Research Software Discovery: Challenges, Risks and Opportunities

Alexander Struck
Research IT
Cluster of Excellence Image Knowledge Gestaltung – An Interdisciplinary Laboratory at Humboldt-Universität zu Berlin
Berlin, Germany
Alexander.Struck@hu-berlin.de

***Abstract*—Research software is an integral part of scientific investigations. The paper identifies challenges, risks and new opportunities in research software publication and discovery. The diverse code discovery landscape is mapped and agents with their business models identified. Examples for discovery tools and strategies are given to support the classification. Reproducibility of research and reuse of code may improve if software discovery was easier. Researcher conducting a search for existing software in the context of a state-of-the-art report or a software management plan could use this paper as a guideline for their information retrieval strategy.**

***Index Terms*—Information retrieval, scientific computing, sustainable development, research software, communities.**

I. Introduction

Writing software has been and is increasingly a large part of scientific research [1] but external reuse is still a challenge. This is partly due to a legacy code base where modern frameworks would require expensive refactoring, although small task solutions (e. g. I/O) see some reuse [2]. State-of-the-art reports or software management plans are sometimes asked for in grant proposals but finding and evaluating existing relevant software requires significant resources. Reviews on existing tools for a particular task require use of a diverse range of platforms to cover it extensively. Here we aim at a mapping of the existing landscape for software discovery and describing some of the agents. The documentation hereafter shall improve discoverability of already existing software with a potential for reuse.

A. On publishing research results

Research has long been communicated via letters, articles and books. Priority is an important aspect of claiming (fame for) a particular result, most notoriously in patents. Societies and commercial publishing houses created the infrastructure for fast and wide distribution. The business model of some publishers and service hosts limits the availability of research results to those academics associated with a resourceful organization. These developments spawned the Open Access movement to make articles and books freely available, sometimes in self-hosted infrastructure. At the same time research results are mostly published as written text and are condensed, either for marketing purposes or due to e. g. page restrictions, that make reproducibility cumbersome. This may have had some influence on the so-called "reproducibility crisis" in several disciplines that (in parts) influenced requests to publish research data. Research software gains more attention as an important part of creating the results and is therefore investigated for its usage [3] and reusability.

Although we see increasing numbers of DOIs being minted for software [4] developing software to solve research problems is not recognized as an academic achievement [1]. Knowledge on "licensing, persistent referencing and citation of research software" requires training [1].

The author would like to thank the German Research Foundation (DFG) for funding the Cluster of Excellence Image Knowledge Gestaltung.

B. On publishing research software

Anecdotal evidence suggests that software developers are sometimes invited to co-author papers where software played a role in achieving the results. This enables citation credit for research software engineers which is still a valid currency in academic circles, useful for careers and general progress. And with journals like Journal of Open Research Software (JORS) the centuries-long tradition of article publications is expanded onto software [5]. Citeability is a prerequisite of recognition in this academic tradition, achieved either by article publications or recent developments like CodeMeta and the Citation File Format (CFF) [6]. Research organizations like the German Max Planck Society promote the publication of software as a repository dump, combined with a persistent identifier (e. g. DOI), made available in a long-term archive [7].

Recent developments in the world of software repositories [8] made more researchers considering the assumed business model of their favorite code hoster [9]. The GitLab software for hosting local code repositories with added functionality is popular in academic circles but administrators in universities are sometimes shy to enable external authentication methods. This may decrease collaboration on and sharing of research software or it may lead cross-institutional research groups to turn to commercial service providers.

Requests for research proposals slowly recognize non-textual matter to be a relevant part of a research result and in turn, programs like Horizon 2020 or selected UK funding proposals (e. g. "Software for the future II") started requiring a data or software management plans including information on publication of such result.

