Software + Discovery Index

report

Software Discovery Index Meeting Report:

REQUEST FOR COMMENTS | DEADLINE: NOVEMBER 1st 2014

Click here to add comments.

INTRODUCTION

The National Institutes of Health (NIH), through the Big Data to Knowledge (BD2K) initiative, held a workshop in May of 2014 to explore challenges facing the biomedical research community in locating, citing, and reusing biomedical software. The workshop participants examined these issues and prepared this report summarizing their findings.

The constituents with the potential to benefit from improved software discoverability include software users, developers, journal publishers, and funders. Software developers face challenges disseminating their software and measuring its adoption. Software users have difficulty identifying the most appropriate software for their work. Journal publishers lack a consistent way to handle software citations or to ensure reproducibility of published findings. Funding agencies struggle to make informed funding decisions about which software projects to support, while reviewers have a hard time understanding the relevancy and effectiveness of proposed software in the context of data management plans and proposed analysis.

Though numerous changes are needed to address all these gaps, the workshop identified one fundamental prerequisite for success: an automated, broadly accessible system enabling comprehensive identification of biomedical software.

This objectives of this “Software Discovery Index” would be:

1. to assign standard and unambiguous identifiers to reference all software,
2. to track specific metadata features that describe that software, and
3. to enable robust querying of all relevant information for users.

If broadly used, this Software Discovery Index will form a cornerstone in a software ecosystem that benefits software developers, software users, journal publishers, and funding agencies.

The workshop attendees agreed that technical resources exist to create both this ecosystem and the needed tools to leverage it. The success of such efforts, however, depends on their acceptance by the scientific community: software developers must obtain identifiers for their software; users must cite software in their publications; journals must leverage and expose these citations; and if this is used properly, and judiciously, funding agencies should use this new wealth of information to shape funding decisions and long-term planning. It is only when each constituency sees benefits of engaging in this effort that significant progress can be made.

The ultimate goal of this effort is to ensure all publicly funded biomedical software is highly accessible to the research community. Making software easier to find, easier to cite, and easier to reuse are all necessary steps. It is also critical, however, to support the continued development and availability of software tools. Without access to both the tools and the scientific literature describing their use, the research community will not be able to select and use the best tools. Without tools maintained in common, open-access repositories such as GitHub and SourceForge, improvements to existing tools will be hampered. In all these areas, better support for software can help maximize the impact of NIH's investment in biomedical research.

A. FRAMEWORK SUPPORTING THE SOFTWARE DISCOVERY INDEX

The workshop identified many potential characteristics and features for an ecosystem in which users can locate, cite, and reuse software. As discussed at

the workshop, one prerequisite for such an ecosystem is the use of unique identifiers are obtained by developers and linked to software wherever it is hosted.

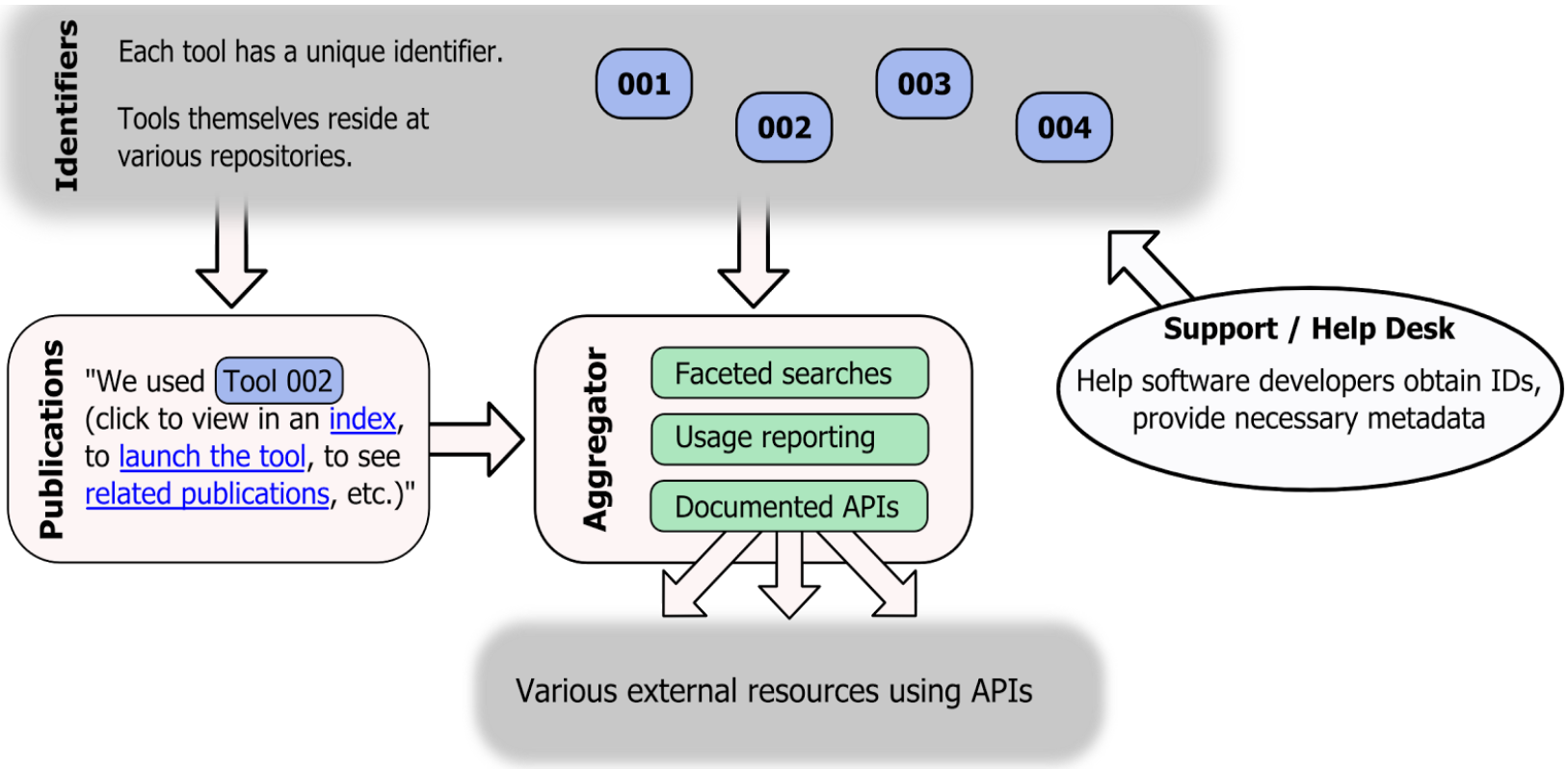


Figure 1. The proposed software ecosystem. Note that this does not show the other complexities such as versioning and attribution (see text).

Unique identifiers

Unique identifiers for biomedical software are critical for all that follows. The specific system of identifiers used is of far less importance than the adoption of those identifiers among software developers, software users, and publishers. Even so, however, the choice of identifiers could make it easier or harder to meet the needs of each of these communities.

The temporally dynamic nature of software development makes unambiguous identification difficult. Individual software packages may have many versions, may be branched along different development paths, and may be bundled into collections with other packages. Identifiers must operate across

all of these cases, both disambiguating and linking related tools.

The system of identifiers should also enable the association of metadata[1] to software. The metadata so associated should facilitate the identification of scientifically relevant software packages. Collecting this information as a static catalog or set of web pages runs the significant risk of perpetuating stale metadata. In facing that challenge, the open-source software community has developed multiple ways to capture metadata on projects with minimal duplication. The most common approach is to define a format in which the project metadata can be stored as part of the project itself and then scraped by any interested parties. This means that the software developers only have to provide the metadata once, enabling the Software Discovery Index and other interested parties to scrape and use it. It also ensures that updates by the software developers are reflected in all repositories. In this effort, controlled vocabularies and ontologies may prove to be useful, but should not be the primary focus of the initial effort.

Connections to publishers

There is increasing recognition within the scientific community that recording how software is used is a critical part of the scientific record. The dissemination of scientific results, however performed, must unambiguously describe the software used to generate those results and the steps performed. With publications currently the *lingua franca* for disseminating biomedical research results, connections with journal publishers will be essential for this effort.

