clinical Nutrition*, 2013.



Inconsistency

an incompatibility between two propositions that cannot both be true

Corrupt data!

Gene name errors are widespread in the scientific literature
Ziemann, Eren and El-Osta, *Genome Biology*, 2016.

Gene name errors are widespread in the scientific literature

Mark Ziemann, Yotam Eren and Assam El-Osta

Genome Biology 2016 **17**:177 | **DOI:** 10.1186/s13059-016-1044-7 | © The Author(s). 2016
Published: 23 August 2016

Abstract

The spreadsheet software Microsoft Excel, when used with default settings, is known to convert gene names to dates and floating-point numbers. A programmatic scan of leading genomics journals reveals that approximately one-fifth of papers with supplementary Excel gene lists contain erroneous gene name conversions.

Keywords

Microsoft Excel – Gene symbol – Supplementary data

The problem of Excel software (Microsoft Corp., Redmond, WA, USA) inadvertently converting gene symbols to dates and floating-point numbers was originally described in 2004 [1]. For example, gene symbols such as *SEPT2* (Septin 2) and *MARCH1* [Membrane-Associated Ring Finger (C3HC4) 1, E3 Ubiquitin Protein Ligase] are converted by default to '2-Sep' and '1-Mar', respectively. Furthermore, RIKEN identifiers were described to be automatically converted to floating point numbers (i.e. from accession '2310009E13' to '2.31E+13'). Since that report, we have uncovered further

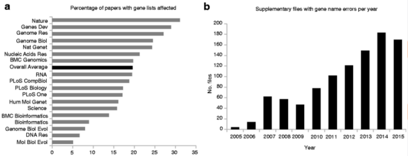


Fig. 1
Prevalence of gene name errors in supplementary Excel files. **a** Percentage of published papers with supplementary gene lists in Excel files affected by gene name errors. **b** Increase in gene name errors by year

Corruption

The process by which a computer database or program becomes debased by alteration or the introduction of errors

Doctored data?

Two Hundred Million Dollar Scientific Grant Fraud Case against Duke University

September 3, 2016 ⁝ National

Federal Prosecutors have launched a gigantic fraud case against Duke University, North Carolina, accusing Duke University of embezzling $200 million in federal research grants, by presenting doctored data with their grant applications. – On a Friday in March 2013, a researcher working in the lab of a prominent pulmonary scientist at Duke University in Durham, North Carolina, was arrested on charges of embezzlement. The researcher, biologist Erin Potts-Kant, later pled guilty to siphoning more than $25,000 from the Duke University Health System, buying merchandise from Amazon, Walmart, and Target—even faking receipts to legitimize her purchases. A state judge ultimately levied a fine, and sentenced her to probation and community service. Then Potts-Kant's troubles got worse. Read the rest here 13:03

Fraud

wrongful or criminal deception intended to result in financial or personal gain

What are drugs good for?

FIGURE 1 | Analysis of the reproducibility of published data in 67 in-house projects.

FROM THE FOLLOWING ARTICLE:

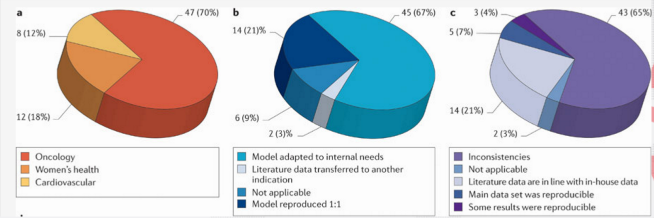
Believe it or not: how much can we rely on published data on potential drug targets?

Florian Prinz, Thomas Schlange & Khusru Asadullah

Nature Reviews Drug Discovery **10**, 712 (September 2011)

doi:10.1038/nrd3439-c1

‹ Back to article | ‹ Back to figures and tables

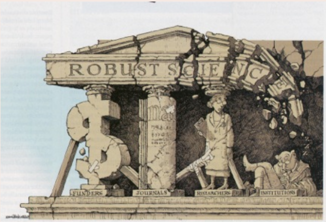


Non reproducibile results

… and this is just one of the worrying replication studies!

