Knowledge Flows in Open Source Software Supply Chains

Audris Mockus
University of Tennessee
audris@utk.edu

[2017-11-03 Fri]

Outline

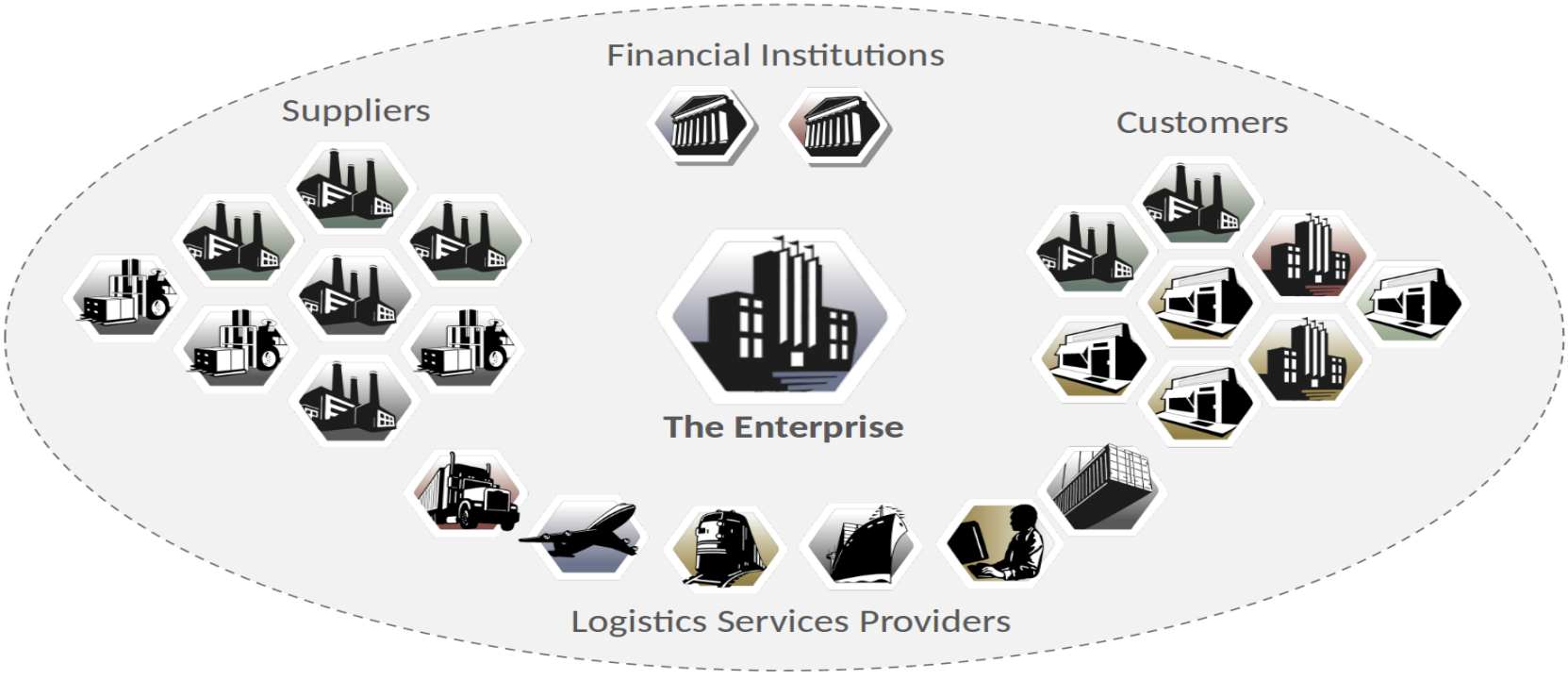
Ad

Because it is a supply chain

- 23 years at Bell Labs/Avaya Labs
- 3 at the University of Tennessee
- Data Science/Big data/Software Engineering
- Looking for interested PhD students
- Please contact at audris@utk.edu

Definition (What is Supply Chain)

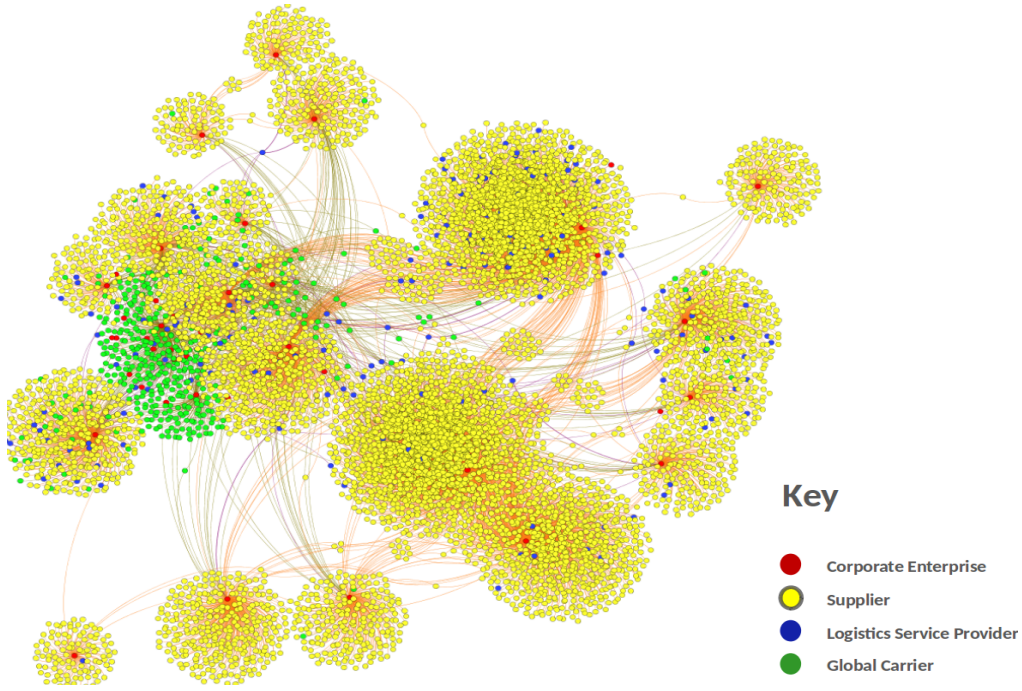
is a set of three or more companies directly linked by one or more of the upstream or downstream flows of products, services, finance, and information from a source to a customer



Graphics courtesy of Diane Palmquist and Greg Kefer, GT Nexus, UT Supply Chain Forum, November 12, 2014

Definition (What is Supply Chain)

is a set of three or more companies directly linked by one or more of the upstream or downstream flows of products, services, finance, and information from a source to a customer



Graphics courtesy of Diane Palmquist and Greg Kefer, GT Nexus, UT Supply Chain Forum, November 12, 2014

Definition (What is Software Supply Chain)

is a supply chain with individual developers and groups (software projects or packages) representing "companies" producing new versions of the source code (e.g., files, modules, frameworks, or entire distributions).
The upstream and downstream flow from projects to end users is represented by the dependencies and sharing of the source code and by the contributions via patches, issues, and exchange of information.

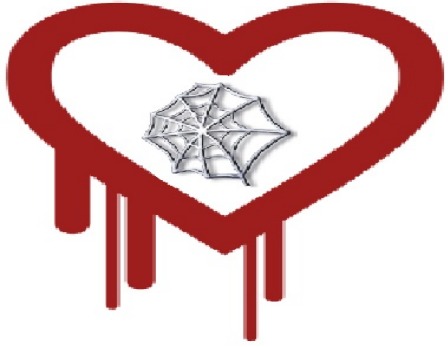
Definition (What are Knowledge Flows)

Transfer of tacit and explicit information, culture, behaviors, customs, or values among individuals or groups that occurs as a result of activities that are based on shared or interdependent artifacts, interests, or objectives [1]. It is not simply an exchange of explicit information as on, for example, StackExchange.