Comprehensively and efficiently tracking the use of software in research requires a new standard for software citations. At present, most software is cited

indirectly by citing either a publication or a URL where the software is described. Citing publications leverages the existing publication citation infrastructure, but it only enables citation of software described in publications. Even software described in publications, if actively-developed, is likely to cycle through many more released versions than publications. URLs pointing at descriptions of software not only fail to provide a standardized mechanism for tracking versions and metadata, but also frequently break as documentation and source code move. To support reproducibility and archiving, a persistent mechanism for citing software, even software no longer being actively developed, is critical.

A consistent system of unique identifiers for software and an API for querying those identifiers will enable a better system for citing software. When used in publications, these identifiers would make it possible to identify all publications using a particular software tool. Retrieving the citations from publications can be accomplished through direct submission from journals, extraction by MEDLINE, and full-text mining.

One major initiative that aims to address this issue is the use of Research Resource Identifiers (RRIDs), currently underway at FORCE11 and lead by a partnership between the University of California, San Diego, and Oregon Health & Science University. The RRID project makes it easier to track key research resources within the biomedical literature by ensuring that authors provide unique identifiers (RRIDs) for each resource used to produce the results of a published study. The initial pilot project was launched in February 2014 with over 30 journals agreeing to ask authors during submission to provide RRIDs for antibodies, for genetically modified animals, and for software tools/databases. The project established a centralized portal (http://scicrunch.com/resources) that aggregates accession numbers from authoritiave registries for each of these types of resources and enables searching by identifiers.

Digital Object Identifiers (DOIs) represent another broadly-used class of identifier. DOIs are widely used in publishing and there are ongoing efforts to leverage their capabilities for software citations. One significant initiative is a collaboration between Mozilla, figshare, GitHub, and Zenodo that allows software developers to easily mint DOIs for their tools[2].

Ultimately, successful use of unique identifiers for software requires not only a structure, but also social adoption. No tracking system will work unless authors begin properly citing the software used in their research. Funding agencies have set a precedent in driving public access among grantees that may be useful in encouraging adoption. Pilot projects have shown that both journals and authors recognize the need for such a system and are willing to adjust workflows to accommodate it.

Use cases

The combination of unique identifiers for software and the use of those identifiers in publications will enable the creation of a rich dataset of software relevant to the research community. This dataset will be captured through the Software Discovery Index, consolidating data on software packages and their use. The Index would not be a new repository for software code, but rather a resource collecting data from many repositories, publishers, and other sources. This Index, though a highly tangible aspect of the effort, is likely to be highly susceptible to feature creep. It is essential, therefore, to select a subset of relevant features appropriate for the next phase of implementation. Not all of the features described here are likely appropriate for the next phase of implementation.

One obvious function of the Discovery Index is to index metadata

describing software. The metadata selected for inclusion must be carefully considered for relevance to various users. A selection of metadata fields are listed in Appendix 1. Appropriate metadata, identified by a broad community, should be useful for a range of systems both within and beyond NIH.

Aggregating data across multiple sources will help make software from multiple sources more comparable, enabling the calculation of software ratings and utility scores. These scores should depend on a range of criteria, including such factors as citations in the scientific literature, documentation, codebase activity, and user community vitality. Though capturing metrics on these attributes will require a significant development effort, this information has the potential to be of tremendous value to the scientific community. Much additional metadata is likely to be of value to the research community even if it is not currently amenable to quantitative comparisons between different software. Even without quantitative measures of software reliability, there are many reliability indicators that would be worth tracking. Completeness of documentation can be measured at several different levels, including the ability to install the code, or understand the impact of changing run-time variables. The presence of unit or integration tests are critically important, but currently impractical to rigorously measure. Inclusion of benchmarking results can help describe the operation of software. Having metrics such as these available, whether or not they are factored into a reliability score, will help researchers selecting software.

As a secondary benefit, exposing the results of various measures may encourage developers to invest in documentation, unit tests, benchmarking, and other best practices. With benchmarking in particular, the Index has the potential to simplify benchmarking by providing gold standard datasets against which benchmarks could be run. Ultimately, if the Index captures multiple

measures of software utility and quality, it should be possible to offer certification levels for software to signal compliance with various best practices. Such recognition, already being examined[3], could both help users wishing to find software and encourage software developers to follow those best practices.

Supporting reproducibility is a critical need and an area where the Index is likely to grow significantly with time. While initially citations alone would be unlikely to provide enough information to enable full reproducibility of published results, they do provide a framework for documenting not only the software used, but also the environment and parameters. In time, one of the major contributions of the Index may be in improving the reproducibility of published analyses.

Finally, to maximize the utility of this index, it is critical that its information be exposed via both a website and an API. The website should provide a convenient point of entry that allows various faceted searches and browsing software tools. This website is expected to evolve significantly as this effort matures, but it is important that the first iteration is usable and streamlined. Over the long term, however, the API is likely to be at least equally important as the website. It is likely that websites serving specific research communities will wish to provide their own filtered views of the data, and this should be encouraged through an API. Moreover, other resources such as Synapse, GitHub, Zenodo, figshare, SciCrunch/NIF, and others may wish to expose their software to this index, and the API should enable this as well. A thoroughly-documented and usable API for both providing data to this aggregator and retrieving data from it will be critical to its long-term success.

The use cases briefly outlined here provide only a small slice of the likely eventual functionality for a successful Software Discovery Index. As the index and

citation patterns expand, the novel datasets generated should be further uses not yet planned.

Complementarity with the Data Discovery Index

The Data Discovery Index (DDI) is a NIH Big Data to Knowledge (BD2K) project. The DDI will enable investigators discover, access, and cite biomedical big data. The DDI aims to cut across disciplines and provide an index that will broadly serve across all NIH investigators. We expect that the Software Discovery Index will be fully compatible with the DDI, with the goal of allowing DDI and Software Index objects appearing in electronic journal articles, and enabling comprehensive retrieval of both data and the software that is utilized to analyze or produce these datasets.

B. CHALLENGES AND REMAINING QUESTIONS

The framework proposed above consists of endorsing identifiers, collaborating with publishers, and developing a Software Discovery Index. Each of these tasks carries particular challenges. Some of these challenges must be solved now, while others should be considered and possible solutions proposed in the first iteration of this effort.

Two important early needs are to define the scope of this project and to provide a help desk. A key element of defining the scope of this initial project will be deciding what software should be covered. Though ultimately the system should not limit itself to NIH-supported software, a limited scope could be useful in the early stages. The help desk should help users navigate this system. Software developers, in particular, will need guidance on obtaining unique identifiers for their software and in crafting useful metadata files. Other users are

also likely to benefit from assistance locating, citing software, and tracking software.

Defining relevant software

Defining relevant software will be a challenge for this effort. Biomedical researchers use a tremendous amount of software that does not need to be captured in this effort. It is relatively clear that no citation is necessary for a text editor, even its features may have greatly helped a researcher. Likewise, it is relatively clear that the statistical analysis package used to analyze a dataset should be cited. A great deal of biomedical software, however, falls between these two extremes. Many researchers use a few lines of script to store parameters for command line programs. Sometimes, those simple scripts grow and become tools in their own right. In other cases, exploratory tools may have been critical for generating initial hypotheses, but not have been used to generate or analyze the published data. It will be important to balance completeness with navigability. Multiple avenues exist for achieving this and it will be important to select with care the approach for the project.

Integrating with other repositories

Aggregating data from multiple sources, though it opens major opportunities to improve software development and design, also requires integration with multiple repositories. The goal of the Software Discovery Index is not to replace existing software repositories, but rather to pull as much information from them as possible and to present that information in a consistent and useful form. This is similar to the role that PubMed plays for journals – PubMed aggregates the results and provides them in a consistent form, but does not oversee curation or peer review. Similarly, the repositories

will likely be the ones that ensure standard metadata and provide some degree of curation. This will require strong relationships with multiple existing repositories and a willingness to work with new repositories that contain relevant software.

