The Debsources Dataset: Two Decades of Debian Source Code Metadata

Stefano Zacchiroli*
*Univ Paris Diderot, Sorbonne Paris Cité, PPS, UMR 7126, CNRS, F-75205 Paris, France
Email: zack@pps.univ-paris-diderot.fr

Abstract—**We present the Debsources Dataset: distribution metadata and source code metrics spanning two decades of Free and Open Source Software (FOSS) history, seen through the lens of the Debian distribution.**

Debsources is a software platform used to gather, search, and publish on the Web the full source code of the Debian operating system, as well as measures about it. A notable public instance of Debsources is available at http://sources.debian.net; it includes both current and historical releases of Debian. Plugins to compute popular source code metrics (lines of code, defined symbols, disk usage) and other derived data (e.g., checksums) have been written, integrated, and run on all the source code available on sources.debian.net.

The Debsources Dataset is a PostgreSQL database dump of sources.debian.net metadata, as of February 10th, 2015. The dataset contains both Debian-specific metadata—e.g., which software packages are available in which release, which source code file belong to which package, release dates, etc.—and source code information gathered by running Debsources plugins.

The Debsources Dataset offer a very long-term historical view of the macro-level evolution and constitution of FOSS through the lens of popular, representative FOSS projects of their times.

I. Introduction

Software is increasingly being distributed to final users by the means of software collections, and installed using package manager tools. Some collections are very tightly curated and integrated, like Free and Open Source Software (FOSS) distributions, others much more loosely so, like so called "app stores". The study of software evolution [1], [2] can no longer ignore software collections as relevant subjects for macro-level studies [3], [4], i.e., evolution studies at the granularity of individual component releases.

The study of software collections, however, poses specific challenges to scholars, due to a common tendency at growing *ad hoc* software ecosystems, made of homegrown tools, technical conventions, and social norms that might be hard to take into account when conducting empirical studies.

The Debsources platform [4] has been developed to counter those challenges in the specific case of Debian:[1] one of the most reputed and oldest (est. 1993) FOSS distributions, often credited as the largest organized collection of FOSS, and a popular subject of empirical software engineering studies (e.g., [5], [3], [6]). Debsources allow to gather, search, and publish on the Web the entire source code of Debian and measures about it. As a result, Debsources eases the study of macro-level FOSS evolution patterns, under the assumption that Debian is a representative sample of popular FOSS projects.

The most notable instance of Debsources is publicly available at http://sources.debian.net. It indexes all currently active (or "live") Debian releases, with several updates per day, as well as historical Debian releases going back almost 20 years.

Contributions: With this paper we release the Debsources Dataset, composed of metadata obtained indexing more than 3.5 billion lines of code, contained in about 90 thousand source package releases, from 18 Debian suites (or "releases"). The dataset relates source code files—about 40 millions of them—to the corresponding source packages, and those to Debian releases. Using suitable Debsources plugins, all source packages have been measured with `sloccount`,[2] indexed with Exuberant Ctags[3] for developer-defined symbols (functions, data types, classes, methods, etc.), and measured for disk usage; SHA256 checksums of all source files have also been computed. All obtained metadata are included in the dataset. The Debsources Dataset is a valuable resource for scholars interested in both long-term evolution studies of FOSS, and also in studying its composition thanks to the sub-file level granularity of its metadata.

The Debsources Dataset comes as a portable, textual dump of the PostgreSQL[4] database that underpins sources.debian.net. The dump discussed in this paper has been taken on Tue, 10 Feb 2015 13:48:40 +0000 and carries the (UNIX) timestamp of 1423576120.

Paper structure: Section II explains how the Debsources Dataset has been assembled and how it can be reproducibly, but costly, rebuilt. Section III describes the data scheme and gives some figures about its content. Section IV shows how to get started using the dataset; Section V discusses potential limitations. Before concluding, Section VI points to related work.

Dataset and software availability: The Debsources platform, used to produce the Debsources Dataset, is Free Software, released under the terms of the GNU Affero General Public License (AGPL). It is available from the Git repos-

This work has been partially performed at IRILL, center for Free Software Research and Innovation in Paris, France `http://www.irill.org`. Unless noted otherwise, all URLs in the text have been retrieved on February 27th, 2015.