We face a science crisis

Our temple of science is crumbling



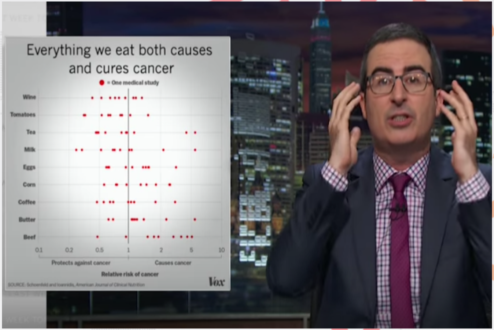
- inconsistencies
- data corruption
- fraud
- non reproducible findings...

(picture from Nature, Sep. 2015)

"Sub-prime science"? (Nicholas Humprey)

The world starts noticing

October 2013



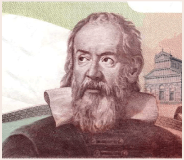
John Oliver, *Science* May 2016

Time to go back to the basics!

what is *science*?

How we built our scientific knowledge

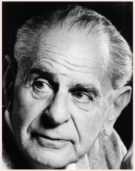
The experimental method



- make an *observation*
- formulate an *hypothesis*
- set up an experiment
- formulate a *theory*

And then we reproduce and verify.

Reproducibility is the key



non-reproducible single occurrences are of no significance to science

Karl Popper, The Logic of Scientific Discovery, 1934

Reproducibility, today

Reproducibility (Wikipedia)

the ability of an entire experiment or study to be *reproduced*, either by the researcher or *by someone else working independently*.
It is one of the main principles of the scientific method.

Why we want it

- foundation of the scientific method
- accelerator of research: allows to build upon previous work
- visibility: reproducible results are cited more often
- transparency of results eases acceptance
- necessary for industrial transfer

reproducibility is *the essence* of *industry*!

Reproducibility in the digital age

For an experiment involving software, we need

- open access to the scientific article describing it
- open data sets used in the experiment
- source code of all the components
- environment of execution
- stable references between all this

Remark

The first two items are already widely discussed!

… what about *software*?

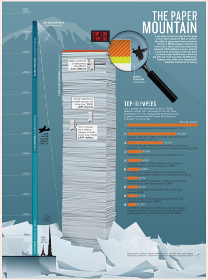
Software is Knowledge

Software is *an essential component* of modern scientific research

Top 100 papers (Nature, October 2014)

> *[...] the vast majority describe experimental methods or sofware that have become essential in their fields.*

`http://www.nature.com/news/the-top-100-papers-1.16224`



Software and reproducibility

A fundamental question

How are we doing, regarding reproducibility, in *Software*?

The case of Computer Systems Research

A field with Computer experts … we have high expectations!
Christian Collberg set out to check them.

Measuring Reproducibility in Computer Systems Research

Long and detailed technical report, March 2014
`http://reproducibility.cs.arizona.edu/v1/tr.pdf`

Analysis of 613 papers

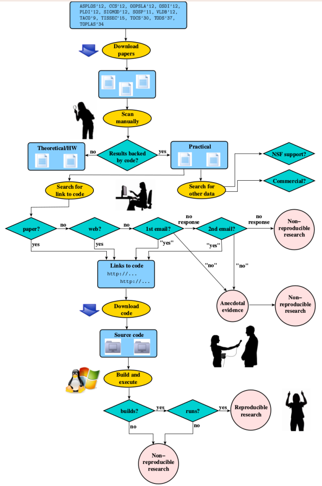
- 8 ACM conferences: ASPLOS'12, CCS'12, OOPSLA'12, OSDI'12, PLDI'12, SIGMOD'12, SOSP'11, VLDB'12
- 5 journals: TACO'9, TISSEC'15, TOCS'30, TODS'37, TOPLAS'34

all very practical oriented

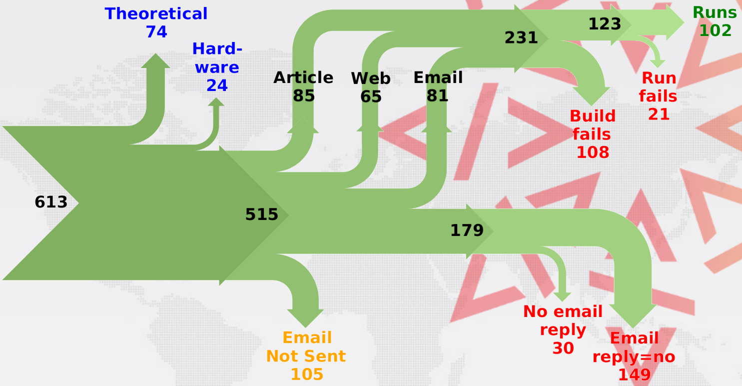
The basic question

can we get the code to build and run?