Evaluating progress and distinguishing this from other efforts

This system is not the first attempt to create an Software Index for NIH-supported software, and we should learn from prior efforts. For example, NIH dedicated significant support to the BioSiteMaps effort. Numerous researchers have created lists of significant software in their own fields, for example curated projects like the Neuroimaging Tools and Resource Clearinghouse (NITRC). Finally, the RRID project and underlying Neuroscience Information Framework Resource Registry have been broadly populated and are used by a broad community. It will be critical to consider what distinguishes this effort from previous efforts as well as any overlap in order to define metrics for success. Some of the key features of this effort that distinguish it from prior efforts include the automated indexing of software, the integration with multiple registries, and the provision of APIs enabling the creation of community-specific user interfaces.

C. IMPLEMENTATION ROADMAP

Below is a preliminary list of milestones involved in implementing a Software Discovery Index:

- Define a checklist for the Minimal information about software (see Appendix 1).
- Develop and implement methods to assign unique identifiers to

software systems, leveraging existing approaches where possible.

- Establish and maintain an API for searching, browsing, entering data, and interacting with journals.
- Establish and maintain a facile and streamlined website with search and browsing capability for software tools (see Appendix 2 for a selection of use cases).
- Partner with journal editors to implement the selected unique identifiers in electronic publications. At a minimum this would require identifying relevant journals, developing file formats and APIs for data exchange, and developing documentation for authors.
- Establish and maintain an Advisory Working Group of international members of the user community, software developers, software repositories, other relevant electronic repositories, and journals.
- Engage effectively with the Data Discovery Index.
- Implement performance metrics for the Software Discovery Index (see Appendix 3). Summary results should be made available publically while detailed results should be made available to funders and advisory groups.
- Promote the Software Discovery Index in journal editorials, conferences, social media, and scientific publications.

D. CONCLUSIONS

The Software Discovery Index Meeting of May 12-13, 2014 was a valuable forum for many important discussing. Participants agreed that the unprecedented abundance of electronically encoded information such as `omic data, imaging, and EHR, the software required to manage and understand this data is has also become increasingly critical to biomedical research. Properly documenting this software is critical.

Meeting attendees also agreed that software is no longer incidental to the

data; the systems used to produce and analyze raw data must be indexed to support analysis reproducibility. Due diligence towards reviewing existing projects, including the RRID project, will provide valuable insights into how this system should operate. Assigning universal locators to software enables significant improvements in the processes for finding, citing, and reusing software. This workshop proposed the use of unique identifiers for software packages, the formation of collaborations with Publishers to track software used in publications, and the creation of a Software Discovery Index to provide information on software packages. If successful, an implementation of these three efforts would benefit software developers, software users, journal publishers, and funding agencies.

E. APPENDIXES

Appendix 1: Minimal information about software (MIAS)

A common set of metadata fields are critical for useful indexing. If this effort only provides refined free-text searching capabilities, it will not be a major improvement over currently-available resources. It is necessary, therefore, to define a key set of minimal fields can that provide maximum value. At the workshop, the following fields were described as candidates for inclusion in this list:

- Persistent identifier
- Software title
- Software version
- Software license
- Links to code repository

- Human-readable synopsis
- Author names and affiliations
- Terms to describe software objectives or functions, and/or the following two bullets (controlled by an appropriate ontology)
- Formats for data inputs and outputs
- Platform, environment, and dependencies
- Associated grants and publications

Appendix 2: Use cases

- Developer: A developer registers their software, she is able to track and quantify all use of their software in scientific publications, through comprehensive and accurate citation of the index-associated identifier. With the ability to find similar types of software packages (e.g., other assembly programs), she would also identify benchmarking data sets, and other related software development efforts.
- User: An NIH funded researcher is seeking software for analysis. They are able to identify the most appropriate software relevant for their study on their data on their computer systems and objectives, and be provided with all information necessary to locate, obtain, and deploy the software.
- NIH: A program officer can identify both the creation and the use of all software funded by a grant they have awarded, analogous to how they can track all papers and citations to those papers funded by a grant they have awarded. They can also identify similar or overlapping products. Review panels can assess software choices in funding proposals and data management plans.
- Publisher: A publisher can associate software with their publications during & for peer review and upon publication for citation. They can also pull & display metrics related to all the research objects surrounding the

article, including software based on the software identifier.

Appendix 3: Metrics and milestones

It is critical to define metrics for this effort. These metrics should be evaluated both in absolute terms and in relative terms, monitoring the growth with time. These metrics are particularly significant because this is not the first effort to make biomedical software more accessible to researchers. This effort will face many of the same challenges faced by previous efforts and it is critical to closely monitor whether it is accomplishing its purpose. Specific metrics proposed for the initial effort included:

- Number of developers contributing software
- Number of software records created
- Software identifiers appearing in and extracted from publications
- Links from publications to software records
- Links between indexed software and other resources, people, and data
- Annotation of existing collections of software packages (e.g., Bioconductor)
- The number of interoperating resources, including repositories, aggregation resources, and user forums
- The use of the APIs to re-package the data for specific use cases
- The proportion of NIH-supported software tracked by the software discovery index

Tracking these metrics will provide insight into the progress of this effort. Progress against these milestones should, wherever possible, be evaluated against milestones. Specific milestones could include the fraction of NIH-supported software included in the first year, the time for machine-actionable links to software in PubMed, and the time for API establishment.

Appendix 4: Existing software indexes

There are numerous existing software indexes serving specific communities, many of unrelated to biomedical researchers. Some of the challenges that this effort will face have also been addressed by these indexes.

There are existing package management systems, notably RPM and dpkg for Linux. These tools facilitate the installation, upgrading, and uninstallation of software packages. Both systems have ways to unambiguously track software packages and ways to aggregate data on those packages, significant requirements articulated at this workshop. It is also interesting to note that these are both low-level tools and that users typically interact with them via higher level interfaces. This sort of modular model fits well with what was described at the workshop, with its focus on providing an extendable framework that others can leverage. This model differs from that famously employed in the Apple App Store and Google Play, where the software is directly hosted and managed on the index itself. The index, as described here, would be a lower-level construct that supports various package management functions but does not itself perform those functions.

SciCrunch/NIF/RRID: The Neuroscience Information Framework, (http://neuinfo.org) is a project of the NIH Blueprint Consortium that has been surveying, cataloging and federating data and resources (tools, materials, services) of relevance to neuroscience since 2008. It has maintained and populated the NIF Registry, a high level metadata catalog of research resources, currently comprising over 11,000 resources, and tracked them over the past 6 years. Through its unique data ingestion and query platform, it has created a search engine for data that searches across over 200 independent databases comprising over 800 million records. Although NIF was developed for

neuroscience, it has expanded well beyond primary neuroscience resources to biomedical resources as a whole. Thus, the software sitting behind NIF was rechristened SciCrunch. SciCrunch allows different communities, e.g., NIF, to use the same infrastructure and data sources to create their own communities. The SciCrunch Registry provides the unique identifiers for software tools and databases for the Resource Identification Initiative (RRID project; http://scicrunch.com/resources). SciCrunch aggregates software tools from multiple repositories, e.g., NITRC. It utilizes authoritative identifiers where possible, and assigns an identifier when the source repository does not.

Participation in the RRID project was voluntary, i.e., not a condition of publication, and was requested by the journals through an email request to the author. The project deliberately did not require journals to modify their journal submission system in order to allow broad participation in the project. To date, ~50 papers have appeared from 11 different journals that use RRID's. Over 200 RRID's have been reported. The FORCE11 working group is collecting data regarding the use of RRIDs in the literature and is making it freely available here. The error rate to date is ~7%. Papers using RRID's can be retrieved from Google Scholar by searching for a particular ID (**Figure 2**). A resolving service has also been developed such so that 3rd party tools can utilize the RRID's to link to a resolvable record as well as to map identifiers where needed. Automated routines based on NLP as being developed to recognize RRID's and to suggest appropriate RRID's based on the resources described. Currently, RRID's are only assigned at the level of the software tool or data resource, that is, it does not specify versioning information. This was a calculated decision, as the primary objective of the RRID pilot was to determine if unique identification of software and other resources would be achievable by publishers and authors. The RRID project in the future will aim to include more detailed and machine actionable

information as per outcomes of these and other related community discussions.