[1]http://www.debian.org

[2]http://www.dwheeler.com/sloccount/

[3]http://ctags.sourceforge.net/

[4]http://www.postgresql.org

itory at http://anonscm.debian.org/cgit/qa/debsources.git; the version used to produce the dataset is shipped as part of it.

The Debsources Dataset is Open Data, made available under the terms of the Creative Commons Attribution-ShareAlike 4.0 International Public License (CC BY-SA). It is available from Zenodo at https://zenodo.org/record/16106, with DOI reference `10.5281/zenodo.16106`.

II. Data Gathering & Reproducibility

The Debsources Dataset has been assembled using Debsources [4] and injecting into it both current and historical Debian releases. Given that both Debsources and all Debian releases are freely available, the dataset can be recreated from scratch following the recipe given below.

a) Deploy Debsources: Installation and configuration instructions are available in the README and HACKING files of Debsources. The latest Git commit in use when the Debsources Dataset of this paper has been taken was 07a7eae4fbce583bfa51c4b376cb4a7fb2776528.

b) Mirror live suites: configure Debsources to mirror from a nearby Debian mirror.[5]

The last mirror update received by sources.debian.net before taking the Debsources Dataset snapshot was dated Tue, 10 Feb 2015, at around 11:00 UTC. To recreate the *exact* same dataset of this paper you should mirror from http://snapshot.debian.org/, passing a nearby time stamp. Alternatively you can mirror from the regular Debian mirror network, and obtain slightly more recent data for all live suites.

c) Trigger the first update run: Run `bin/debsources-update`. This will inject into your Debsources instance all previously mirrored Debian live suites.

d) Mirror historical releases: Mirror http://archive.debian.org using `rsync`, to retrieve all historical releases, which are no longer available from the regular mirror network.

e) Inject historical releases: Inject one by one all previously retrieved historical releases using `bin/debsources-suite-archive add` *`RELEASE`*.

f) Dump the database: If you need an actual database dump, in the same format used by the Debsources Dataset, you can dump the obtained Postgres database with something like `pg_dump --no-owner --no-acl -Fp debsources | xz -9 -c > debsources.$(date +%s).xz` If you do not need an actual dump, the Debsources Dataset is now available in your local *debsources* database.

The injection process is I/O-bound: the time needed to complete it depends mostly on I/O write speed. For reference, it took us ∼5 days to inject archived suites + 8 days for the live ones = ∼2 weeks—using 7.2 kRPM disks in RAID5, which are quite slow by today standards and certainly not optimized for write speed. Dumping and compressing the Postgres database took about 1.5 hours.

Disk usage is as follows: 166 GB for the local mirror + 682 GB for extracted source code on disk + 78 GB for the DB = ∼926 GB. The total disk size is quite tolerable for server-grade deployments, but might be demanding for desktop/laptop ones. If you are only interested in the metadata, both the local mirror and the actual source code can be removed after the initial injection, dividing disk usage by a factor 10.

The above figures assume that all the plugins used to produce the Debsources Dataset are enabled: disk usage, sloccount, ctags, and checksums. A significant part of the processing time (∼40%) is devoted to ctags indexing. If you are not interested in those data, injection time can be cut in half by disabling the corresponding plugin.

Note that to *use* the Debsources Dataset you do *not* need to go through the above process, which is documented here for information and reproducibility purposes only. The dataset is ready to use as is; see Section IV for a quick start guide.

