Debsources

Live and Historical Views on Macro-Level Software Evolution

Matthieu Caneill Stefano Zacchiroli
zack@pps.univ-paris-diderot.fr

Laboratoire PPS, Université Paris Diderot

18 September 2014
8th International Workshop on
Empirical Software Engineering and Measurement
Turin, Italy

Macro-level software evolution

> *"Software evolution in the large"*
>
> — *Gonzalez-Barahona et. al, 2009*

The study of software evolution, at the scale of software collections, at the granularity they allow (e.g., releases of individual software components).

On studying software collections

Pros

- relevant/popular software distribution model
- long lives (e.g., decades)
- uniform access to the history of contained software
- help with (researcher) selection bias

Cons

- *ad hoc* software ecosystems
- homegrown tools, conventions, social norms

Debian

- popular Free and Open Source Software (FOSS) distribution
- 20+ years of history
- one of the largest curated software collections

- good proxy of popular/ relevant FOSS projects
- popular ESE/MSR subject
- *ad hoc* software ecosystem (tools, conventions, etc.)

Debsources goal

Ease macro-level software evolution studies on FOSS as a whole, using Debian as a proxy.

Debian

- popular Free and Open Source Software (FOSS) distribution
- 20+ years of history
- one of the largest curated software collections

- good proxy of popular/ relevant FOSS projects
- popular ESE/MSR subject
- *ad hoc* software ecosystem (tools, conventions, etc.)

Debsources goal

Ease macro-level software evolution studies on FOSS as a whole, using Debian as a proxy.

What is Debsources?

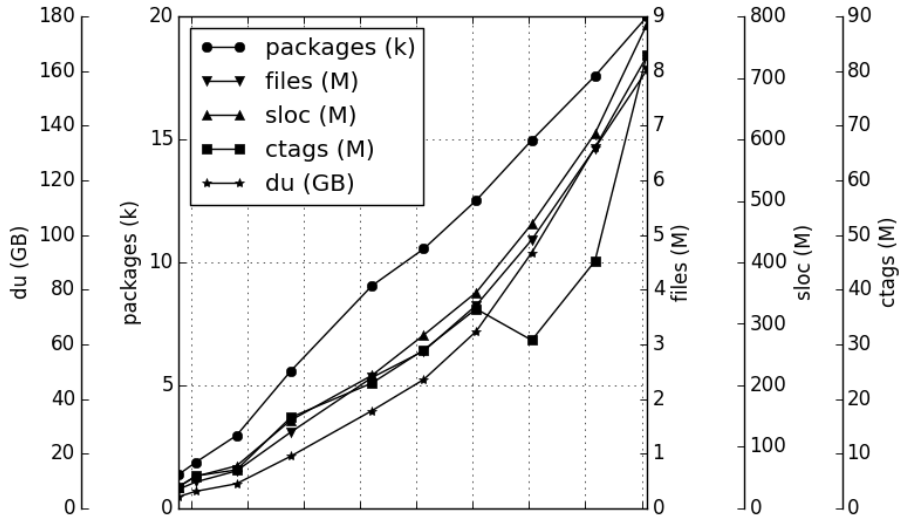
1. an infrastructure to publish Debian source code on the Web
2. a notable instance,[1] indexing *all* Debian source code to date

For end users:

- browse/search source code
- syntax highlighting
- pinpoint code lines, add msgs