Why think of FLOSS as a supply chain?

Because it is a supply chain

- Distributed authority/control
  - A lack of understanding and visibility
- Need for risk management
  - Companies may withdraw support, developers may move on
  - See "Roads and Bridges" report from Ford foundation
  - E.g. Heartbleed: single developer for all e-commerce



Impact of Supply Chain Visibility

- Inventory savings of 20% of value
- Increased forecast accuracy of about 25%
- Improved SLAs to consistent 98% levels
- Freight charge reductions from 5% to 3.5% of volume
- Decrease of inventory on stock from just over 10 days to fewer than 7 days
- Reduction in workforce by 10%

Source: Why Supply Chain Leaders Should Aim for End-to-End Supply Chain Visibility by 2016, November 1, 2013

SC to KF map

Supply Chain

- Nodes: customers, enterprises, suppliers of information, materials, logistics, and financing
- Links: material, information, and financial flows

Knowledge Flows

- Nodes: packages, authors, maintainers
- Links: runtime, build, optional dependencies, author-file/project-induced relationships, mentor follower relationships

Visibility

Information that developers have about the inputs, processes, sources and practices used to bring the product to consumers/market. This includes complete supply chain visibility including traceability for the entire supply chain. Visibility is, generally, inwardly/developer focused.

Transparency

Information that developers share with their consumers about the inputs, processes, sources and practices used to bring the product to the consumer. It is more outwardly focused/from the consumer perspective than visibility.

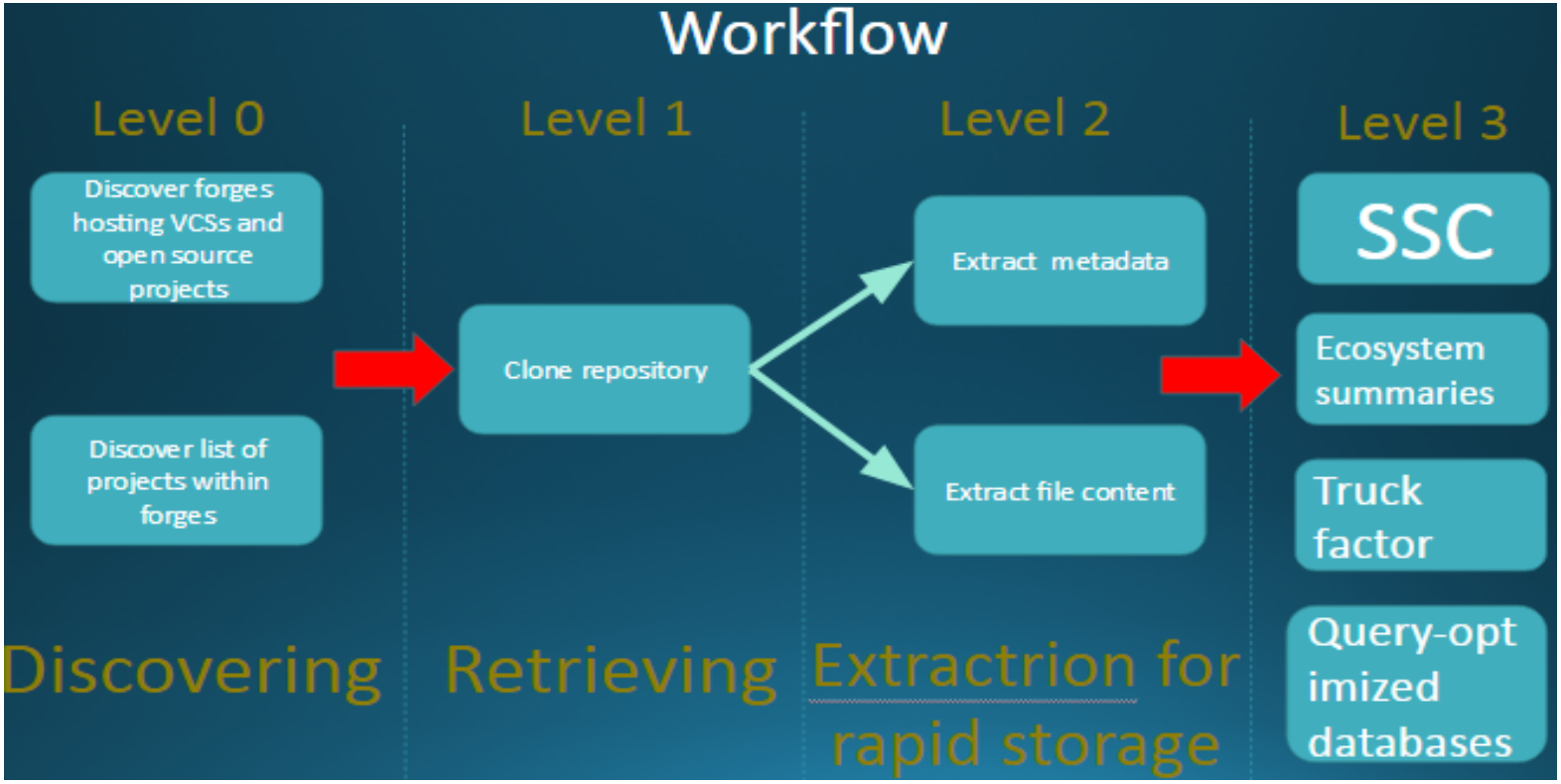
Constructing FLOSS Network

- Get the data
- Store
- Produce accurate networks
- Estimate risks

Data collection/storage

- Get the data
  - VCS - discovery (50+M projects and growing)
  - Package managers (over 30 and growing)
- Store the data
  - Simply cloning ($\geq$\$500TB)
- Constructing networks
  - How to structure of the database?
- Analytics
  - How to do in reasonable time?