III. Database Schema

The Debsources Dataset comes as a Postgres database. Its SQL schema, generated by PostgreSQL Autodoc,[6] is shown in Figures 1–2. A brief description of each table is given below:

- *package_names:* Debian *source* package names. They are stable across releases of the same package, and usually correspond to upstream FOSS project names, e.g., bash, linux (the kernel), libreoffice, etc.
- *packages:* source package releases (or "versioned source packages"), with one table entry for each known version of a given package name, e.g., bash 2.03-6, bash 3.2-4, linux 3.16.3-2, libreoffice 4.3.3-2.
- *suites_info:* static information about known Debian releases; for each release its name, version, and release date are given.
- *suites:* mappings from versioned source packages to Debian releases.
- *files:* individual source files, pointing back to the containing source package; identical files, shipped by multiple packages, have multiple entries in this table. Note that due to space constraints the actual *content* of source files is not included in the Debsources Dataset.
- *checksums:* SHA256 checksums of all source files. Note that some files (e.g., special device files) cannot be checksummed, so not all file entries have corresponding entries in this table.
- *sloccounts:* `sloccount` results for each versioned source package. The table contains one entry for each $\langle$language,versioned-package$\rangle$ pair; the absence of a language entry for a given package means that the language has not been detected by `sloccount` on that version.
- *ctags:* `ctags` results for each versioned source package. The table contains one entry for each developer-defined symbol, in a given source file, together with precise file location and kind of symbol (function, type, method, etc).
- *metrics:* simple, uniform metrics about known packages are collected here. Currently the only metric of this kind