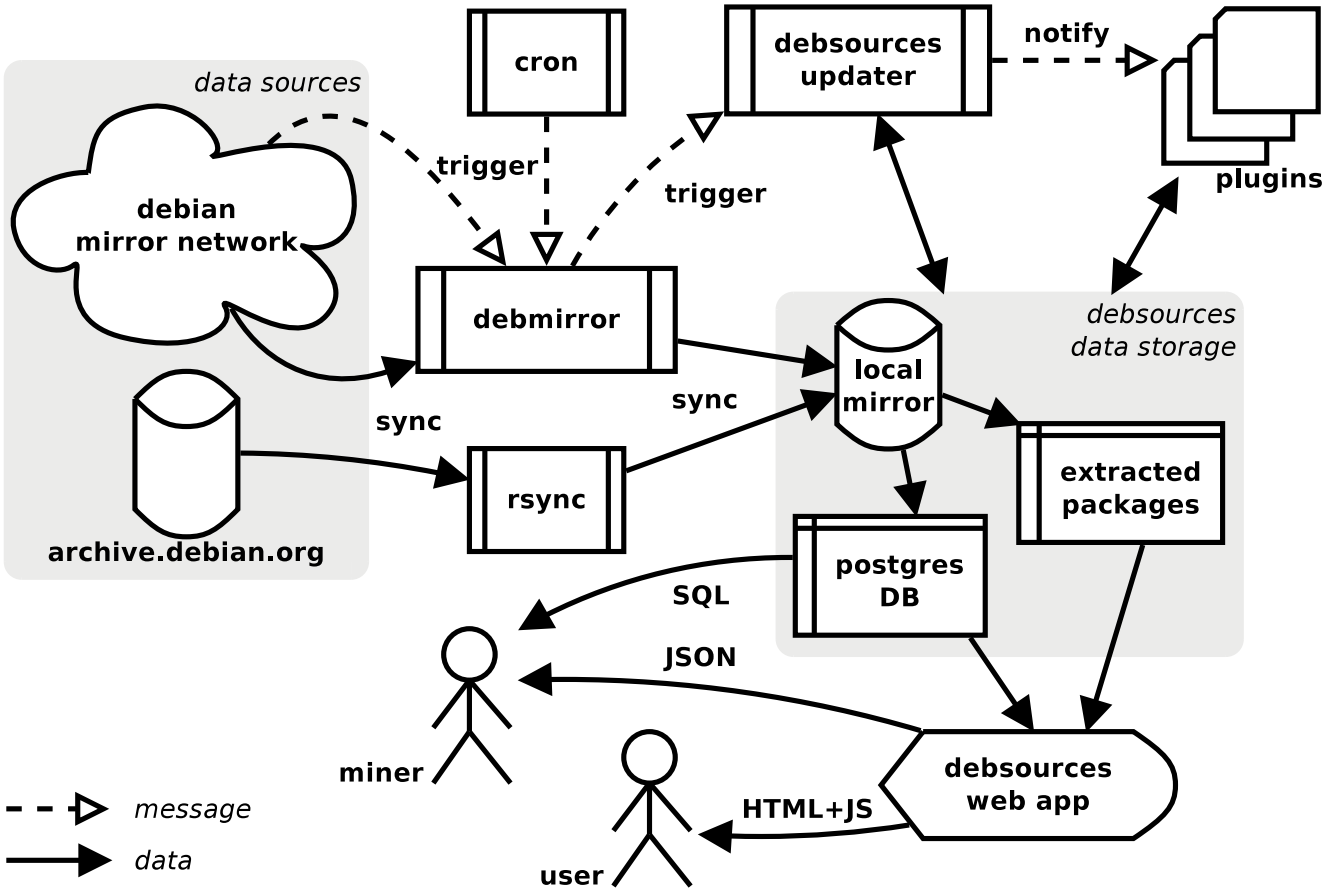
For data miners:

- Debian macro-level evolution
- 20+ years of history
- live/perennial monitoring



[1] `http://sources.debian.net`, source code available, license AGPL

Architecture



In essence, Debsources does the heavy lifting of maintaining a general purpose [always up to date] storage for Debian source code, enabling plugin authors to focus on data extraction.

Available plugins

- disk usage
- `sloccount`
- ctags (functions, classes, types, etc.)
- checksums (SHA256)
- ~~file count~~ (implicit)

Self-assessment: very little effort needed to write plugins for popular source code metrics.