Illustration: Data collection workflow



Elements

- git objects: commits, trees, blobs, and tags
- sha1 based on object content

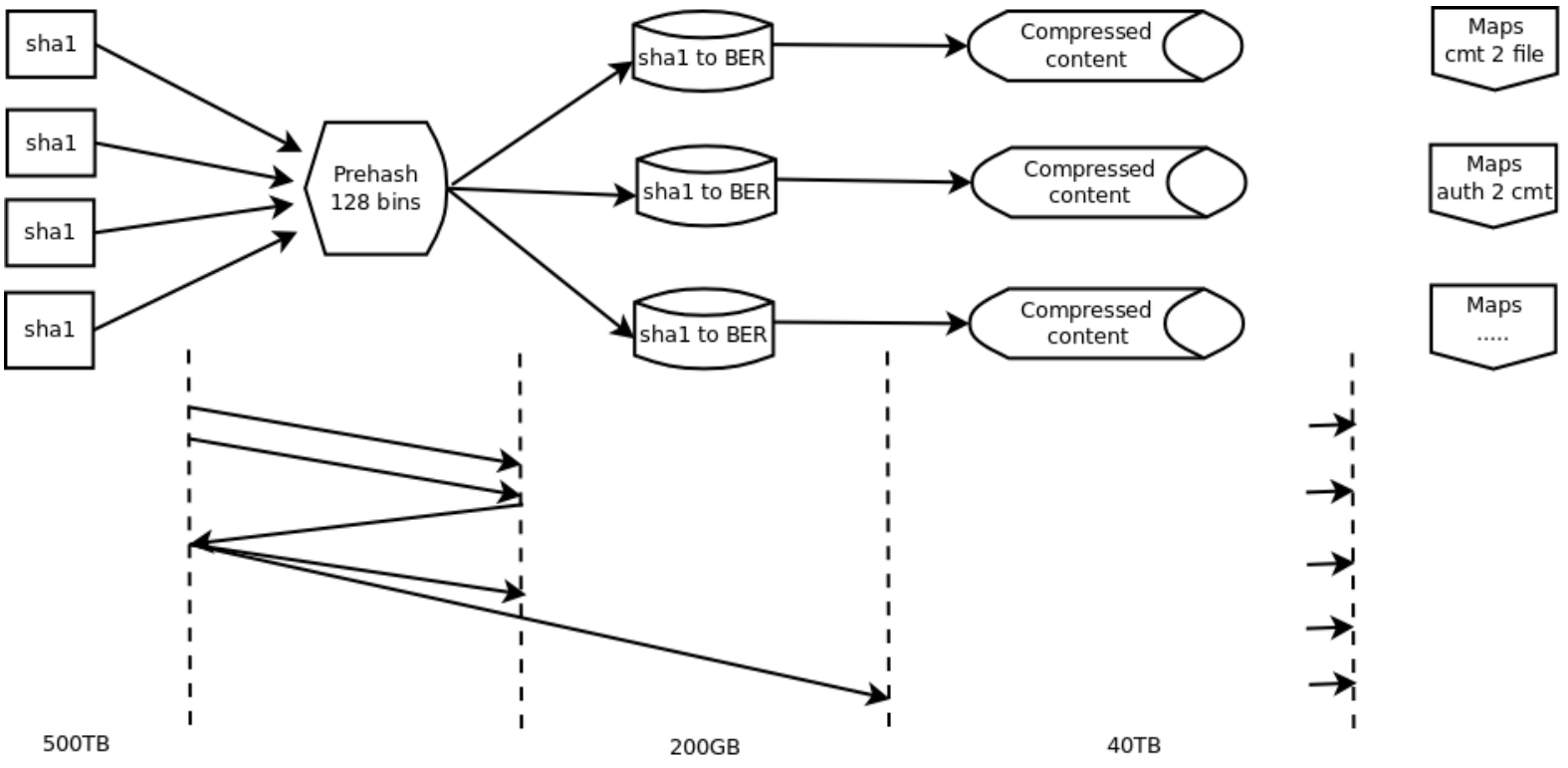
Rapid collection of new data

- Cashing of the object IDs via git sha1
- Pre-hash to 128 TokyoCabinets (200G)
  - maps sha1 to a BER compressed integer

Minimize storage needs

- Store only once (500TB $\rightarrow$ 40TB)
  - Avoids overlap: git clone prA and git clone prB
  - Needs custom git backend to enable updates
  - Store heads for each repo

Database Structure and Updates



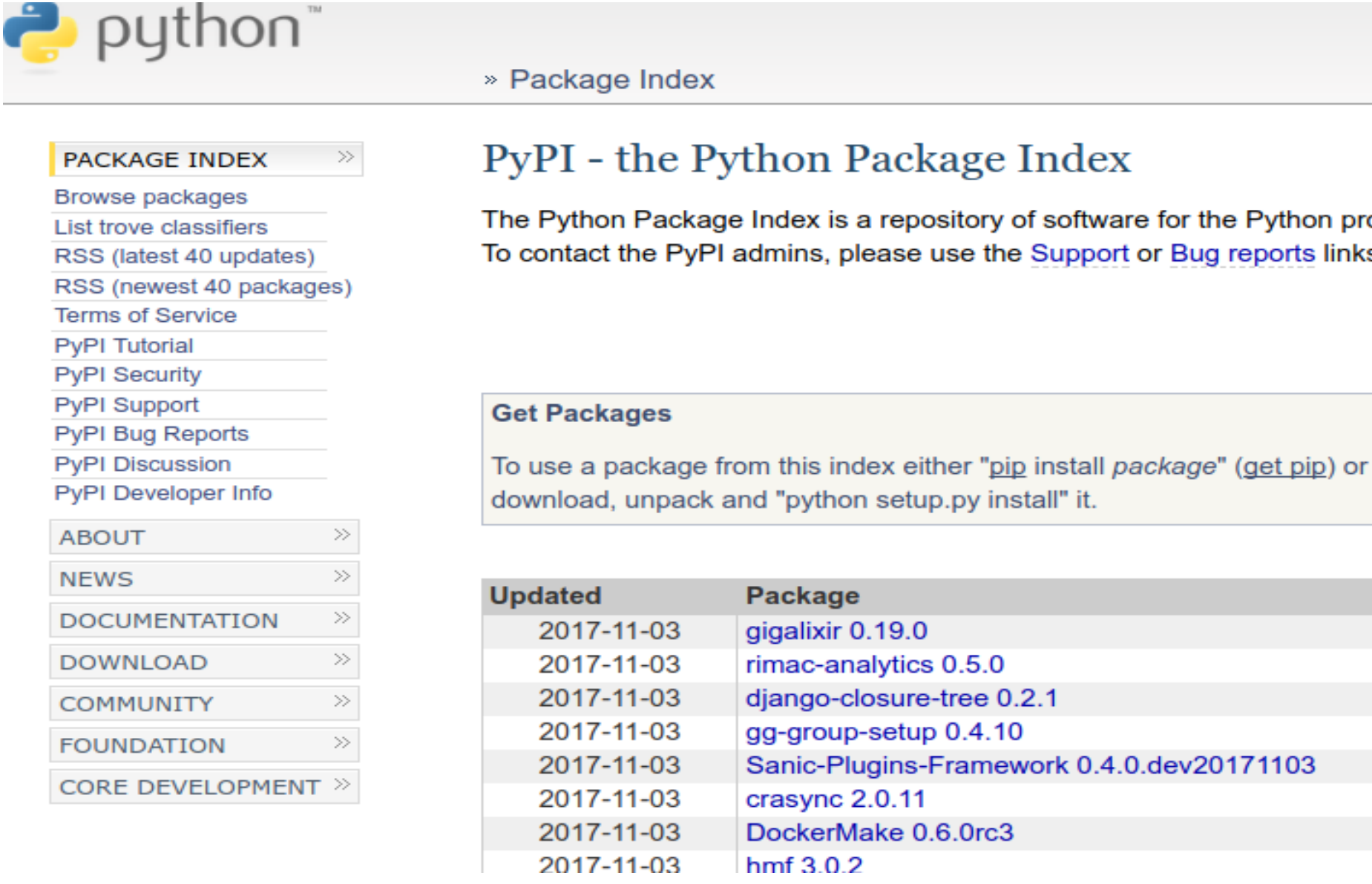
Ability to do fast analysis: Map and Inverse Map

- Commit to parents, created blobs, repository, file names modified, author, date, commit comment
- File name to blob
  - e.g. package.json maps to over 6M blobs
  - NAMAPACE to 120K blobs
  - setup.py to 1M blobs
  - Makefile to 7M blobs
- Text indexing of commit messages, file names, author names, and file content