C. Research Software Discovery

The F in FAIR stands for findability. Reuse of software would probably enhance sustainability due to increased attention and community building. But relevant research software is scattered over many places and the knowledge thereof is limited. We lack defined criteria on how many repositories one would have to consult and what search strategy to follow in order to conduct a comprehensive state-of-the-art report. Those reports are supposed to prevent lavish allocation of taxpayers' money to e. g. fund new software development. And we yet have to invent criteria for the evaluation of research software and its reusability, although this kind of evaluation is receiving some attention (by e. g. binder, Containerization and hosts like CodeOcean). Software requirement specification could help in identifying existing solutions but is hardly possible due the exploratory nature of the research process [10]. "[T]he lack of established repositories for research software makes it difficult to find existing software solutions,

potentially resulting in the unnecessary duplication of development work" [1]. Such redundant work may also be caused by mistrust in third-party code or the lack thereof in closed-source applications [11]. Users are asked to make an informed choice on the solutions available [1] but have problems understanding the inner workings. Some rely on personal recommendation or perceived quality when an article publication describes the software in a peer-reviewed journal [12]. Some consult their favorite search engine to identify software that may solve their problem. Some need to inquire with their local IT department due to rules and regulations on what software is allowed to run on their machines. Others admit that looking for and evaluating existing research software requires too many resources and is not considered in cost calculations of a project.

II. Agents improving discovery

A. Researchers

After positive evaluation of (third-party or self-written) code researchers may share recommendations in forums or social media environments StackExchange which are not necessarily research-oriented. Some enthusiasts will create lists out of perceived necessity to cover a certain field and share useful resources. This could be a review article like [13] or take the form of a curated website. Some projects may get funded to do research on data or software repositories like https://www.re3data.org/.

B. Funding and Supporting Agency

Although recognition of research software development as an academic achievement is still low funders provide money for software development and may suggest or require publication of code. Some do not mentioned code at all, like the Horizon 2020 program. Some (funding) organizations may provide a repository for publication and archiving.

- Numfocus
- Research Software Directory
- Science Gateways Catalog

C. Publishers

Some academic publishing houses started to request the deposition of data relevant to a text publication (e. g. PLOS ONE guidelines). Some recommend particular discipline-specific repositories, other publishers encourage deposition of data within their commercial environment (e. g. Elsevier with Mendeley Data). While submission requirements have been rolled out for data relevant for an article in the first place, we are seeing some uptake regarding software submission [14]. Some specialized outlets like the IPOL journal [15] go even further. We quote from IPOL policy: "the publication of each algorithm is fourfold and includes:

- a manuscript containing the detailed description of the published algorithm, of its bibliography, along with commented examples and a failure case analysis;
- a software implementation of the algorithm in C, C++ or Matlab;
- an online demo, where the algorithm can be tested on data sets uploaded by the users;
- an archive containing extensive online experiments."

This obviously requires reviewers to be computationally literate. Most journals focused on research software (e. g. JORS & JOSS) follow similar strategies in their policies or guidelines (e. g. JORS recommendations respectively JOSS author guidelines). These journals incentivize to market research software by writing an article about it, in parts due to the established recognition system of article citations.

III. Tools and their offerings

A. Code Aggregators

Source and binary code is hosted in several locations. If there is a chance to aggregate code from several platforms and present it under a common metadata schema, it may ease its discovery. A promising attempt is currently under way by Software Heritage [16]. This archive aims at preserving code by crawling from several repositories and assigning persistent identifiers [17]. Their sources are:

- live and updated regularly: GitHub, Debian
- one shot archival: Gitorious, Google Code and GNU
- in progress: Bitbucket

This should cover a good amount of publicly available research software and acts as an archive, so that code is not lost when depublished in its original repository.

B. Search Engine Approach

The "Bielefeld Academic Search Engine (BASE)" is one of many platforms harvesting (mixed-content) repositories and offering one UI for information retrieval [18]. Unfortunately they only index metadata and not the full text or source code. But they offer SOLR-based indexing for you to discover more than 100,000 research software projects. The GUI offers to use doctype=software and qualifiers for licenses. Search engines like Google may offer even more recall (e. g. from project home pages) but lower precision when searching for software. Here, source code search engines fill a niche, examples being "sourcegraph", OpenGrok and "krugle".