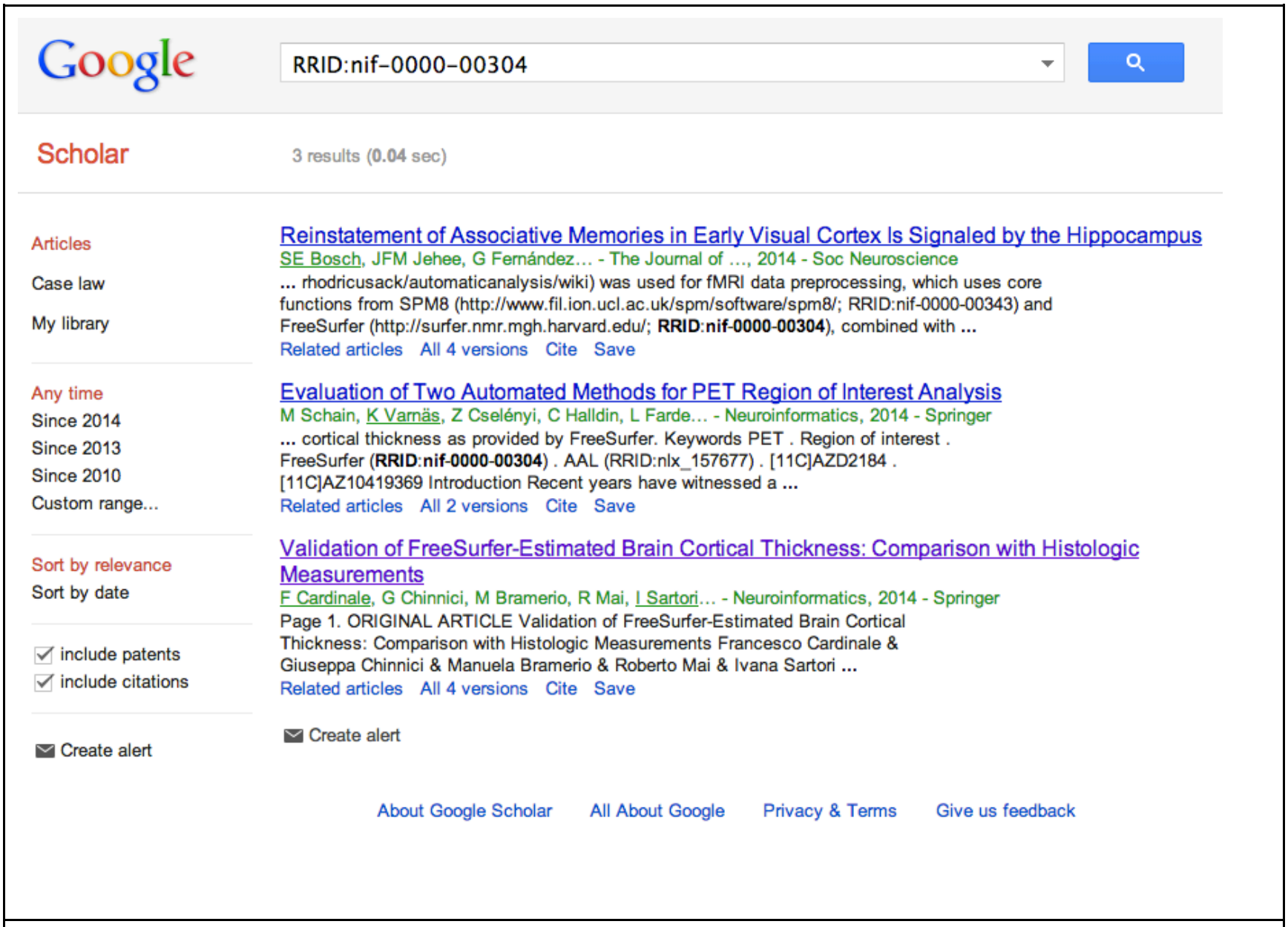


Figure 2: Google Scholar results showing papers citing the RRID for FreeSurfer.

Neuroimaging Informatics Tools and Resources Clearinghouse (NITRC): Since 2006, the Neuroimaging Informatics Tools and Resources Clearinghouse (NITRC) has provided a comprehensive support infrastructure for resources, including software, in the neuroimaigng domain (including MRI, PET, EEG, MEG, SPECT, CT and optical neuroimaging tools and resources). NITRC fosters a user-friendly clearinghouse environment for the neuroimaging informatics community. NITRC's goal is to support researchers dedicated to enhancing, adopting, distributing, and contributing to the evolution of previously funded neuroimaging analysis tools and resources for broader community use. Located

at www.nitrc.org, NITRC promotes software tools, workflows, resources, vocabularies, test data, and now, pre-processed, community- generated data sets (1000 Functional Connectomes, ADHD-200) through its Image Repository (NITRC-IR). NITRC gives researchers greater and more efficient access to the tools and resources they need, including: better categorizing and organizing existing tools and resources via a controlled vocabulary; facilitating interactions between researchers and developers through forums, direct email contact, ratings and reviews; and promoting better use through enhanced documentation.

nanoHUB.org: Starting in 2002, the NSF-sponsored Network for Computational Nanotechnology established a web site at nanoHUB.org to support the National Nanotechnology Initiative. Any user within the community can contribute a simulation/modeling or analysis tool to this platform. Tools are not only cataloged, but hosted, so that any user can run the tool through the web via the click of a button–without having to download or install any software. In 2013, more than 13,100 users launched some 500,000 simulation jobs using more than 340 different simulators contributed by the community and installed on nanoHUB. These tools have been used by 22,649 students across 1,165 courses at 185 institutions. nanoHUB also hosts more than 4,000 other resources—including seminars, tutorials, animations, and even complete courses—that help to document the tools and educate new users. In the last 12 months alone, nanoHUB served more than 300,000 unique users with this content, and that number has been doubling every 18 months. In June 2011, the National Science and Technology Council's Materials Genome Initiative for Global Competitiveness highlighted nanoHUB as an exemplar of "open innovation" that is critical for global competitiveness. The HUBzero software that powers nanoHUB.org is available as open source, and more than 60 other projects have

used the same software to create similar “hubs” for their own scientific community.

Bioinformatics Links Directory: Initiated in 2001 at the University of British Columbia as a tool to manage “links” for a bioinformatics core facility (akin to “Pedro’s list”) Francis Ouellette’s group has maintained this bioinformatics links directory to its current maturity which is now a resource with more than 1,400 links providing provenance for a given resources, databases and tools. This is now maintained at bioinformatics.ca, but will be migrated to a “.org” URL in the near future. **http://bioinformatics.ca/links_directory/**