Most complex plugin (ctags): ≈100 SLOCs

Plugin — example (sloccount)

```
def add_package(session, pkg, pkgdir, file_table):          # plugin excerpt
    if 'hooks.fs' in conf['backends']:
        if not os.path.exists(slocfile):  # run sloccount only if needed
            try:
                cmd = ['sloccount'] + SLOCCOUNT_FLAGS + [pkgdir]
                with open(slocfile_tmp, 'w') as out:
                    subprocess.check_call(cmd, stdout=out,
                                          stderr=subprocess.STDOUT)
            except subprocess.CalledProcessError:
                if not grep(['^SLOC total is zero,', slocfile_tmp]):
                        # rationale: sloccount fails
                    raise  # when it can't find source code
            finally:
                os.rename(slocfile_tmp, slocfile)
    if 'hooks.db' in conf['backends']:
        slocs = parse_sloccount(slocfile)
        db_package = dbutils.lookup_package(session, pkg['package'],
                                            pkg['version'])
        if not session.query(SlocCount).filter_by(package_id=db_package.id)\
                                       .first():
            # ASSUMPTION: if *a* loc count of this package has already been
            # added to the db in the past, then *all* of them have
            for (lang, locs) in slocs.iteritems():
                sloccount = SlocCount(db_package, lang, locs)
                session.add(sloccount)
```

sources.debian.net — do it yourself

1. deploy the Debsources software
2. point it to a nearby Debian mirror
   - optional: request push update notifications
3. trigger 1st update run `$ bin/update-debsources`

4. mirror `archive.debian.org` `$ rsync`
5. inject all archived suites `$ bin/suite-archive add`

- processing time (I/O bound): ≈13 days (8 for live suites)
- disk usage:[2] ≈840 GB
  - 150 GB (mirror) + 610 GB (extracted packages) + 75 GB (DB)

[2] data snapshot, 9 March 2014

sources.debian.net — do it yourself

1. deploy the Debsources software
2. point it to a nearby Debian mirror
   - optional: request push update notifications
3. trigger 1st update run `$ bin/update-debsources`

4. mirror `archive.debian.org` `$ rsync`
5. inject all archived suites `$ bin/suite-archive add`

- processing time (I/O bound): ≈13 days (8 for live suites)
- disk usage:[2] ≈840 GB
  - 150 GB (mirror) + 610 GB (extracted packages) + 75 GB (DB)

[2] data snapshot, 9 March 2014

sources.debian.net — dataset

data model (excerpt)