Lets look at some networks

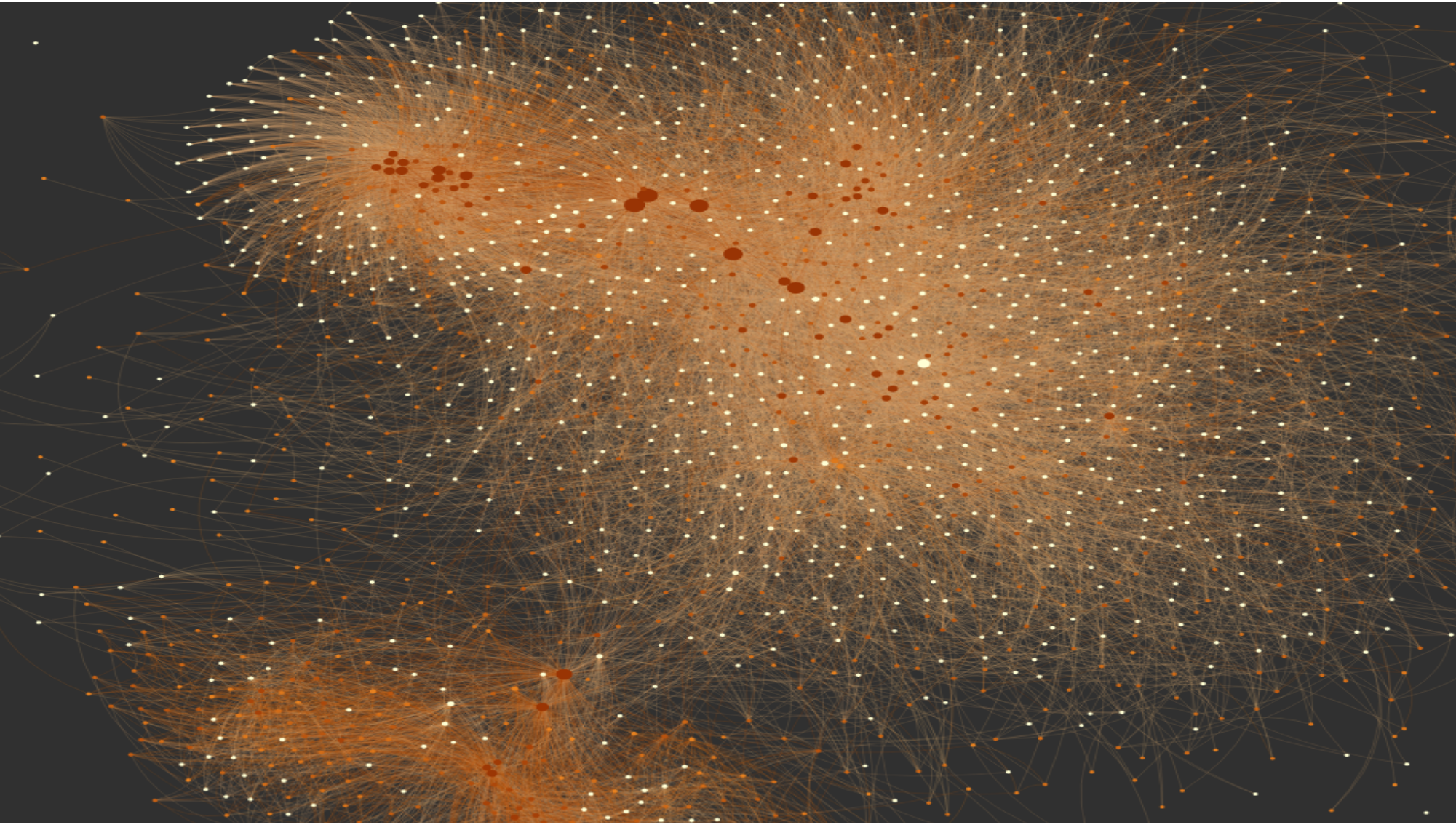
- How do they look?
- What properties they have?
- Do they vary among different supply chains?

PyPi


python
» Package Index
PACKAGE INDEX
Browse packages
List trove classifiers
RSS (latest 40 updates)
RSS (newest 40 packages)
Terms of Service
PyPI Tutorial
PyPI Security
PyPI Support
PyPI Bug Reports
PyPI Discussion
PyPI Developer Info
ABOUT
NEWS
DOCUMENTATION
DOWNLOAD
COMMUNITY
FOUNDATION
CORE DEVELOPMENT
PyPI - the Python Package Index
The Python Package Index is a repository of software for the Python pro
To contact the PyPI admins, please use the Support or Bug reports links
Get Packages
To use a package from this index either "pip install package" (get pip) or
download, unpack and "python setup.py install" it.
Updated
Package
2017-11-03 gigalixir 0.19.0
2017-11-03 rimac-analytics 0.5.0
2017-11-03 django-closure-tree 0.2.1
2017-11-03 gg-group-setup 0.4.10
2017-11-03 Sanic-Plugins-Framework 0.4.0.dev20171103
2017-11-03 crasync 2.0.11
2017-11-03 DockerMake 0.6.0rc3
2017-11-03 hmf 3.0.2

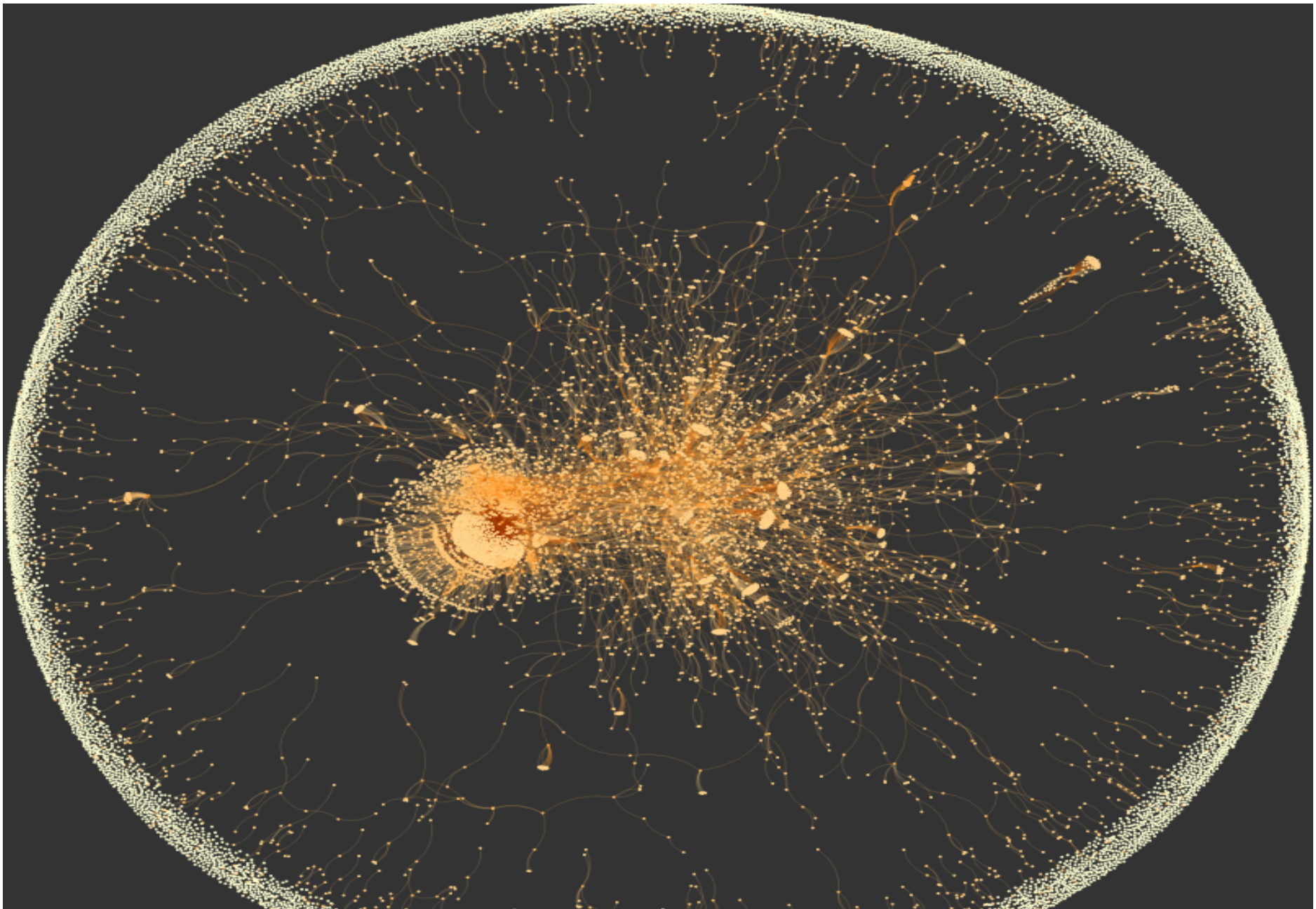
Runtime dependencies: PyPi A

http://kgullikson88.github.io/blog/pypi-analysis.html



Runtime dependencies: PyPi B

http://ogirardot.github.io/meta-deps/



CRAN

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

The Comprehensive R Archive Networ

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these ve of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Li package management system in addition to the link above.