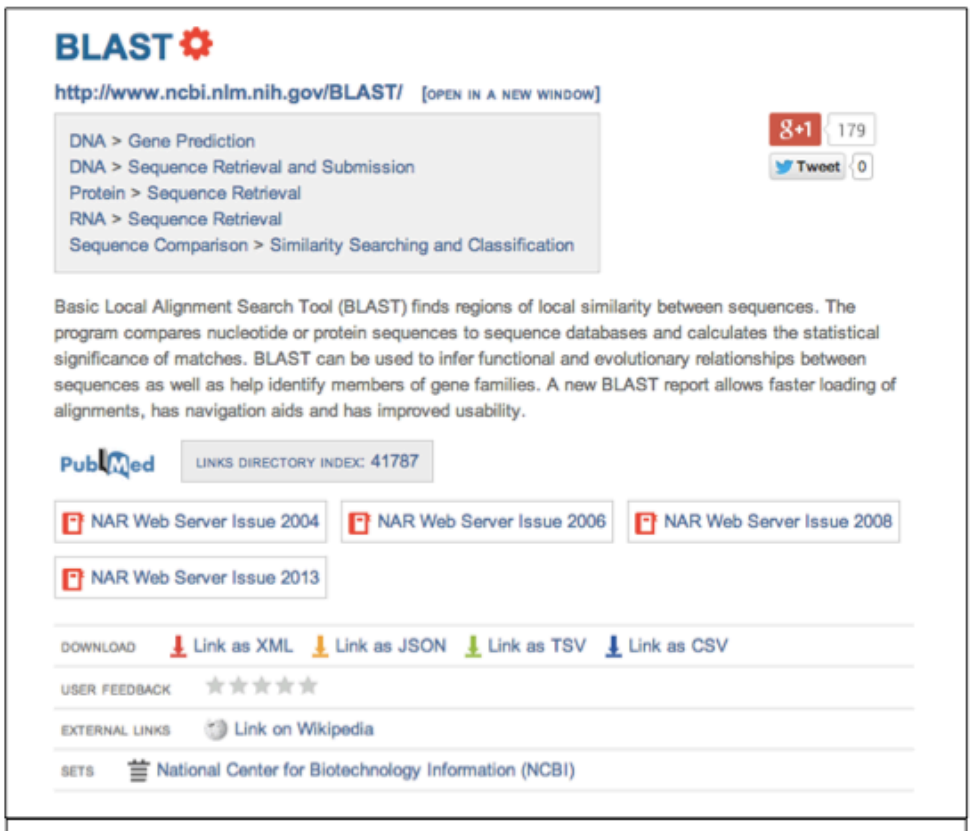


Fig 3 Links directory result for NCBI’s BLAST
http://bioinformatics.ca/links_directory/tool/9622/blast

Appendix 5: Workshop participants

Below is an alphabetical list of the individuals who participated in the workshop and the preparation of this report.

Vivien Bonazzi (NIH)

Phil Bourne (NIH / OD)

Steven Brenner (University of California, Berkeley)

Robin Brown (NIH / NCI)

Ishwar Chandramouliswaran (NIH / NCI)

Jennifer Couch (NIH / NCI)

Sean Davis (NIH / NCI)

Leslie Derr (NIH / OD)

Asif Dhar (Deloitte LLP)

Luke Dunlap (Deloitte LLP)

Kevin Eliceiri (University of Wisconsin – Madison)

Leigh Finnegan (NIH / NHGRI)

Ian Fore (NIH / NCI)

Melissa Haendel (Oregon Health & Science University)

Martin Hammitzsch (Deutsches GeoForschungsZentrum GFZ (German Research Centre for Geosciences))

Tram Huyen (NIH / NIAID)

Daniel S. Katz (National Science Foundation (NSF))

Jennie Larkin (NIH / NHLBI)

Jennifer Lin (Public Library of Science (PLOS))

Peter Lyster (NIH / NIGMS)

Ron Margolis (NIH / NIDDK)

Gabor Marth (Boston College)

Maryann Martone (University of California, San Diego)

Michael McLennan (Purdue University)

Martin Morgan (Fred Hutchinson Cancer Research Center)

Francis Ouellette (Ontario Institute for Cancer Research)

Vinay Pai (NIH / NIBIB)

Andreas Prlic (University of California, San Diego)

Will Schroeder (Kitware, Inc.)

Michael Sherman (Stanford University)

Heidi Sofia (NIH / NHGRI)

James Taylor (The Johns Hopkins University)

Kaitlin Thaney (Mozilla Science Lab)

Chris Wellington (NIH / NHGRI)

Mike Kellen (Sage Bionetworks)

David Kennedy (University of Massachusetts Medical School)

Owen White (University of Maryland School of Medicine)

---

[1] ‘Metadata’ refers to the information model used to uniquely identify and minimally describe software entities. Community-driven standards would identify required versus optional metadata fields to instantiate entities in the Index.

[2] https://guides.github.com/activities/citable-code/

[3] Neil Chue Hong, Minimal information for reusable scientific software (http://dx.doi.org/10.6084/m9.figshare.1112528).

55 thoughts on “report”

Martin Fenner
October 7, 2014 at 7:57 pm

The workshop and the report come very timely, as the scholarly community is working in numerous initiatives to make scientific software more discoverable and to give software authors due credit. The importance of persistent identifiers and a central searchable index with metadata can’t be stressed enough and I applaud this group for this activity. As always, the devil is in the details, and I have a number of comments/concerns regarding the report.

Persistent identifiers are a social contract. For them to work you need a set of

agreed principles between the organization issuing the identifiers and the person or organization using the identifier with the software. For journal articles (publishers) and datasets (data centers) these roles are clear, but it is not clear to me from the report of who would do that for the software. A software developer, or a code repository (at least popular commercial repositories such as SourceForge or Github) is the wrong partner for this, as they have other priorities than thinking about how to keep the software available for the next 20 years or more.

One possible approach is that data repositories such as figshare or Zenodo take that role, building on the work they have already done. Or we see more specialized software repositories evolve that cater to those needs and with additional features compared to traditional data repositories. I see one of the biggest gaps right now in repositories for long-term preservation of software, using an intelligent integration with code repositories. Without this partner with a long-term perspective persistent identifiers don't work.

Another consideration is duplication of effort. I would argue that it would be a bad idea to invent a new persistent identifier just for software. I personally think that DOIs are perfect for this, and I would wish that the report would use stronger language to support DOIs for this activity. (DataCite and CrossRef) DOIs have required metadata (e.g. version, license, contributors, related identifiers) stored in a central, searchable metadata store, and in particular DataCite has considered software in its metadata schema. DOIs also work with a lot of existing infrastructure, the extra effort to include special considerations for software would be much smaller than building a new infrastructure for persistent identifiers for software.

Lastly, software should be cited the same way as books, articles and data, with

the citation appearing in the references list (the citation to the software, not only the citation to a paper describing the software). This is not only something that authors, readers, publishers are familiar with, but this makes it much easier to actually find these software citations. Not only are software citations within the body text very difficult to find if the content is not open access, but the variations in formatting (software name only or also link, in many different places of the manuscript, etc.) make it very hard for automated tools to find these software citations.

Something that DataCite is not providing is a Software citation index, and a service build based on DOIs should be developed to fill that void. This service should also cooperate with CrossRef, as we hopefully will see an increasing number of software citations in the reference lists of scholarly papers. These software citations should not be extracted only from journal articles, but wherever they happen, most importantly other software packages and in association with datasets. Another important activity that is missing, and is mentioned in the report, is a listing of appropriate software repositories (both code repositories and repositories for long-term preservation). Databib and re3data, together with DataCite, are doing this for data repositories, and a software repository discovery index can either be a part of those services, or a separate activity.

↳ **Reply**

Vijayaraj Nagarajan
October 8, 2014 at 12:59 pm

Excellent point about the potential problems with relying on a third-party repository.

NCBI has the ability and is already successfully performing archiving of genome scale data. If not for all of the software, why not create a small facility under the hood of NCBI to archive the public funded software ?

Such a system could be more robust, with easy integration in to other data components of NCBI. This would also leverage on the expertise that NCBI has acquired all these years in creating an excellent archiving ecosystem. I don't think infrastructure resources would be of a concern for such an in-house repository, considering its scope and size in relation to the existing archives of NCBI.

↳ Reply

Steven Salzberg
October 8, 2014 at 9:09 pm

This is a good suggestion: NCBI is already set up to handle repositories, and it would be a simple matter for them to create a small software repository. I see no need for anything more elaborate, if we even need this at all. Most good software gets published and we have papers to cite already.

↳ Reply

Istvan Albert
October 7, 2014 at 7:59 pm

What seems to be missing from the MIAS fields is any mention of minimal user support. Shouldn't there be some requirement that ensures that there are

avenues available to assisting someone having problems or needing advice?

This is why Biostars exists https://www.biostars.org/ and the hundreds of thousands of visitors that the we get monthly indicate that the problem of software support is just as important as that of categorizing it properly.

↳ Reply

W. Trevor King
October 7, 2014 at 9:26 pm

I think encouraging researchers to cite software they used to conduct their research and analysis is important, but I'm happy leaving the threshold up to the researchers (e.g. not listing their text editor, but listing their experiment-controls and stats packages with version numbers). It also makes sense to require publically-funded software to be archived somewhere to ensure it will be accessible in the future. Beyond that, I don't see much need for additional tooling. Won't users be able to find software by looking through the papers written by folks doing similar research or through a generic search engine?