C. Programming Language Package Repository

Some researchers using a particular language may package their functionality in a library or package, made available via built-in package managers which connect to repository platforms like CRAN. The R community created an ecosystem around the language to support developers. Software discovery examples that may also exist for other programming languages are:

- Rseek
- METACRAN
- Crantastic

D. Commercial Software Providers

Scientific (high performance) computing may be a profitable field. We see several vendors with their products in the market (with more or less lock-in). Examples are:

- Revolution Analytics (now belonging to Microsoft)
- Matlab
- Mathematica

E. Code Repositories

Public platforms hosting code and offering some basic management functionality include:

- GitHub
  Here, the long-lasting question for a sustainable business model has been answered by Microsoft. The competitor GitLab sent a congratulation note.
- Google Code
  Google decided to discontinue this service but still offers the archive. Example search URL
- Sourceforge.net
  is still active and is being recommended as a research software

repository but lost some of its reputation due to questionable business model decisions [19].
- Bitbucket.org
  owned by Atlassian is another code management platform which is popular due to feature-rich UI and free unlimited private repositories, because not every researcher wants to have code published publicly.

Search functionality varies but may include language and license filters. Some of these platforms facilitate(d) contribution and reuse.

F. Application Repositories

Some research requires tools developed for mobile devices. Executable binaries can be found in repositories like Apple's AppStore for iOS or Google's Play Store. Source code would need to be made available via other channels. The availability of tools is subject to terms of service and at the hands of uploaders. The long-term availability is unknown.

G. Executable Environment Hosts

As argued before, evaluation of third-party code is expensive and requires some level of literacy. But a successful evaluation may increase reuse and save taxpayer's money. It may also benefit reproducibility of articles introducing new software (algorithms), e. g. [20]. Here, some providers start to offer such environments, utilizing container technology popularized by Docker and Singularity. Code Ocean calls their Docker containers "Capsules". Biocontainers.pro is another service with similar offerings. Jupyter notebooks [21] can be published in environments like "binder".

H. Research (Data) Repositories

There are many public research results repositories that offer functionality for text, media, data and/or software. Some of them specialize in particular content features, others offer mixed content functionality (e. g. netlib.org). Publicly funded repositories include Zenodo.org, a privately owned platform would be Figshare.com and we see many repositories in university environments. Some organizations decide to reuse existing repository software like DSpace or develop their own CMS (e. g. Research Software Directory) reusing repository infrastructure like Zenodo.org. A variety of business models, funding schemes and API definitions require extensive evaluation for the maintainer and the users of such repositories.

I. Curated Web Lists and Catalogs

These platforms catalog software but do not host code. Some are focused on a particular discipline, some link to resources not limited to software code. Funding and maintenance are crucial for this type of resource. The following listing exemplifies the range of existing features and problems.

- Ideas
  is a catalog in the field of economics. From their about page: "RePEc (Research Papers in Economics) is a collaborative effort of hundreds of volunteers in 99 countries to enhance the dissemination of research in Economics and related sciences. The heart of the project is a decentralized bibliographic database of working papers, journal articles, books, books chapters and software components, all maintained by volunteers."
- openscience.org/software
  offers: "Collecting links about and providing a home for Open Source scientific software projects". The software listing has not been updated since March 2017 and is run by one person.
- 101 Innovations
  is a project investigating the creation and use of research tools, including software. They use Google Spreadsheet to display information and gather user input. The tools are assigned to and sorted by predefined research process phases. Entries in the Google Spreadsheet below line 659 appear to be unformatted user input which does not match the intended indexing of tools.
- Connected Researchers
  This is another website run by one person who takes suggestions. The latest postings were announced in November 2017 and the comment section of this WordPress blog has entries from the recent past.
- DiRT directory is a well-known repository but has problems with metadata quality. It continues to be funded by the Andrew W. Mellon Foundation and receive contributions, e. g. via Twitter mentions.
- swMath stands out here as it connects papers with software in both ways, publications related to the introduction of new software of papers mentioning (the reuse of) existing mathematical software. This continues to be funded and maintained by a research organization.