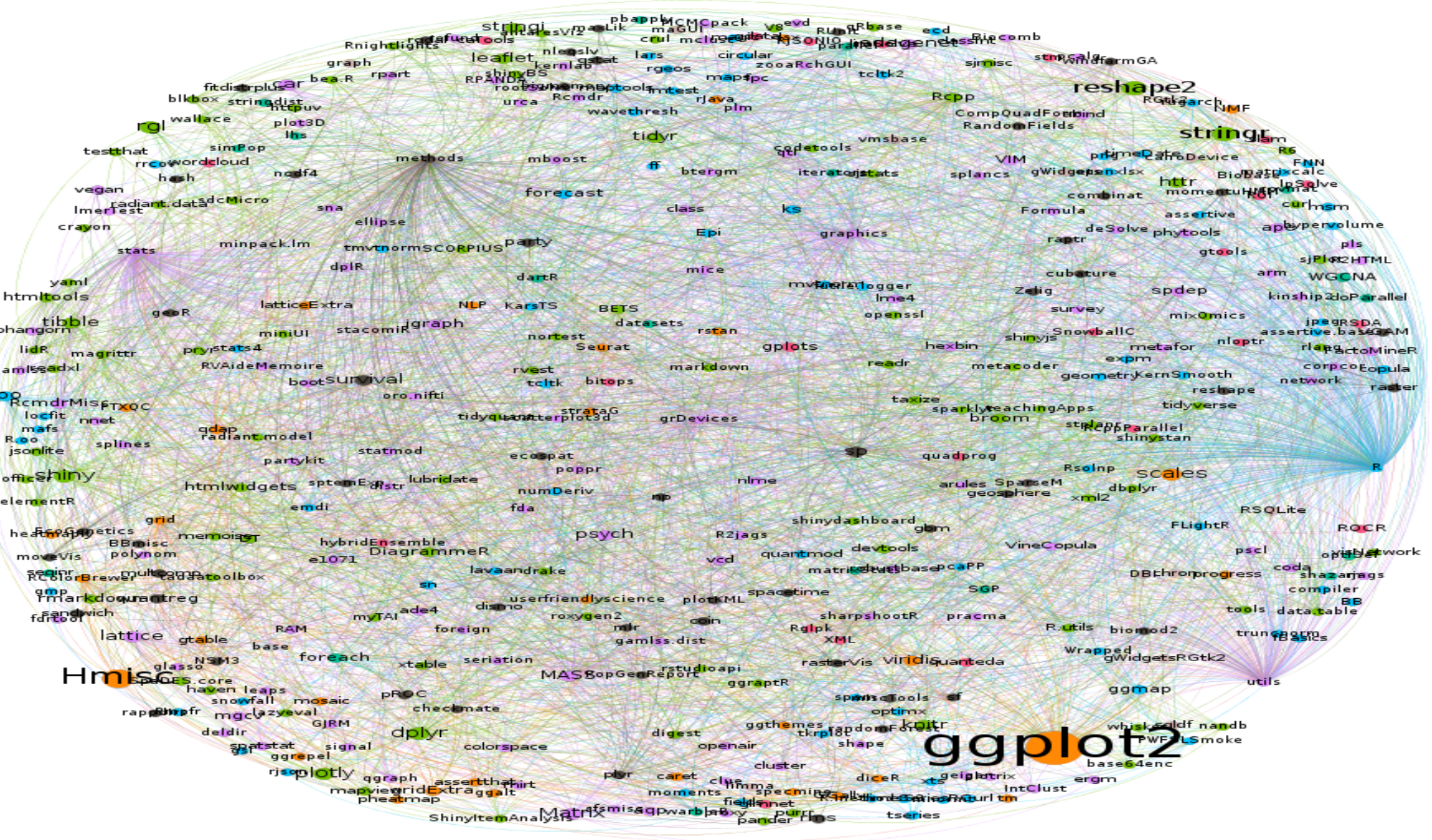
Source Code for all Platforms

Windows and Mac users most likely want to download the precompil binaries listed in the upper box, not the source code. The sources hav compiled before you can use them. If you do not know what this mea probably do not want to do it!

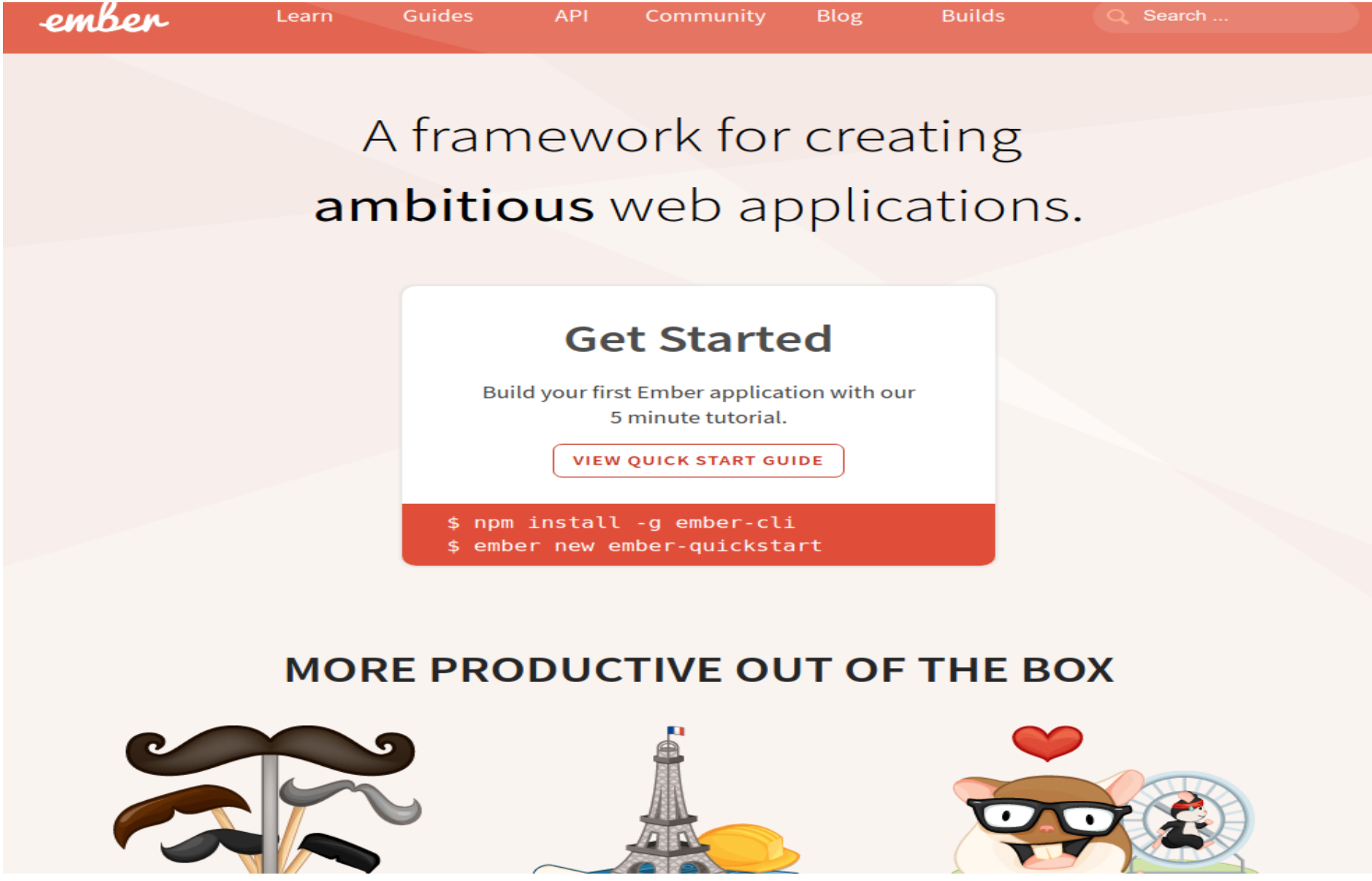
- The latest release (Thursday 2017-09-28, Short Summer) R-3.4.2.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created time periods before a planned release).
- Daily snapshots of current patched and development versions a available here. Please read about new features and bug fixes be filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