[5]https://www.debian.org/mirror/list

[6]http://autodoc.projects.pgfoundry.org/

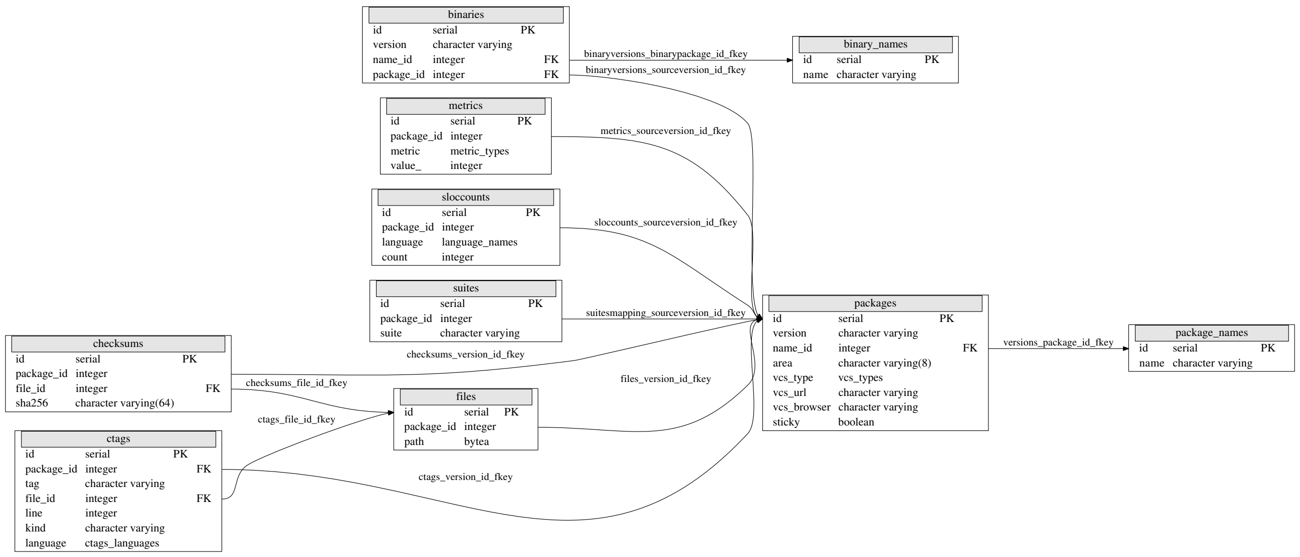

Fig. 1. Debsources DB schema, part 1: main tables

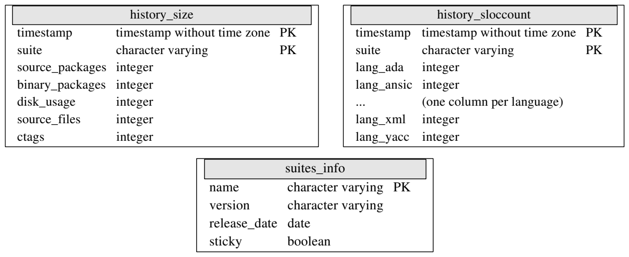

Fig. 2. Debsources DB schema, part 2: history and auxiliary tables

TABLE I
DEBSOURCES DATASET TABLE SIZES

table	tuples	disk size
checksums*	37,169,366	3,873 MB
ctags*	367,372,807	24 GB
files	37,267,117	2,746 MB
history_size	14,489	960 KB
history_sloccount	13,566	2,504 KB
metrics*	87,274	3,800 KB
package_names	30,264	1,488 KB
packages	87,274	9,696 KB
sloccounts*	310,554	13 MB
suites	123,451	5,408 KB
suites_info	18	16 KB

is disk usage. This table thus reports source total code size, in bytes, for each versioned source package.

- *history_sloccount, history_size:* historical, per-release time series of `sloccount` and dataset size values

To give an idea of the breadth of the Debsources Dataset, Table I gives the sizes of the main tables in the dataset, as number of tuples and occupied disk space.

If tables that take a considerable amount of disk space are unneeded, they can be dropped (possibly at import time) to save space. In particular, the ctags table is very large, but relevant only for sub-file granularity analyses.

The 18 injected suites cover all live releases, and all historical releases with the exception of Debian 1.1 "buzz" and 1.2 "rex". The exception is due to the old, non-standard package format used by those releases. Supporting such format is not hard, but requires an additional abstraction layer that is not implemented in Debsources yet.

The dataset contains $\sim$30,000 differently named packages, occurring in $\sim$90,000 distinct $\langle$name,version$\rangle$ pairs, for an average of 2.88 versions per package. The number of mappings between versioned packages and suites, $\sim$120,000, is significantly higher than the number of packages due to packages occurring in multiple releases.

The dataset covers $\sim$40 M source files, a whopping $\sim$370 M developer-defined symbols (ctags), and $\sim$300,000 $\langle$language,package$\rangle$ pairs for an average of 3.56 different programming languages occurring in each package version.

IV. How to Use

The Debsources Dataset is a textual dump of a PostgreSQL database, compressed in XZ format.[7] The dump has been obtained from Postgres 9.3 using `pg_dump`, but it should be compatible with any version of Postgres $\geq$ 9.1.

To use the dataset you should first install Postgres, then create a dedicated database, and finally import the dataset into it. For the last two steps you can proceed as follows, acting as a user with suitable Postgres permissions:

1) `createdb debsources`
2) `xzcat debsources.`*`TS`*`.xz | psql debsources`

where *TS* is the version (timestamp) of the Debsources Dataset you have got, e.g., 1423576120.

[7]http://tukaani.org/xz/format.html

On a modern high-end laptop equipped with a fast SSD disk, the import takes about 3.5 hours. The freshly imported database will require about 80 GB of disk space, 50 GB of which occupied by indexes.

To query the dataset you can then connect to the database via the Postgres API and run ordinary SQL queries. For interactive querying the database, the standard Postgres shell can be invoked as `psql debsources`.

V. Threats to Validity

As discussed in Section III, the Debsources Dataset does not include the first 2 Debian releases. Additionally, due to a regression in `dpkg-source` (see http://bugs.debian.org/740883), 12 packages from historical releases cannot be extracted and are missing from the dataset. We do not expect such a tiny number of packages to significantly impact the usefulness of the dataset.

`sloccount` and Exuberant Ctags are starting to show their age and suffer from a lack of active maintenance. Most notably, they do not support languages like Scala and JavaScript, which might then be underrepresented in the dataset. The case of JavaScript is particularly worrisome, due to its increasing popularity for server-side Node.js applications.

The binaries and binary_names tables, relative to Debian *binary* packages, are currently empty in the dataset due to the lack of support in Debsources. This should not affect any analysis targeting *source* code only, which is the main focus of the Debsources Dataset.

The content of the history_sloccount and history_size tables is not easily reproducible, due to the unwieldiness of versioning the dataset as a whole. It should be possible to recreate them injecting into Debsources data from http://snapshot.debian.org, but that service "only" goes back to 2005. Also, the history_size table started collecting data a few days later than history_sloccount.