Some individuals feel the need to create such lists but allow contribution via git on e. g. GitHub. Some of these examples border on meta catalogs. These are all mixed content resources which may include software itself.

- Awesome Awesomeness
  offers two categories: language and general, that point to resources in research software development
- https://awesome.re
  is another community curated list of links to resources not related to a particular subject.
- Awesome Hacking
  is an example of a subject area meta listing.

J. Meta Catalogues

Some projects get funding for complex attempts to make repositories (and other useful resources) more accessible by listing and indexing them. One example is re3data.org which is a registry of research results repositories. Some of the listed repositories may only contain software and others mixed content. This registry tries to solve the challenge for researchers that data and software may exist somewhere but the sheer amount of repositories makes identification of reusable results hard. Such a registry could also prove helpful in identifying a suitable location to publish code and/or data. Unfortunately the implementation of re3data.org leaves much room for improvement, due to too many false positives and a questionable classification. Take the DBpedia entry as an example, where content is classified as:

- Databases
- Scientific and statistical data formats
- Software applications
- Source code
- Standard office documents
- Structured text
- other

DBpedia is a graph database offering Linked Data triples gathered from Wikimedia projects.

Science Gateways Catalog is another noteworthy attempt to catalog such resources.

K. Software Citation

A part of the challenges to discover relevant research software is the lack of citations or heterogeneity to cite (pieces of) software. This is currently being investigated by the research community [22]. We are seeing service offerings like converter tools between different formats and pages dedicated to educate the community, e. g. https://research-software.org/citation. Furthermore, the platform "Generation R" has published a series of articles on the topic which may increase awareness for software citation in general.

IV. Conclusions

Research software still lacks the recognition as actual research achievement. Some research software is published but it is not trivial to find relevant tools. This is in part due to business models, information silos and lack of trust in third-party code. Evaluation is expensive and hardly undertaken, even if journal articles introduce new software. Steps towards easing evaluation are taken by e. g. platforms offering executable environments. Attempts to register and index software repositories (resp. "science gateways") are undertaken but remain to prove beneficial for the "end-user developer" and should be coordinated. Some discipline-specific platforms or websites, driven by the research community, have gathered a significant amount of resources and are widely used by the respective community. Software may also get lost due to de-publication or retired platforms but code aggregation tries to minimize the loss.

V. Open questions and future work

We currently lack an understanding of how well research software is findable, given the tools to solve a particular problem may be scattered over many locations. We still have to determine and understand how many places a researchers should search in order to satisfy his/her own interest, or the reviewers questions and the funders quality assurance measures. We can assume that the business model of some platforms may influence visibility of certain code or projects but would require more empirical investigation. The overlap between certain tools regarding their features is likely to be significant. We probably have to accept redundant developer work due to a variety of reasons but lack an understanding of economical consequences for funded research projects. And it might be interesting to investigate how Current Research Information Systems (CRIS) may become helpful in identifying relevant research software. If indexing repositories and other locations proves to be beneficial, a study should be undertaken to determine a metadata schema.

References

[1] M. Katerbow and G. Feulner, "Recommendations on the development, use and provision of research software," 2018. [Online]. Available: https://zenodo.org/record/1172988

[2] V. R. Basili, D. Cruzes, J. C. Carver, L. M. Hochstein, J. K. Hollingsworth, M. V. Zelkowitz, and F. Shull, "Understanding the high-performance-computing community," *IEEE software*, vol. 25, no. 4, pp. p29–36, 2008.