I don't see how any of this will help "Review panels can assess software choices in funding proposals and data management plans." Assessing software seems much more complicated and subjective than something you can condense into some indexed metadata. Is there an active community using and developing this software? How responsive are the maintainers to bug reports and patch submissions? I think assessing that sort of thing needs a more organic touch.

↳ Reply

Vijayaraj Nagarajan
October 8, 2014 at 3:05 pm

This is why it might of great help to create our own first of its kind in-house biomedical software repository. This would enable us to put that “organic touch”…. as being done in the NCBI GEO repository and indexing. NCBI does an amazing job of giving this organic touch to all of the submissions made to this repository, which has made GEO an overwhelming success.

↳ Reply

Nicholas Provart
October 8, 2014 at 1:43 pm

Seems like a worthy initiative but I think the buy-in from publishers is important, otherwise the effort will end up like http://metadatabase.org/ (Bosler et al., NAR, 2012; doi: 10.1093/nar/gkr1099), i.e. not being updated. The creation of a MIAS is a good idea!

↳ Reply

Vijayaraj Nagarajan
October 8, 2014 at 3:00 pm

While trying to rope in the publishers, to start with, it might help to require all NIH funded software to be deposited and to obtain an ID from this software indexing system, for inclusion in their manuscripts, like “omics” data are being deposited in NCBI GEO, obtained an accession and included in manuscripts.

↳ Reply

Francis Ouellette
October 8, 2014 at 6:16 pm

I think the publishers **and** the funders will be drivers in this process. I think NIH will be on board 🙂

↳ Reply

Prash Suravajhala
October 17, 2014 at 9:39 am

Thank you Nicholas, one and all.

On Metadatabase, we are working on it 🙂

1. Sometimes the Bioproject@NCBI and the accessions created for genome sequencing projects are NOT cited. It would be a good idea, if the two links come under one common repositories.

2. A wiki repository for all known unknowns (uncharacterized proteins, regulatory regions, KIAAs etc.) would be wonderful! These submissions could be linked to GEO database.

3. My suggestions would be to have a wikis enabled real time chat created for all publicly funded software developers. There are several software that are created under the umbrella of ‘Open Access” which could support this cause.

Prash Suravajhala recently posted…Prash

↳ Reply

Vijayaraj Nagarajan
October 8, 2014 at 2:41 pm

“The goal of the Software Discovery Index is not to replace existing software repositories”..

Do any good biomedical software repository even exist currently, that we are talking about replacing ?

If we are talking about GitHub/SourceForge… like repositories, should we not note that they are a general software repository, with a fraction of the software relevant to the biomedical community ?

Since there are no good existing biomedical software repository, why not create one ?

As stated in my reply to Martin, NCBI has excellent infrastructure and capabilities to host such a repository. NCBI has been doing the genome scale data archival for years in an efficient and useful way. Integrating a software versioning system to the already existing NCBI ecosystem might be a feasible option.

It could be complementary to the existing general purpose software repositories and also provide a tight integration to the Data Commons system at NCBI.

“This is similar to the role that PubMed plays for journals – PubMed aggregates the results and provides them in a consistent form, but does not oversee curation or peer review”

It might help to consider modeling the software index based on a more similar data index, than based on the PubMed.

Sure, PubMed does an excellent job of aggregating the publications. But, I see journals as a totally different ecosystem compared to software repositories. Software repositories are more like Data repositories, their primary goal is archiving, not peer-reviewed publishing.

Also, please consider that the PubMed is only dependent on the publishers submission of data. Whereas, this software index would have to depend on the individual authors (like data submissions to NCBI GEO). So, it might be worth considering to model this index based on the experience NCBI has gained in archiving genome scale data (like GEO).

↳ Reply

Vijayaraj Nagarajan
October 8, 2014 at 3:11 pm

When trying to define the “software” and the “MIAS”… it might help to address these questions…

1. will this index include only software that has an associated publication or grant number ?
2. will this index include web databases and web applications that do not have anything deposited in a separate repository (a whole lot of biomedical tools are of this nature) ?
3. how does this index handle tools that are posted in authors websites or third party web sites and not in a standard repository?

↳ Reply

Michael Kurtz
October 8, 2014 at 3:49 pm

ASCL/ADS The Astrophysics Source Code Library (http:www.asdl.net) was founded in 1999; The software is stores and provides access to is indexed by the Smithsonian/NASA Astrophysics Data System (ads.harvard.edu), astronomy's central digital resource for text based indexing and access. Currently 930 ASDL software entries are indexed by the ADS. A recent description, and the papers leading up to it are listed:
2014ASPC..485..477A The Astrophysics Source Code Library: Where Do We Go from Here? vy Allen, A., et al
1999AAS…194.4408N The Astrophysics Source Code Library by Nemiroff, R. J., and Wallin, J. F.
1994ASPC…61…41H An Astronomical Software Directory Service by Hanisch, R.J., Payne, H. and Hayes, J.J.E.
1989lisa.conf..215R An Astronomical Software Documentation Collection at the Smithsonian Astrophysical Observatory, by Rhodes, C., Kurtz, M.J., and Rey-Watson, J.E.

↳ Reply

Vijayaraj Nagarajan
October 8, 2014 at 4:40 pm

ADS has 930 software entries.
A quick search for "bio" entries in github, got me 6,353 entries.
Similar search for "bio" entries in sourceforge got me about 3,225 entries.

Looking at the above numbers, looks like it might not be appropriate to compare the proposed index with PubMed (where there are millions of entries).

Also, even if we expect the number of “bio” software to grow in to tens of thousands, that number would still be significantly low, warranting its own well curated biomedical repository.

↳ Reply

Herbert Sauro
October 8, 2014 at 3:50 pm

We need to distinguish here between those problems that are not I fact problems and others which are real. The non-problem in my opinion is unable to find software. Anyone with an ounce of common sense can find software. In systems biology which is my field, I have not problem in finding software I need. A combination of google searches, talking to colleagues and scanning the software lists, eg at sbml.org, or the standard code repositories is more than sufficient to find out what’s out there. I am also involved in synthetic biology and exactly the same comment applies there. I feel find the software is not a significant issue.

As for the other issues cited:

1. “Software users have difficulty identifying the most appropriate software for their work.” The answer to this issue is that we are supposedly professionals. If we cannot figure what tools to use in out work there are much bigger problems at work.

2. “Journal publishers lack a consistent way to handle software citations or to ensure reproducibility of published findings.” This I think is a genuine problem, since reproducibility is at the core of the scientific method. Far too much computational science is not reproducible due either to missing software or more worrying lack of transparency in relation to the algorithms used.

3. “Funding agencies struggle to make informed funding decisions about which software projects to support” Another valid problem but part of this could be delegated to a combination of usage statistics and peer review. The really difficult question to answer are those tools which are very rarely used but are critical to research and there are probably many of those.

↳ Reply

Todd Smith
October 12, 2014 at 5:54 pm

I agree with these points. Indeed on number three. Basic research is about long tails and highly specialized tools that support these endeavors will be low use due to the nature of the users. But the work and insights can have high value later on. We all want an app store concept, but must not be overly simplistic in interpreting the data.

I would also suggest that commercialization activities be factored in for informing funding decisions. Have the software tools been licensed to commercial organizations – e.g. do they have income from other activities? What is the nature of the license exclusive, non-exclusive. This might be interesting data to have and understand.

↳ Reply

Ilya Goldberg
October 8, 2014 at 4:15 pm

As I was reading the report, I kept wondering what problem this was trying to solve that was not already solved by systems like GitHub.

A lot of talk about unique identifiers and some kind of clearing house for generating these. It would have been informative to see/understand how Git actually works, since it solved this problem quite well, without relying on a central authority to issue globally unique software version identifiers (!). Git is used world-wide for mission-critical software, where misidentification of specific software versions could lead to disaster of far greater consequence than misidentifying software for biomedical research. It is very unlikely that a better or more trust-worthy system could be developed solely for the purposes of identifying software used in biomedical research.

Keeping track of software is by no means a new problem, and the specific demands of software used in biomedical research are not sufficiently different from other fields, including other scientific fields, to justify a separate effort.