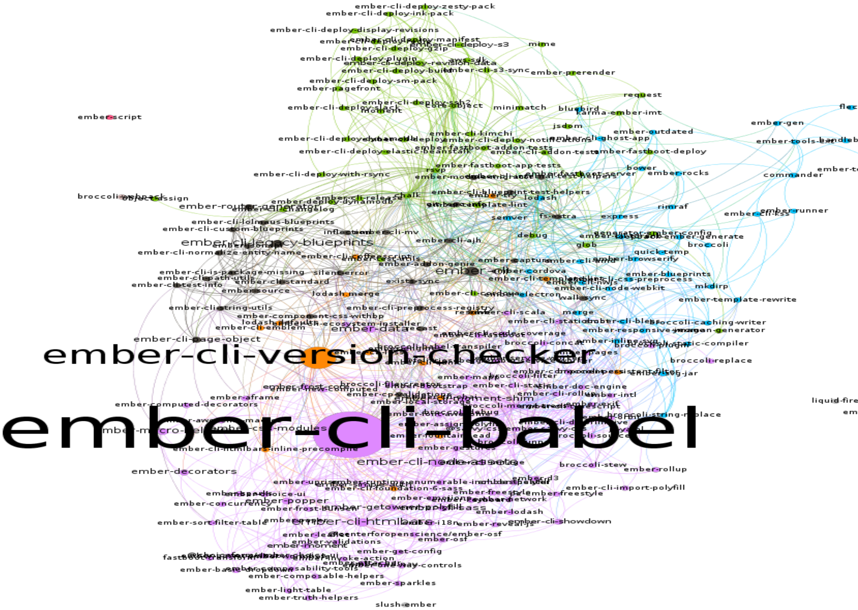
Runtime dependencies: CRAN



ember


ember
Learn
Guides
API
Community
Blog
Builds
Search ...
A framework for creating
ambitious web applications.
Get Started
Build your first Ember application with our
5 minute tutorial.
VIEW QUICK START GUIDE
$ npm install -g ember-cli
$ ember new ember-quickstart
MORE PRODUCTIVE OUT OF THE BOX

Runtime dependencies: ember



Code-reuse-induced dependencies

- Collect all the blobs for a project
- Look for all other projects that contain the same blobs
- Investigate blobs that span many projects
- What are these code reuse patterns?

Code-reuse-induced — emberjs

- Build tools: rake — for Ruby on Rails
- Testing: qunit — testing framework
- Runtime: jQuery – a JavaScript library
- Framework: epf – Emberjs Persistence Foundation

Code-reuse-induced — emberjs

- Prior incarnations: SproutCore/Amber.js - early name for the EmberJS project,
- Hard forks: innoarch/bricks.ui - a hard fork of EmberJS that was developed as a separate project.
- Tutorials cookbooks/nodjs: early code examples
- Package manager: package.json — for npm

Identify boundaries of supply chains

- As a network cluster:
  more links inside the cluster than outside
- Lets call these clusters ecosystems
- How to identify them?
  - Community detection
- How to compare them?
  - Properties of the networks

Clustering/community detection

- Based on:
  - Code reuse network
  - Package manager dependency network
  - Development tool sharing network
  - Mentorship network
- Are these clusters different?
- Implications for risk?

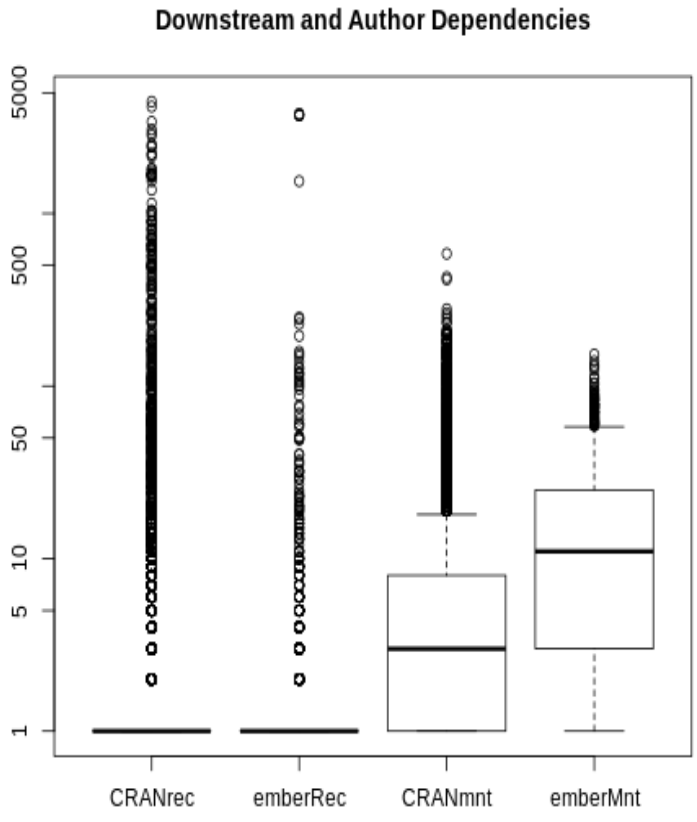
Is CRAN different from emberjs?

- Core package: ember-source, R-base
- npm has 500K+ packages
  - ember-source recursively depends (runtime+dev+optional) on 3,544 npm packages
  - 4,559 packages named ember* recursively depend on 20K npm packages
  - 9,961 authors in the EmberJS ecosystem had participated in 1.1 million FLOSS projects
- CRAN has 13K packages

Compare: CRAN vs emberjs

Downstream/Author

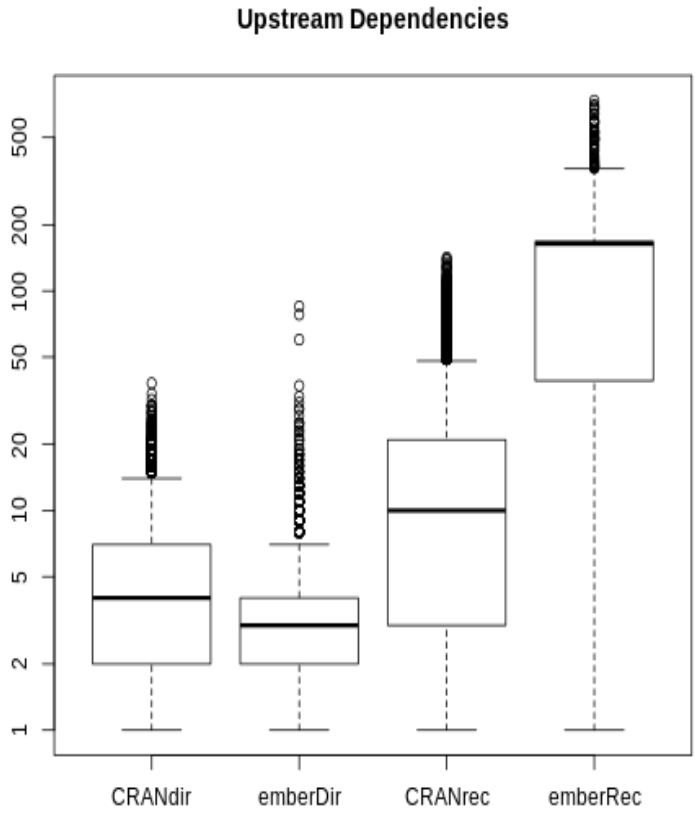
- Only 12% of ember and 25% of CRAN packages have dependents
- ember-cli-babel has 3.7K dependents
- Rcpp has 4.4K dependents
- Fewer packages/mntnr in CRAN



Compare: CRAN vs emberjs

Upstream

- ember packages have few direct (2) and many recursive (200)
- CRAN packages have more direct (3) but few recursive (9)



We constructed/compared the networks

- So what?
  - Estimate risk and
  - Provide recommendations for
    - End User
    - Developer (using components)
    - Contributor (learning/gaining reputation)
    - Organization (strategy)

Types of risk

- License/Regulatory
- Breaking Changes
- Lack of updates
- Corporate Involvement
- Exploits
- Truck factor
- Technology advancements

What are main challenges?