The workflow



The result



This can be debated (see `http://cs.brown.edu/~sk/Memos/Examining-Reproducibility/`), but…

… that's a whopping 81% of non reproducible works!

In Software Engineering

Even higher expectations, and yet similarly disappointing results
`http://fr.slideshare.net/carloghezzi18/icse-2009-keynote-15919951`

Reference journal

ACM Transactions on Software Engineering and Methodology (TOSEM)

- analysis by Carlo Ghezzi, in 2009, of TOSEM from 2001 to 2006
- 60% of papers refer to a tool
- 20% only are *installable*

Reference conference

International Conference on Software Engineering (ICSE)

- analysis by Zannier, Melrik, Maurer 2006
- complete absence of replication studies

Pressure to make research code available is now raising

Evaluation of software artefacts (optional)

- tools are usable, in line with expectations
- started as a contest in 2011 (ESEC/FSE) (winner *Vouillon and Di Cosmo*)
- now going mainstream: POPL'17, POPL'16, ECOOP'16, OOPSLA'16, CGO'16, VISSOFT'16, PLDI'16, CGO'15, PPoPP'15, VISSOFT'15, ISSTA'15, OOPSLA'15, PLDI'15, POPL'15, CAV'15, ECOOP'15, FSE'15, ISSTA'14, OOPSLA'14, PLDI'14, ECOOP'14, FSE'14, SAS'13, OOPSLA'13, ECOOP'13, FSE'13, FSE'11

Use the Source, Luke!

Some people claim that having (all) the source of the code used in an experiment is *not worth the effort* (see “Replicability is not Reproducibility: Nor is it Good Science”, Chris Drummond, ICML 2009)

Sure, diversity *is* important, but:

- Source code is like the proof used in a theorem: can we really accept *Fermat statements* like “the details are omitted due to lack of space”?
- modern complex systems makes even the simplest experiment depend on a wealth of components and configuration options
- access to *all* the source code is not just necessary to *reproduce*, it is also useful to *evolve and modify*, to *build new experiments* from the old ones

The reasons (or, “the dog ate my program”)

Why so much software fails to pass the test?

Many issues, nice anecdotes, and it finally boils down to

- *Availability*
- *Traceability*
- Environment
- Automation (do *you* use continuous integration?)
- Documentation
- Understanding (including Open Source)

The first two are important *software preservation issues*

Yes, code is fragile:

it can be destroyed, and we can lose trace of it

Software is fragile

like all digital information, FOSS is fragile

- inconsiderate and/or malicious code loss (e.g., Code Spaces)
- business-driven code loss (e.g., Gitorious, Google Code)
- for obsolete code: physical media decay (data rot)

If a website disappears you go to the Internet Archive…

… where do you go if (a repository on) GitHub goes away?

Software is spread all around

Fashion victims

- many disparate development platforms
- a myriad places where distribution may happen
- projects tend to migrate from one place to the other over time

One place to bind them…

… where can we find, track and search *all* the source code?

Web links *are not* permanent (even *permalinks*)

there is no general guarantee that a URL... which at one time points to a given object continues to do so

T. Berners-Lee et al. Uniform Resource Locators. RFC 1738.

404

URLs used in articles *decay*!

Analysis of *IEEE Computer* (Computer), and the *Communications of the ACM* (CACM): 1995-1999

- the *half-life* of a referenced URL *is approximately 4 years* from its publication date.

D. Spinellis. The Decay and Failures of URL References. Communications of the ACM, 46(1):71-77, January 2003.

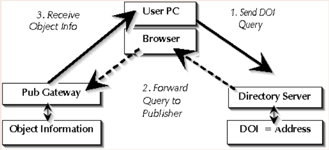
Similar findings in Lawrence, S. et al. *Persistence of Web References in Scientific Research*, IEEE Computer, 34(2), pp. 26–31, 2001.