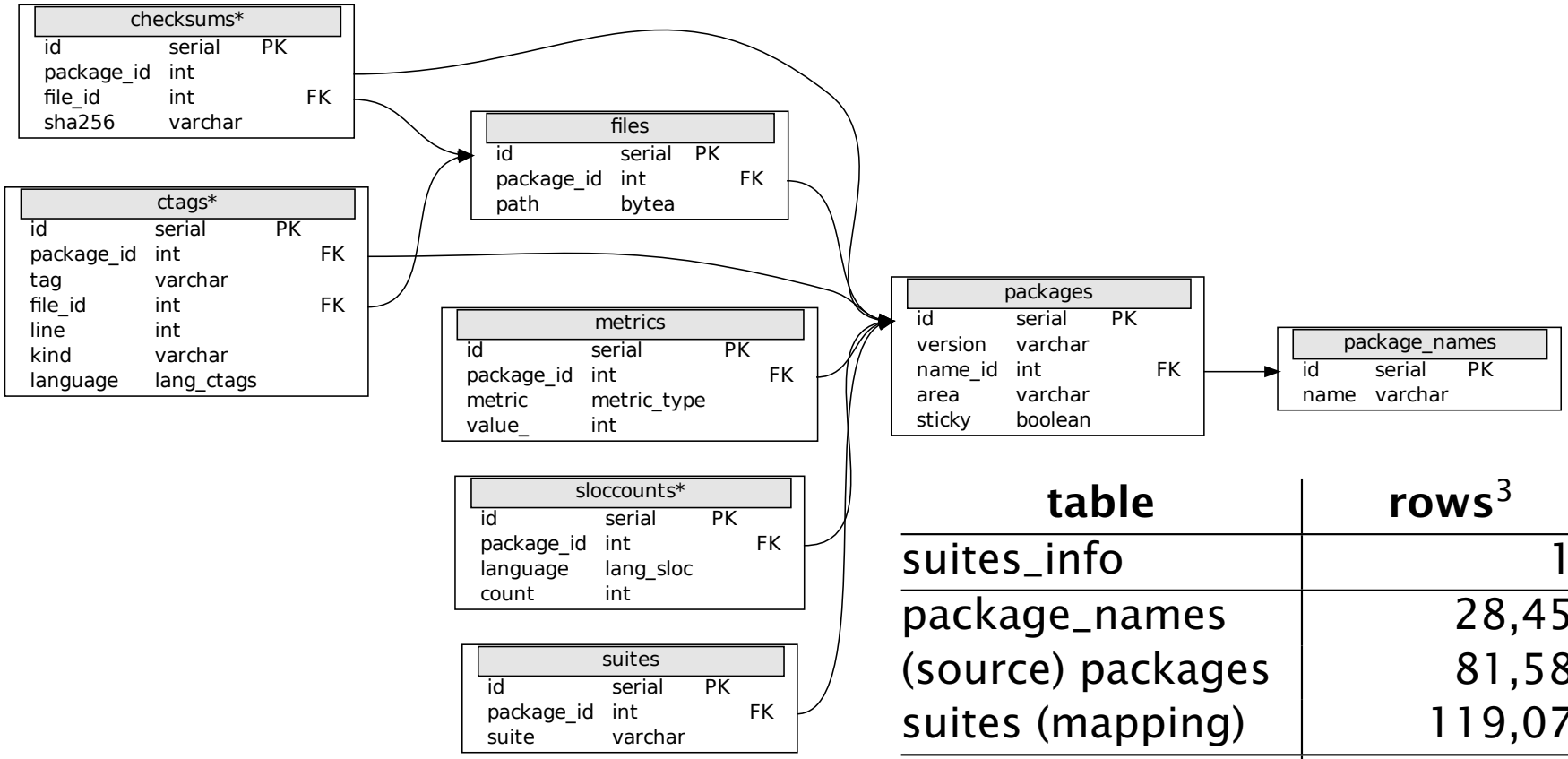


table	rows[3]
suites_info	16
package_names	28,454
(source) packages	81,582
suites (mapping)	119,078
metrics* (e.g., du)	81,582
sloccounts*	290,961
checksums*	33,495,057
ctags*	317,853,685