VI. Related Work

Debsources [4] is the software platform used to produce the Debsources Dataset. It can be used to recreate the dataset (see Section II). It can also be used, after import (see Section IV), to browser and search the metadata; however, as the dataset does not include the actual source code, rendering of individual source code files will not work.

Reproducing the findings of a former macro-level software evolution study [3] motivated in part the development of Debsources. That study also shows the results of running `sloccount` on Debian releases over the 1998–2007 period. The Debsources Dataset covers twice that period, offers more metadata (ctags, disk size, checksums), and is publicly available from archival storage (Zenodo), whereas the dataset URL from [3] has been down for a few years now.

The Ultimate Debian Database (UDD) [7] has assembled a large dataset about Debian and some of its derivatives, and is a popular target for mining studies [5]. UDD however lacks the time axis—with the sole exception of a history table used to store time series which cannot be recreated from local storage.

Boa [8] is a DSL and an infrastructure to mine FOSS project collections like forges. Boa's dataset is larger in scope than Debsources (e.g., it contains SourceForge) and also more fine grained, reaching down to the VCS level, but does not correspond to curated software collections like FOSS distributions. That has both pros (it allows to peek into unsuccessful projects) and cons: contained projects are less likely to be representative of what was popular at the time. The time horizon is also more limited than that of Debian.

FLOSSmole [9] is a collaborative collection of datasets obtained by mining FOSS projects. Many datasets in there are about Debian but no one is, by far, as extensive as the Debsources Dataset.

VII. Conclusion

We have presented the Debsources Dataset: source code metadata and measurements spanning two decades of Free and Open Source Software (FOSS) history, through the lens of the popular Debian distribution.

The dataset contains increasingly more fine-grained information (software packages $\rightarrow$ releases $\rightarrow$ source code files $\rightarrow$ checksums $\rightarrow$ developer-defined symbols) about more than 3.5 billions lines of source code, from popular FOSS projects, representative of the state of FOSS at their times.

The Debsources Dataset is publicly available as Open Data, documented, and reproducible using Debsources and Debian Project data. However, recreating it takes a non-negligible amount of resources. Its availability as a ready to use database dump can therefore ease the work of scholars interested in macro-level software evolution, and in the history and composition of FOSS.

References

[1] F. P. Brooks, Jr., *The mythical man-month: essays on software engineering*, 2nd ed. Addison-Wesley, 1995.

[2] M. M. Lehman, "Programs, life cycles, and laws of software evolution," *Proceedings of the IEEE*, vol. 68, no. 9, pp. 1060–1076, 1980.

[3] J. M. González-Barahona, G. Robles, M. Michlmayr, J. J. Amor, and D. M. Germán, "Macro-level software evolution: a case study of a large software compilation," *Empirical Software Engineering*, vol. 14, no. 3, pp. 262–285, 2009.

[4] M. Caneill and S. Zacchiroli, "Debsources: Live and historical views on macro-level software evolution," in *ESEM 2014: 8th International Symposium on Empirical Software Engineering and Measurement*. ACM, 2014.

[5] J. Whitehead and T. Zimmermann, Eds., *Mining Software Repositories, MSR 2010*. IEEE, 2010.

[6] P. Abate, J. Boender, R. Di Cosmo, and S. Zacchiroli, "Strong dependencies between software components," in *ESEM*, 2009, pp. 89–99.

[7] L. Nussbaum and S. Zacchiroli, "The ultimate debian database: Consolidating bazaar metadata for quality assurance and data mining," in *MSR*. IEEE, 2010, pp. 52–61.

[8] R. Dyer, H. A. Nguyen, H. Rajan, and T. N. Nguyen, "Boa: a language and infrastructure for analyzing ultra-large-scale software repositories," in *ICSE*. IEEE / ACM, 2013, pp. 422–431.

[9] J. Howison, M. Conklin, and K. Crowston, "FLOSSmole: A collaborative repository for FLOSS research data and analyses," *IJITWE*, vol. 1, no. 3, pp. 17–26, 2006.