What does not exist, and oddly, this report stops short of recommending, is an actual repository. A repository sponsored and hosted by the NIH (not necessarily by NLM), based on Git, would be something new, interesting and worth expending effort on. Importantly, it would be a compelling repository to actually deposit software like this into. If there is no repository, then this project will inevitably be limited to solving problems that have already been largely addressed elsewhere. Not recommending an actual repository is in fact more than a little baffling.

Its not like hosting a repository would be that hard – there are half a dozen open-source software repository frameworks to chose from (Fossil, Gitorious, Gitlab, Allura, etc.). Any of these can be forked and modified to enforce MIAS and issue DOIs that correspond to specific commits/tags/releases.

Now that that's done – lets talk about data repositories!

↳ Reply

Hugues Sicotte
October 8, 2014 at 6:52 pm

Good idea. I often recommend to bioinformatics developers to publish their software ahead of a journal publication, but people want a definitive link. Having a centralized search system and identifying system is a good step toward removing the main obstacle to more broad dissemination. However, it would be important that the system is version aware, tie litterature links to specific versions (or ranges of versions), and allow multiple citation link per version, .. especially since some software give rise to multiple publications over their lifetime.

Are you going to host tools code?

Why not have NCBI provide the resource?

↳ Reply

Eduard Hovy
October 8, 2014 at 7:55 pm

I read the report with interest and appreciation. Overall, it says everything that most people believe, and doesn't say anything particularly startling or controversially new. I am however not moved to believe that this is not Yet Another Well-meaning Attempt to Create a Universal Index. We are all (uncomfortably) aware of equivalent efforts. And we all use them sparingly, if at all, in our own daily lives. How much of your data/papers/software/etc. that YOU produced in the past 30 days did you upload to such a repository?

I believe we need to explore the possibility of an automatically grown index that could use unique identifiers (but needn't) and that could accept people's manual uploads (but needn't), and that provides for data and software functionality akin to what Google and Yahoo! do for text documents.

Some colleagues and I are currently trying to build such a thing, specifically for data (not yet software). Is there anyone who might be interested in helping try it out? What we'd ask from you is
– a taxonomy of your principal metadata terms, measurement units, etc.
– as many tables of data as you can give us (in xml, csv, or spreadsheet format, or in clean pdf) — either given to us to download, or preferably hosted by you on your own server, just giving us pointers and access

What we'd like to do is ingest each dataset (= table, say), compute the characterizing info for each part of it (= column or row, say), and store this in a google-like vector space that allows others to enter queries and (hopefully) find your data, if appropriate.

↳ Reply

Mihai Pop

October 8, 2014 at 9:23 pm

I read the report and the other comments made here and, while I certainly sympathize with a number of the points raised, I am uncertain that a software index would truly solve the relevant problems in our field. I would specifically like to touch upon a number of problems that significantly affect our community yet would not be addressed by a software index:

1. **Appropriate attribution.** One can argue that we already have mechanisms for citing software through the paper in which it was described, or a link to the repository where the software is hosted. There are bigger issues, however, related to this topic. First and foremost, scientists are not required (by the journals in particular) to cite software even when the mechanisms exist, and are often actually discouraged from doing so by arbitrary limits on the number of citations within a paper. As a result software citations get buried in supplementary material where they are not tracked by citation indexes or even search engines. This is a much more serious concern than the need to uniquely identify a particular tool, and should be forcefully addressed by the funding agencies and the bioinformatics community (journals are unlikely to change their operations unless forced to do so)
2. **Reproducibility of analyses.** While the issue of reproducibility and provenance is widely discussed in our community, and is critical for clinical applications, I think it hides a much more serious issue impacting our community. This is the issue of the correctness and validity of the analytical pipeline. A reproducible wrong result is still wrong. What our community needs is the enforcement of sounds software engineering practices, such as the clear documentation of the algorithms and code, the development of validation metrics, and the active testing of the code. Instead of simple meta-data about a

software package, we need to be able to fully understand what the package does and how we can evaluate that the package actually does what it promises. For most bioinformatics software, even the most heavily used and cited, we, as a community, lack such an understanding.

3. **Novelty and Useability.** One of the issues implied by the report is that many software tools never end up being published (the fastx toolkit, for example) despite being heavily used. The real problem is that we currently do not have well accepted mechanisms for actually publishing software tools. Some journals prefer short application notes that provide little if any detail about what a particular tool does. Other journals set a high bar for publication,either favoring tools that do 'sexy' things or that significantly push a particular field forward. There are many tools that are good and useful but cannot be easily published, and a software index would definitely help them. On the other hand, however, there are many more tools that duplicate existing resources, or are poorly maintained, or are even incorrect. They already pollute the published research in part because we, as a community, do not have the mechanisms (or energy) to truly evaluate software tools especially if all we get to evaluate is a half-page summary of what a tool purports to do. An ungated software index would just exacerbate this problem and make it harder for users to find the tool best suited to their needs.

4. **Ease of finding the right tool for the job.** A major misconception in our field is that a simple 'Google-like' search can guide a novice biologist to the right bioinformatic tool. Much as biology (and biological manipulations in the lab) is complex, so is the analysis of the biological data. One should not use a bioinformatics tool unless they understand well what the tool does (and more importantly what it doesn't) and what its underlying assumptions and limitations are. Even 'simple' operations such as sequence alignment are broadly mis-understood in the community. I have heard numerous times of fast

aligners for short read data that are ‘as sensitive as Blast’, statement that, in its general context, is incorrect. Blast was designed to identify sequence similarities across long evolutionary distances – something that none of the short read aligners can do (in fact most do much worse, not being able to tolerate more than a couple of differences between the strings being aligned). If statements such as the one I mention here make their way in the ‘metadata’ associated with the software in the index they may incorrectly drive a naïve user trying to characterize a newly discovered protein to the use of software entirely unsuited for this task. I would also like to point out that the cultural image of the ‘bioinformatically naïve biologist’ is patronizingly incorrect, despite being quite prevalent. Most of the biologists I interact with, especially the new generation, have a deep understanding of bioinformatics and do not need a new ‘Google’ in order to find the software they need in their work.

↪ Reply

Jens Preussner
October 9, 2014 at 7:56 am

Great comment, Mihai, I agree with you on all points. I would like to emphasize you second and third point. For me, as a bioinformatician by training, it sometimes feels like being constantly in search for software. Sometimes, existing software needs some modification (i assume its open-source) and not having documented code and test cases mostly leads me to not use that software at all. A gated repository, as you put it, would definitely help. I also would like to see preprint services to accept software, like Steven foresees it, this would enable open code review by a greater community and hopefully reduce the amount of poorly documented software – by simultaneously enhancing reproducibility.

↳ Reply

Steven Salzberg
October 8, 2014 at 9:40 pm

I think Mihai Pop's comments eloquently state some of the real problems we face as methods (software) developers, and I second those remarks. I also want to second and re-iterate some of the other comments, such as Herbert Sauro's comments that "We need to distinguish here between those problems that are not I fact problems and others which are real." He goes on to point out that the report identifies several non-problems. In a similar vein, Trevor King comments that " I don't see much need for additional tooling."

I would add to these comments by expressing my concern that the report is calling for the creation of something we simply do not need. It appears to be a proposal for NIH to fund the construction of some kind of new software repository for bioinformatics software. Perhaps I am reading too much between the lines, but I see no need for valuable, ever-scarcer NIH resources to go into a resource like this. I would much rather see them go into supporting the scientists developing the software. As others point out above, we have SourceForge, GitHub, and other places to deposit software now. We also have the new bioRxiv (and arXiv) for papers, and with a very small push I'll bet that bioRxiv would be happy to take software submissions, which would then have their own DOI identifier.

I also agree with Herbert Sauro's comment in response to this statement in the report: "Software users have difficulty identifying the most appropriate software for their work." Sauro's response is: "The answer to this issue is that we are supposedly professionals. If we cannot figure what tools to use in out work there

are much bigger problems at work."

I think it's useful to discuss the issues of providing support for software development, and for obvious reasons (I develop software tools myself) I think this work deserves strong support. However, I do not see the need for, and do not support the creation of, a new system for "linking software, publications, and users."