The Digital Object Identifier (DOI)

Example: doi:10.1109/MSR.2015.10

- to find what 10.1109/MSR.2015.10 is, go to a *resolver* (e.g. doi.org)
- this returns `http://ieeexplore.ieee.org/document/7180064/`
- at this URL we find …

Architecture of the DOI infrastructure



- DOI resolution *can change*
- content at URL *can change*
- no *intrinsic* way of noticing
- persistence based on *good will*

We are at a turning point

Software is

- an *essential component* of modern scientific research
- a *key mediator* for accessing *all* information
- at the heart of our society (communication, entertainment, administration, finance, health, energy, transportation, education, research, politics)

In a word

Software embodies our collective Knowledge and Cultural Heritage

And yet… we are loosing, and/or loosing trace of it…

It's time to take action!

The Software Heritage Project

Software Heritage
PRESERVING TECHNICAL KNOWLEDGE

Our mission

Collect, organise, preserve and share the *source code* of *all the software* that lies at the heart of our culture and our society.

Past, present and future

Preserving the past, *enhancing* the present, *preparing* the future.

Software Source Code is *different*

“Programs must be written for people to read, and only incidentally for machines to execute.” Harold Abelson, Structure and Interpretation of Computer Programs

Distinguishing features

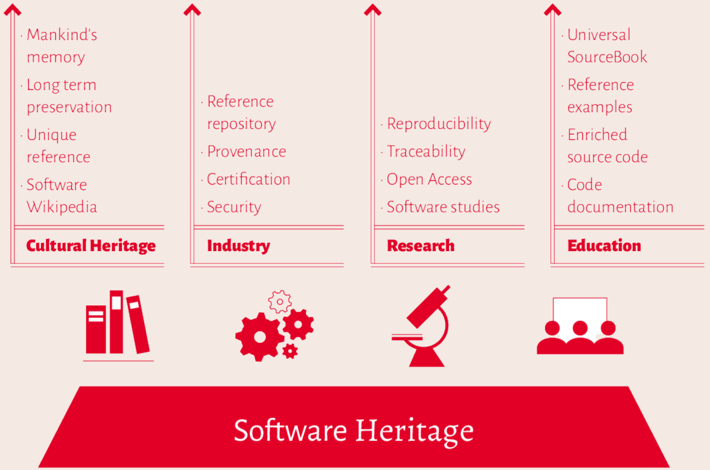
- *executable* and *human readable* knowledge (an *all time new*)
    - even hardware is… software! (VHDL, FPGA, …)
    - *text files are forever*
- naturally *evolves* over time
    - the *development history* is key to its *understanding*
- complex: large *web of dependencies*, millions of SLOCs

In a word

- software *is not just another* sequence of bits
- a software archive *is not just another* digital archive

We are working on the foundations

one infrastructure to build them all



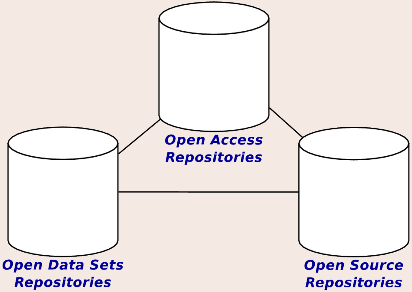
Supporting more accessible and reproducible science

A global library referencing all software used in all research fields

- completes the infrastructure for Open Access in science
- provides intrinsic persistent identifiers needed for scientific reproducibility
- enables large scale, verifiable software studies

The Knowledge Conservancy Magic Triangle

The Knowledge Conservancy Magic Triangle



Legenda (links are important!)

- articles: ArXiv, HAL, ...
- data: Zenodo, ...
- software: *Software Heritage* to the rescue

Free and Open Source Software is crucial

D. Rosenthal, EUDAT, 9/2014

you have to do [digital preservation] with open-source software; closed-source preservation has the same fatal "just trust me" aspect that closed-source encryption suffer from.

design decision

Software Heritage will:

- provide *full details* on its architecture
- make available *all the source code* used
- use *open standards*
- encourage a *collaborative* development process
- unleash and leverage *the power of the community*

Replication is the key

Thomas Jefferson, February 18, 1791

... let us save what remains: not by vaults and locks which fence them from the public eye and use in consigning them to the waste of time, but by such a multiplication of copies, as shall place them beyond the reach of accident.

design decision

Software Heritage will:

- provide easy means for making copies
- encourage the growth of a mirror network
  - using *a variety* of technologies
  - spanning *multiple* continents
  - under *diverse* control structures
    - no single decisional point of failure!
      (remember Google code, Gitorious, ...)