[3] data snapshot, 9 March 2014

Replication study

Gonzalez-Barahona, Robles, Michlmayr, Amor, and German

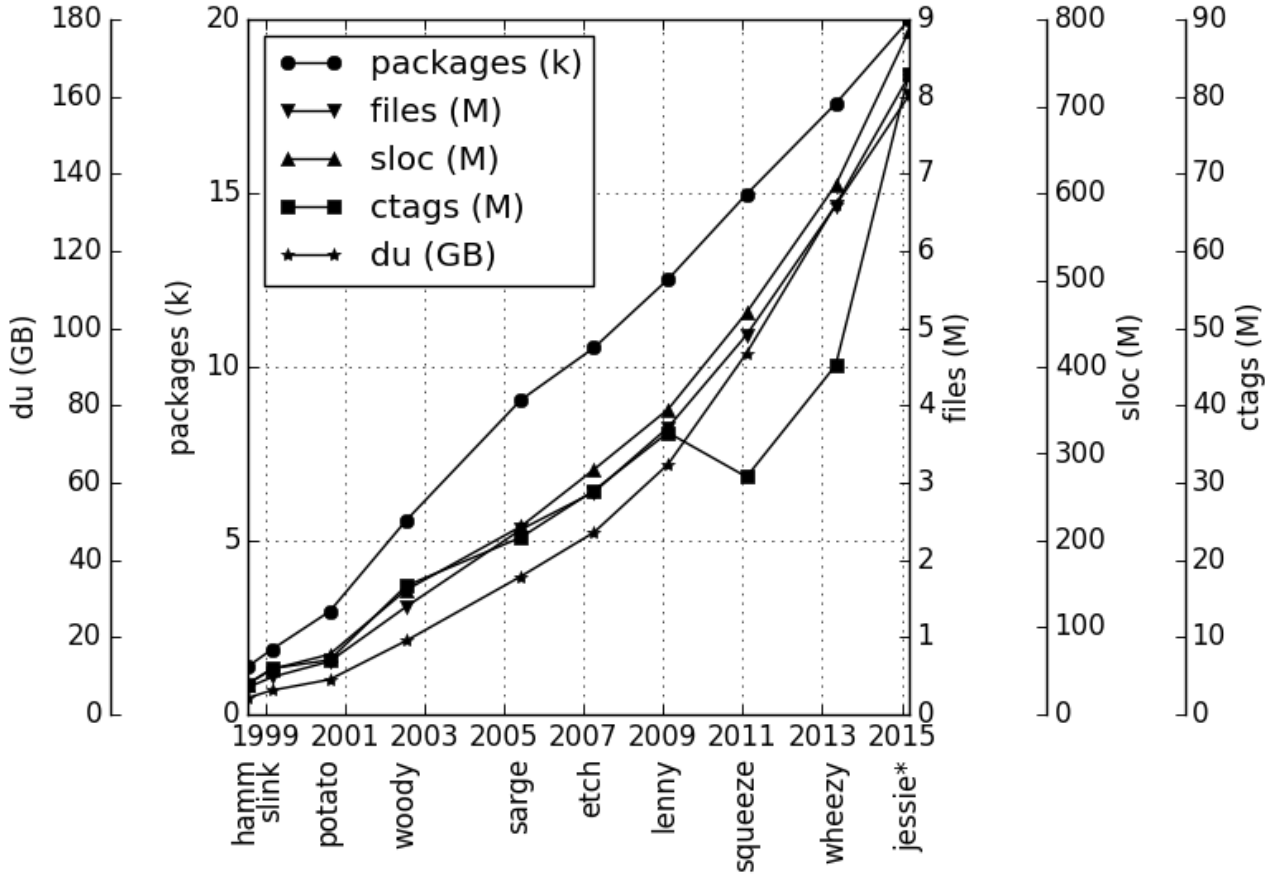
Macro-level software evolution: a case study of a large software compilation