[3] K. Li, E. Yan, and Y. Feng, "How is r cited in research outputs? structure, impacts, and citation standard," *Journal of Informetrics*, vol. 11, no. 4, pp. 989–1002, nov 2017. [Online]. Available: https://doi.org/10.1016/j.joi.2017.08.003

[4] M. Fenner, D. S. Katz, L. H. Nielsen, and A. Smith, "Doi registrations for software," 2018. [Online]. Available: https://blog.datacite.org/doi-registrations-software/

[5] N. C. Hong. In which journals should i publish my software? [Online]. Available: https://www.software.ac.uk/index.php/which-journals-should-i-publish-my-software

[6] S. Druskat, R. Haines, and J. Baker, "Citation file format (cff) - specifications," 2018. [Online]. Available: https://zenodo.org/record/1242911

[7] S. Janosch. (2017) How to assign a doi to software within mpg. [Online]. Available: http://www.de-rse.org/blog/howto/2017/05/08/how-to-assign-a-doi-to-software-within-mpg.html

[8] D. Bass and E. Newcomer. Buying github would take microsoft back to its roots. [Online]. Available: https://www.bloomberg.com/news/articles/2018-06-03/microsoft-is-said-to-have-agreed-to-acquire-coding-site-github

[9] D. Oberhaus. 13,000 projects ditched github for gitlab monday morning. [Online]. Available: https://motherboard.vice.com/en_us/article/ywen8x/13000-projects-ditched-github-for-gitlab-monday-morning

[10] J. Segal and C. Morris, "Developing scientific software," *IEEE Software*, vol. 25, no. 4, pp. 18–20, jul 2008. [Online]. Available: https://doi.org/10.1109/ms.2008.85

[11] R. Sanders and D. Kelly, "Dealing with risk in scientific software development," *IEEE Softw.*, vol. 25, no. 4, pp. 21–28, Jul. 2008. [Online]. Available: http://dx.doi.org/10.1109/MS.2008.84

[12] L. N. Joppa, G. McInerny, R. Harper, L. Salido, K. Takeda, K. O'hara, D. Gavaghan, and S. Emmott, "Troubling trends in scientific software use," *Science*, vol. 340, no. 6134, pp. 814–815, 2013.

[13] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010. [Online]. Available: https://arxiv.org/abs/0906.0612

[14] A. Editor, "Does your code stand up to scrutiny?" *Nature*, vol. 555, no. 7695, pp. 142–142, mar 2018. [Online]. Available: https://doi.org/10.1038/d41586-018-02741-4

[15] M. Colom, B. Kerautret, N. Limare, P. Monasse, and J.-M. Morel, "IPOL: A new journal for fully reproducible research; analysis of four years development," in *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, jul 2015. [Online]. Available: https://doi.org/10.1109/ntms.2015.7266500

[16] R. Di Cosmo and S. Zacchiroli, "Software Heritage: Why and How to Preserve Software Source Code," in *iPRES 2017: 14th International Conference on Digital Preservation*, Kyoto, Japan, Sep. 2017. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01590958

[17] M. Gruenpeter, "Software preservation: A stepping stone for software citation," 2018. [Online]. Available: http://genr.eu/wp/software-preservation/

[18] D. Pieper and F. Summann, "Bielefeld academic search engine (BASE)," *Library Hi Tech*, vol. 24, no. 4, pp. 614–619, oct 2006. [Online]. Available: https://doi.org/10.1108/07378830610715473

[19] Gluster. How far the once mighty sourceforge has fallen... [Online]. Available: https://www.gluster.org/how-far-the-once-mighty-sourceforge-has-fallen/

[20] F. Olivieri. (2017) Generation of private sound with a circular loudspeaker array and the weighted pressure matching method. [Online]. Available: http://doi.org/10.24433/co.692f87c0-7d3b-4608-ba4f-59a923d9faa8

[21] K. Thomas, R.-K. Benjamin, P. Fernando, G. Brian, B. Matthias, F. Jonathan, K. Kyle, H. Jessica, G. Jason, C. Sylvain, and et al., "Jupyter notebooks - a publishing format for reproducible computational workflows," *Stand Alone*, vol. 0, no. Positioning and Power in Academic Publishing: Players, Agents and Agendas, p. 8790, 2016. [Online]. Available: http://doi.org/10.3233/978-1-61499-649-1-87

[22] A. M. Smith, D. S. Katz, and K. E. N. and, "Software citation principles," *PeerJ Computer Science*, vol. 2, p. e86, sep 2016. [Online]. Available: https://doi.org/10.7717/peerj-cs.86