Why us? Because the Source Code is our DNA!

it is at the heart of our work

- we *write* software
- we *read and reuse* software
- we *distribute* software
- we *understand* how software works

Bottomline

it is our *duty* and our *privilege* to take care of Software preservation

The people

The core team

- Roberto Di Cosmo
- Stefano Zacchiroli
- Nicolas Dandrimont (Engineer)
- Antoine Dumont (Engineer)
- and *Jordi, Quentin and Guillaume*

Scientific advisors

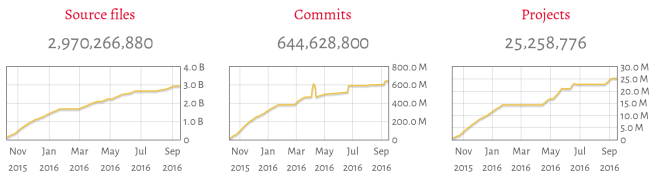
- Serge Abiteboul (French Sience Academy)
- Jean-François Abramatic (former W3C director)
- Gerard Berry (Gold Medal, French Science Academy)
- Julia Lawall (Coccinelle, Linux Kernel, Outreachy)

The archive

Our sources

- GitHub — all public repositories, as of April 2016
- Debian — daily snapshots of all suites since 2005–2015
- GNU — all historical releases up to August 2015
- Gitorious — retrieved full mirror from Archive Team
- Google Code — retrieved full mirror from Google

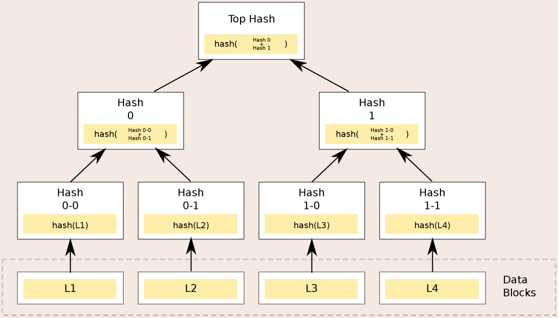
Some numbers



The *richest* source code *graph* already, … and growing daily!

The archive in a few pictures

Merkle tree (R. C. Merkle, Crypto 1979)



Combination of

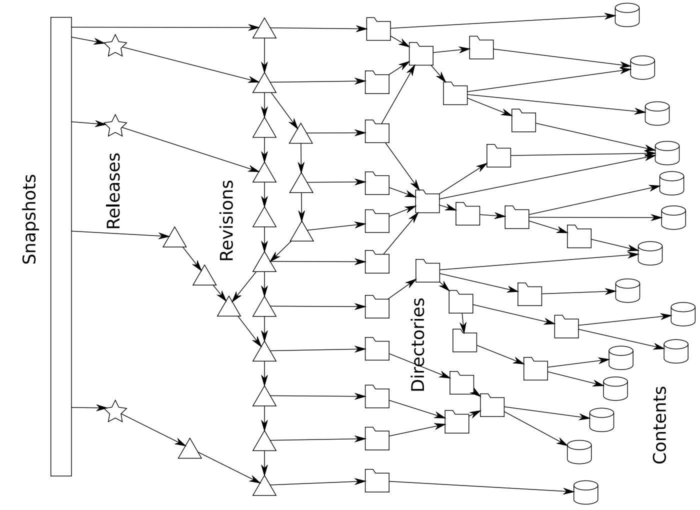
- tree
- hash function

Classical cryptographic construction

- fast, parallel signature of large data structures
- widely used in *git*, *bitcoin*, etc.

The archive in a few pictures

A giant (extended) Merkle DAG



The road ahead

Planned features...