Empirical Software Engineering, 14(3):262–285, 2009

original	**replica**	(remarks)
total (release) size	✓	larger dataset
package size	✓	
package maintenance	✓	more precise diff size evaluation
programming lang.	✓	more lang. (make, SQL, XML)
file sizes (per lang.)	✓	relying on file extension (bug)
dependencies	✗	no *binary* packages, yet

We have obtained slightly different results than our reference study, but confirmed the general trends and updated them in light of 7 extra years of evolution history.

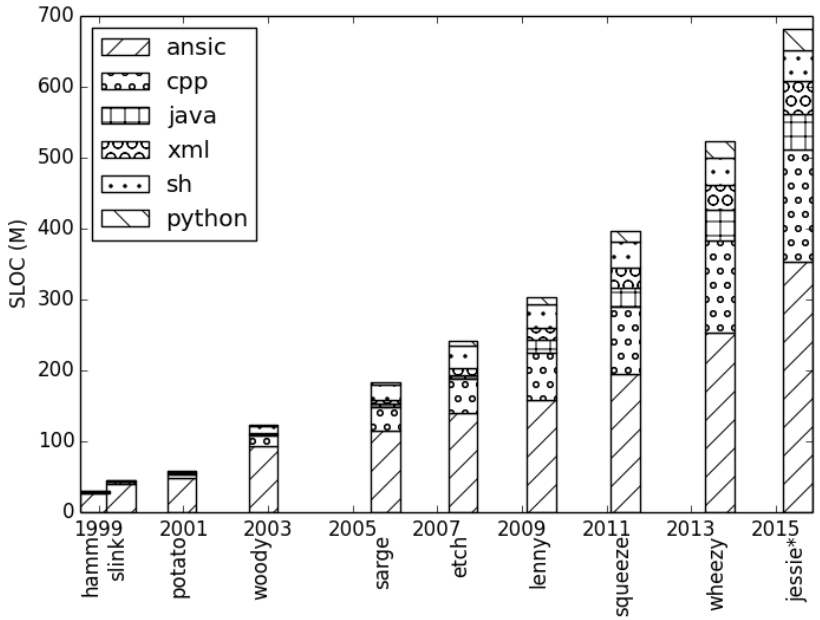
Replica highlight #1: total size

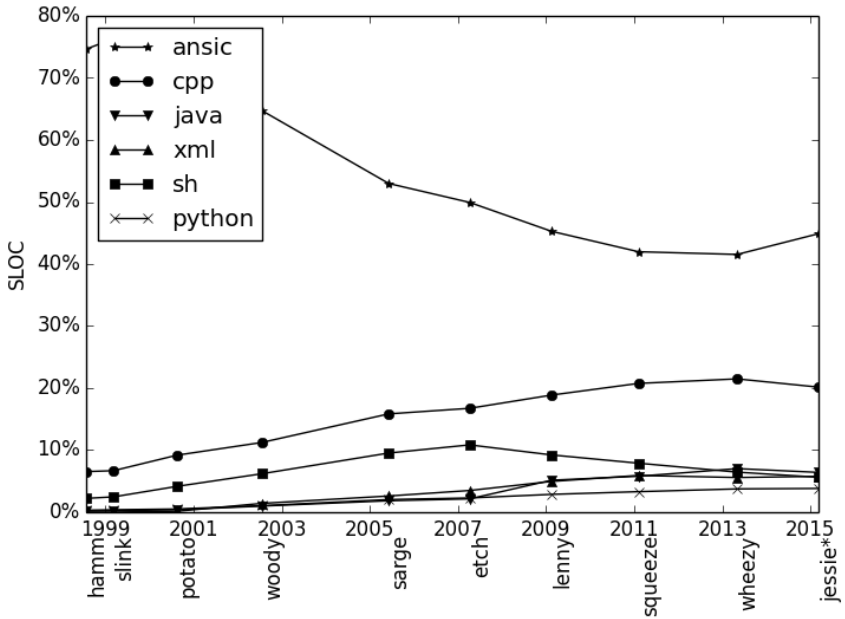


- larger dataset (≈400 pkg./rel.)
- correlation confirms Herraiz et. al, 2006 & 2007
- exception: package count (distro-level refactoring?)
- pre-*etch* (2007): growth rate slows down (allegedly, due to complexity ceiling)
- post-*etch*: growth rate increases

Replica highlight #2: programming languages

top-5 most popular programming languages in Debian over time





Recent trends (post-*etch*, 2007):

- C still leads, steady (absolute) growth
- C stops losing (relative) ground to C++
- decrease of Perl/Shell popularity
- Python rises (more maintainable glue code?)
- Lisp halves its popularity
- Java no longer under-represented

Replica highlight #3: package maintenance

Changes between Debian releases: ‘c’ for common, ‘u’ for unchanged (upstream), and ’m’ for modified packages (common \ unchanged):

from \ *to*	slink	potato	woody	sarge	etch	lenny	squeeze	wheezy	jessie*	sid*
hamm	1324c 842u	1198c 463u	1079c 270u	958c 175u	864c 148u	782c 124u	719c 100u	670c 81u	648c 75u	663c 75u
slink		1657c 742u	1455c 384u	1281c 252u	1155c 210u	1037c 172u	941c 136u	881c 113u	852c 105u	872c 105u
potato			2456c 935u	2118c 551u	1881c 436u	1683c 352u	1497c 271u	1399c 220u	1359c 210u	1387c 211u
woody				4588c 1688u	3953c 1156u	3497c 908u	3021c 633u	2787c 520u	2680c 486u	2752c 494u
sarge					7671c 3832u	6828c 2597u	5903c 1717u	5353c 1369u	5102c 1240u	5259c 1272u
etch						9230c 4578u	8041c 2906u	7216c 2205u	6881c 1948u	7088c 2000u
lenny							10836c 5272u	9631c 3676u	9181c 3153u	9457c 3249u
squeeze								13117c 6812u	12464c 5425u	12902c 5622u
wheezy									16543c 10132u	17042c 10519u
jessie*										19795c 19593u

	from previous suite to slink	potato	woody	sarge	etch	lenny	squeeze	wheezy
modified pkgs	556m	1305m	3127m	4462m	2879m	3287m	4129m	4453m
changed files per pkg	54.6%	64.4%	65.3%	67.5%	58.9%	59.8%	60.4%	57.2%

Debsources: Live and Historical Views on Macro-Level Software Evolution

- Debsources is a flexible platform to monitor large FOSS collections over long periods of time.
- The `sources.debian.net` dataset is valuable to scholars interested in macro-level software evolution.
- Debsources can turn one-shot studies (yours?) into live and perennial monitors of software evolution traits.

Thanks!
Questions?

Stefano Zacchiroli
zack@pps.univ-paris-diderot.fr
http://upsilon.cc/zack