- Operational data are treacherous - unlike experimental data
  - Multiple contexts
  - Missing events
  - Incorrect, filtered, or tampered with
- Continuously changing
  - Systems and practices are evolving
- Challenges measuring or defining accuracy
- Potential for misinterpretation

Network construction challenges

Identity for individuals and packages

- Within VCS
- Among VCS
- Other sources

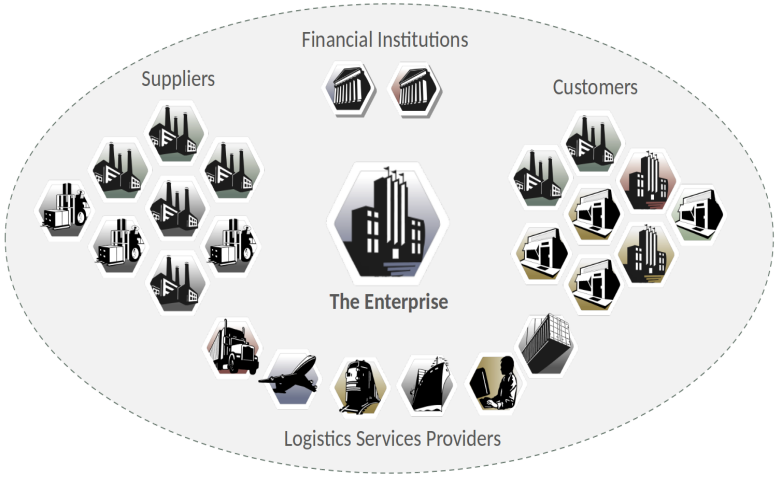
Cross-package-manager dependencies

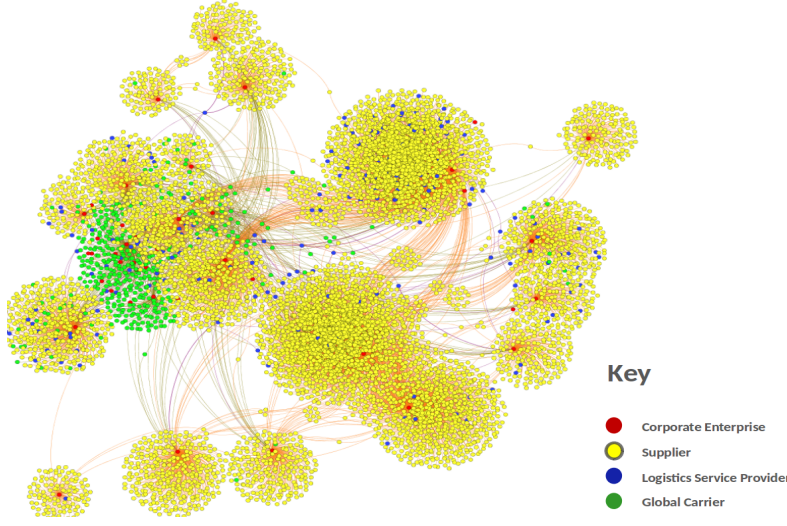
OD: commits in VCS

- Context:
  - Why: merge/push/branch, fix/enhance/license
  - What: e.g, code, documentation, build, binaries
  - Practice: e.g., centralized vs distributed
- Missing: e.g., private VCS, no links to defect
- Incorrect: tangled, incorrect comments, wrong dates, authors
- Filtered: small projects, import from CVS
- Tampered with: `git rebase`

Summary

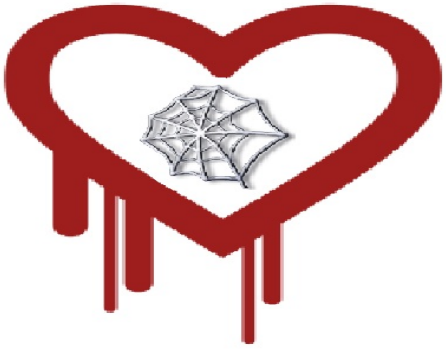
- What are software supply chains?
- Why care about FLOSS supply chains?
- SSCs
  - How to construct
  - Explore
  - Measure
- Operational data challenges





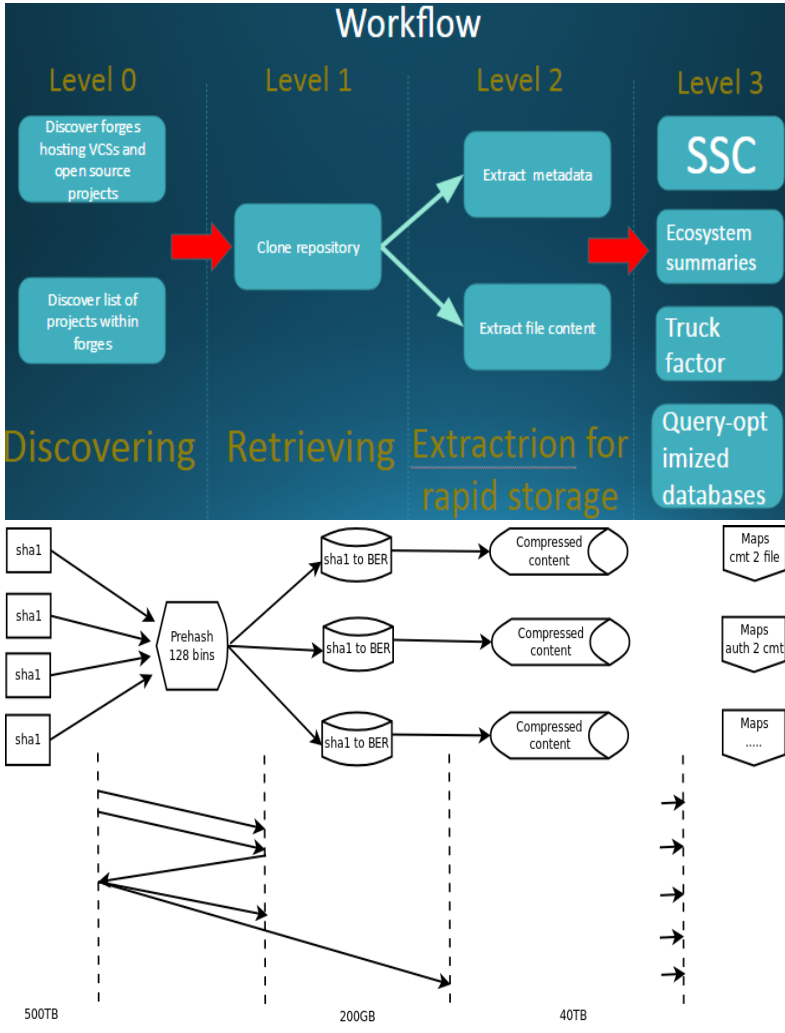
Summary

- What are software supply chains?
- Why care about FLOSS supply chains?
- SSCs
  - How to construct
  - Explore
  - Measure
- Operational data challenges