- *lookup* by hashes for contents (done)
- *provenance information* for all the content
- *browsing*: wayback machine for software source code
- *full text search*: dive into the Software Heritage archive
- *download*: git clone from Software Heritage

... and many more one could imagine

all the world's software development history in a single graph!

that makes a 5TB database already...

Making it happen

Inria as initiator

- funds the *bootstrap phase* of Software Heritage
- an agreement with UNESCO is coming soon!

Support and first partners

ACM, Bell Labs, Creative Commons, DANS, Eclipse, Engineering, FSF, OSI, GitHub, GitLab, IEEE, Informatics Europe, Microsoft, OIN, OW2, SIF, SFC, SFLC, The Document Foundation, The Linux Foundation, ...

Going global

building an *open, multistakeholder, nonprofit* organisation

Come in, we're open!

The road to success

- *adoption* : get users *today* to ensure preservation *tomorrow*
- *collaboration* : prepare the path for *everybody* to participate
- *legitimacy* : *one* shared infrastructure, *not* dozens of "me toos"

Everybody is needed!

researchers	*many* scientific challenges (please ask!)
developers	Software Heritage is itself Open Source!
transversal	find the many source code repositories
partners	contribute to the effort

Conclusion

Software Heritage is

- a revolutionary *reference archive* of *all* software ever written
- a fantastic new tool for *research* software
- an international, open, nonprofit, *mutualized infrastructure*
- at the service of our community, at the service of society!

Now open

`www.softwareheritage.org` - *sponsoring, partnerships*
`wiki.softwareheritage.org` - *working groups, leads*
`forge.softwareheritage.org` - *our own code*

Questions?

Metadata alignment

Many concepts related to source code

- project, archive, source, language, licence, bts, mailing list, …
- developer, committer, author, architect, …

Many existing ontologies, catalogs

- DOAP, FOAF, Appstream, schema.org, ADMS.SW, …
- Freecode (40.000+), Plume (400+), Debian (25.000+), FramaSoft (1500+), OpenHub (670.000+), …

Challenge : scale up metadata to millions of projects

- *reconcile* existing ontologies
- *link* and *check* existing catalogs with Software Heritage
- handle *inconsistent data* and *provenance information*
- synthesise missing information (machine learning)

Software phylogenetics

The Software Diaspora

- Code often *migrates* across projects : forks, copy-paste
- Code gets *cloned* : reuse, language limitations, code smells
- Projects *migrate* across forges : fashion, functionality
- Projects get *cloned* : mirrors, packages

Challenge: tracing software evolution across billions of files

- rebuild the history of software artefacts
- identify code origins
- spot code clones
- build project impact graphs

Distributed infrastructure

The software graph

- files
- directories
- commits
- projects

all de-duplicated in Software Heritage

Challenge: design efficient architectures and algorithms

- replication and availability
- navigation
- what happens to CAP? (updates are nondestructive!)
- query

Code search: an old problem

A natural need

- Find the definition of a function/class/procedure/type/structure
- Search examples of code usage in an archive of source code
- you name it...

A natural approach

- Regular expressions

We have all used *grep* since the 1970's!

where is the challenge?

Finding a needle in a haystack: size matters!

How do we search in *millions* of source code files?

Google code search (open 2006, closed 2011)

see `https://swtch.com/~rsc/regexp/regexp4.html`
reborn in 2013 for Debian `http://sources.debian.net/`

how

- build an inverted index of *trigrams* from all source files
- *map* regexps to trigrams
- *filter* files that may match
- run *grep* on each file (using the cloud)

performance

scaled reasonably well up to *1 billion lines of codes*

Challenge: scaling up code search

What about *all the source code* in the world?

Software Heritage is *two orders of magnitude* bigger already

- over *two billion* unique source files
- *hundreds* of billions of LOCs

We need new insight for handling this.

Beyond regular expressions?

Advanced code search requires

- language specific *patterns*
- working on *abstract syntax trees*

Regular expressions are a nice *swiss-army knife* approximation, can we build a specific tool that scales?

Software as Big Data

Remember the numbers

- 21 million repositories ingested (10M next in line)
- 500 million commits
- 2.5 billion unique source files / 200 TB of raw source code

and growing by the day!

Challenge: what can machines learn here?

- programming patterns
- developer skills
- vulnerabilities
- bugs and fixes