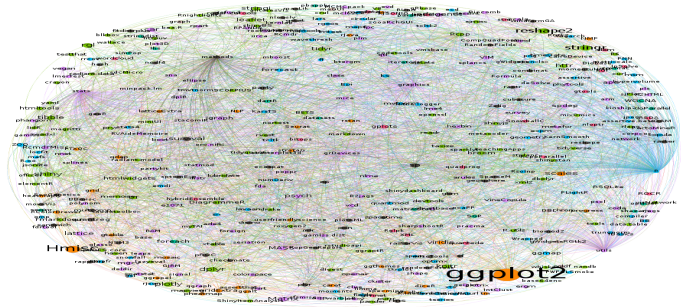
Summary

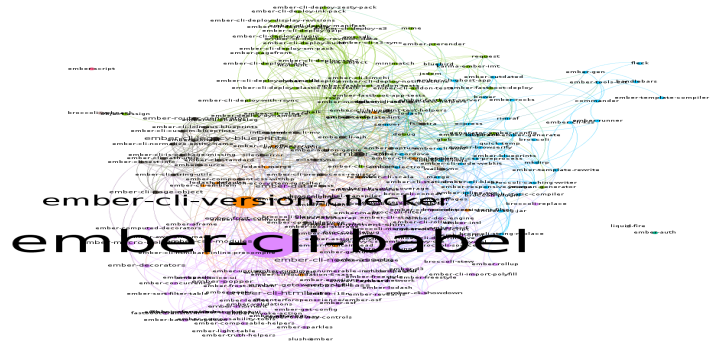
- What are software supply chains?
- Why care about FLOSS supply chains?
- SSCs
  - How to construct
  - Explore
  - Measure
- Operational data challenges

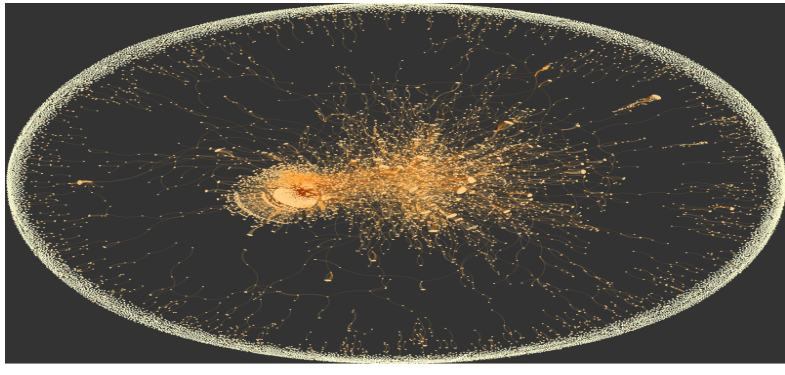


Summary

- What are software supply chains?
- Why care about FLOSS supply chains?
- SSCs
  - How to construct
  - Explore
  - Measure
- Operational data challenges

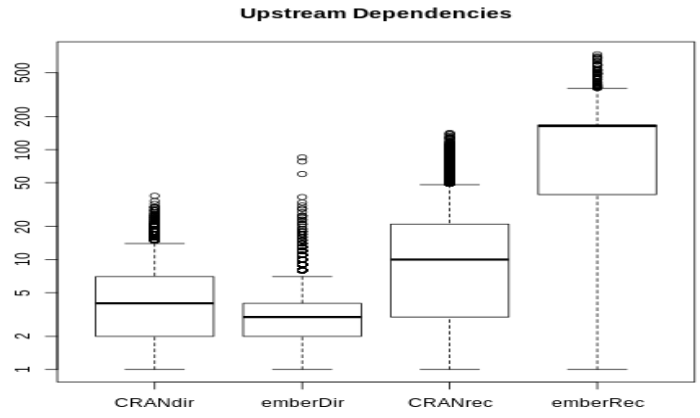


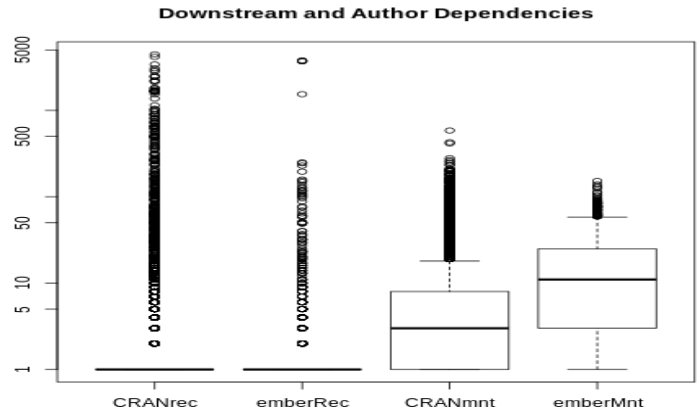




Summary

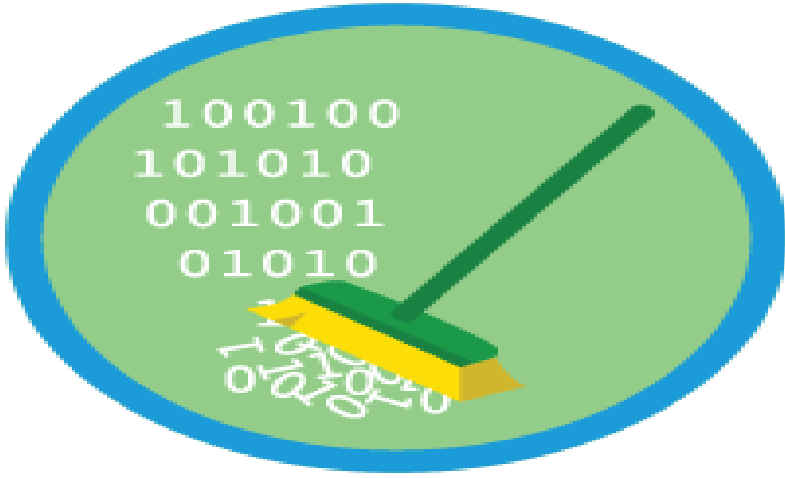
- What are software supply chains?
- Why care about FLOSS supply chains?
- SSCs
  - How to construct
  - Explore
  - Measure
- Operational data challenges





Summary

- What are software supply chains?
- Why care about FLOSS supply chains?
- SSCs
  - How to construct
  - Explore
  - Measure
- Operational data challenges



Bio

Audris Mockus worked at AT&T, then Lucent Bell Labs and Avaya Labs for 21 years. Now he is the Ericsson-Harlan D. Mills Chair professor in the Department of Electrical Engineering and Computer Science of the University of Tennessee.
He specializes in the recovery, documentation, and analysis of digital remains left as traces of collective and individual activity. He would like to reconstruct and improve the reality from these projections via methods that contextualize, correct, and augment these digital traces, modeling techniques that present and affect the behavior of teams and individuals, and statistical models and optimization techniques that help understand the nature of individual and collective behavior. His work has improved the understanding of how teams of software engineers interact and how to measure their productivity.
Dr. Mockus received a B.S. and an M.S. in Applied Mathematics from Moscow Institute of Physics and Technology in 1988. In 1991 he received an M.S. and in 1994 he received a Ph.D. in Statistics from Carnegie Mellon University.

Abstract

The open source universe represents the ultimate ecosystem of individuals and organizations creating and enhancing common good. Its health may be critical for sustained innovation, but little is understood beyond a few well studied large projects and ecosystems. We quantify aspects of knowledge supply chains in this universe by constructing, refining, and analyzing a time series of bipartite networks representing individuals and the source code. To accomplish that we first collect version control data representing a large fraction of code changes in public version control systems. We then devise and fit behavioral fingerprinting models to resolve synonyms and homonyms for developer, project, and source code IDs. The resulting time-dependent relationships among developers are then interpreted as knowledge flows and analyzed to quantify their global and local properties. Surprisingly, these networks represent several types of online activities in addition to software development and are not separated into individual projects but link the vast majority of developers into a single connected sub-graph. This global knowledge supply chain can then be studied to understand and quantify its emergent properties and associated risks.

References

Audris Mockus.
Succession: Measuring transfer of code and developer productivity.
In *2009 International Conference on Software Engineering*, Vancouver, CA, May 12–22 2009. ACM Press.