↳ Reply

James Taylor
October 13, 2014 at 6:38 pm

I agree with you Steven (and Mihai). I was at this workshop, and my conclusions after all of our discussion were very similar. The workshop video is online . At 2:18:30 is when I say "maybe we shouldn't build anything at all". I think most of these problems are already solved well enough that this would not be a great way to spend (much) NIH money.

↳ Reply

Carl Boettiger
October 8, 2014 at 10:29 pm

The report is both timely and focused on key issues confronting our community, including the challenges of identifying, citing, and reusing software. The appendices do an excellent job in outlining key metadata, metrics, and use cases which help frame the discussion. The proposal does well to focus on the importance of identifiers and the creation

of a query-able metadata index for research software, but leaves out an essential element necessary to make this work.

This proposal sounds very much like the CrossRef and DataCite infrastructure already in place for academic literature and data, respectively; and indeed this is an excellent model to follow. However, a key piece of that infrastructure is missing from the present proposal — the social contract between repository or publisher and the index itself.

CrossRef provides unique identifiers for the academic literature (CrossRef DOIs), but it also defines specific metadata that describe that literature (as well as metrics of its use), and embed that information into a robust, query-able, machine-readable format. DataCite does the same for scientific data. These are exactly the features that the authors of the report seek to emulate.

Just as CrossRef itself does not host academic papers but only the metadata records, the SDI does not propose to host software itself. This introduces a substantial challenge in _maintaining the link_ between the metadata and the software itself. The authors have simply proposed that the metadata include “Links to the code repository.” If CrossRef or DataCite DOIs worked in this way, we would soon loose all ability to recover many of the papers or the data itself, and we would be left with only access to the metadata record and a broken link. DOIs were created explicitly to solve this problem, not through technology, but through a _social contract_.

The scientific publishers who host the actual publications are responsible for ensuring that this link is always maintained when they change names,

etc. Should the publisher go out of business, these links may be adjusted to point to a new home, such as [CLOCKSS](http://clockss.org). This guarantees that the DOI always resolves to the resource in question, regardless of where it moves. Should a publisher fail to maintain these links, CrossRef may refuse to provide the publisher any additional DOIs, cutting it off from this key index. This is the social contract. Data repositories work in exactly the same way, purchasing their DOIs from DataCite. (While financial transaction isn't strictly necessary for the financial contract, it provides a clear business model for maintaining the key organization responsible for the index).

Without such a mechanism, links in the SDI would surely rot away, all the more rapidly in the fast-moving world of software. Without links to the software itself, the function of the index would be purely academic. Yet such a mechanism requires that the software repositories, not the individual software authors, would be willing to accept the same social contract, receiving (and possibly paying for) identifiers on the condition that they assume the responsibility of maintaining the links. It is unclear that the primary software repositories in use to day (Sourceforge, Github, Bitbucket, etc) would be willing to accept this.

Data repositories already offer many of the compelling features of this proposal. Many data repositories accept a wide array of file formats including software packages, and would provide such software with a permanent unique identifier in the form of a DataCite DOI, as well as collecting much of the essential metadata listed in report's Appendix 1, which would then already be accessible through the DataCite API in a nice machine-readable format.

Using DataCite directly by integrating software repositories with data archives is thus a promising strategy to address this issue, as the report suggests in pointing to recent initiatives (“One significant initiative is a collaboration between Mozilla,
figshare, GitHub, and Zenodo”). At present, this approach leaves several major issues unaddressed. (1) The current implementation does not support versioning of the DOI, though DataCite DOIs do support it. (2) there should be support for an automated mechanism to update the archive when a new version of the software is released (e.g. a new tag added to the Git repo). (3) though DataCite already defines the “software” datatype, other metadata fields may need to be added, depending on the minimal metadata decided by the community. (4) More importantly, the data repositories need to be sure to capture this metadata, ideally from automatic extraction from the software, and report it in the appropriate fields of the DataCite schema.

Explicitly focusing on such collaborations between data repositories, software repositories, and the DataCite Index & API would be preferable to designing a system from scratch. Simply querying the DataCite API for software entries would then provide the desired features outlined in the report.

Other challenges would still remain. Software is more dynamic than data; tracking and citing versions correctly through changes, branches and merges would be difficult. Software also has a finite life-cycle, moving from proof-of-principle to mature to deprecated. But the report accurately acknowledges that the primary challenges would be the social ones. The ability to reliably archive, cite, and measure the use of software (including as it’s own citation line in published articles) helps provide the incentives for these cultural shifts.

↳ Reply

Pingback: Comments on Software Discovery Index Report | Think Links

Paul Groth
October 9, 2014 at 9:07 am

I think this report reflects both demand for and consensus around the issues of software identification and use in biomedical science. As a creator and user of scientific software both installable and accessible through Web APIs, from my perspective anything that helps bring transparency and visibility to what scientific software devs do is fantastic.

I would like to reiterate the comments that we need to build on what's already being used by software developers and is already in the wild and call out two main points.

##Package management

There was some mention of package managers but I would suggest that this should be a stronger focal point as much of the metadata is already in this management systems. Examples include:

– Python (https://pypi.python.org/)
– R (http://cran.r-project.org)
– Javacript (https://www.npmjs.org)
– Perl (http://www.cpan.org)

My suggestion would be that a simple metadata element could be suggested that would allow software developers to advertise that their work be indexed by the Software Discovery Index. Then the process would involve just indexing

existing package manages used. (Note, this also works for VMs. See https://registry.hub.docker.com).

For those that chose not to use a package repository, I would suggest leveraging Schema.org that already has a metadata description for software (http://schema.org/SoftwareApplication). This metadata description can already be recognized by search engines.

Software vs Software Paper

One thing that I didn't think came through strongly enough in the report was the distinction between the software and a potential scientific paper describing the software . Many developers of scientific code also publish papers that tease out important aspects of e.g. the design, applicability or vision of the software. In most cases, these authors would like citations to accrue to that one paper. See for example Scikit-learn [1], the many pages you find on software sites with the "please" cite section [2], or the idea of software articles [3, 4].

This differs from the idea that as an author of a publication one should reference specifically the software version that was used. My suggestion would be to decouple these tasks and actually suggest that both are done. This could be done by allowing for author side minting of DOIs for a particular software versions [5] and suggesting that authors also include a reference to the designated software paper. The Software Discovery Index could facilitate this process by suggesting the appropriate reference pair. This is only one idea, but I would suggest that the difference between Software and Software paper should be considered in any future developments.

This could also address some concerns with referencing APIs.

Summary

In summary, I would emphasize the need for the reuse of existing infrastructures for software development, publication of scientific results, and search. There is no need to “role your own”. I think some useful, simple and practical guidelines would actually go a long way in helping the scientific software ecosystem.

These comments are also posted at http://thinklinks.wordpress.com/2014/10/09/comments-on-software-discovery-index-report/

[1] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. 12 (November 2011), 2825-2830.

[2] http://matplotlib.org

[3] http://www.elsevier.com/about/content-innovation/original-software-publications

[4] http://www.biomedcentral.com/bmcbioinformatics/authors/instructions/software

[5] e.g. http://www.webcitation.org

↳ Reply

Pingback: Comments on Software Discovery Index Report | Web & Media VU

Murat Sincan

October 9, 2014 at 1:26 pm

1- The report is focused on free and open source or academic software but there are significant number of papers that utilize commercial software packages as well.

2- There is an emphasis on “finding or identifying” software and mentions issues related to being able to find software. I believe a non-curated index of tools is less useful and less needed than a curated resource. If the user is experienced they will generally know the software landscape and all available tools but inexperienced users may get even more confused when presented by a long list of packages without appropriate curation and annotation from the experts. The curation of software is best done by the user community. Platforms like https://www.biostars.org or seqanswers.com are good examples of these community annotations where experienced users provide links to existing software and discuss advantages, disadvantages, pitfalls and best practices with less experienced users. One approach may be to link out to existing discussions and community annotations for each tool whereas another option may be to replicate the functionality of these tools within the new proposed index. I prefer linking out to the existing resources for community annotation and feedback.

3- I believe it will be beneficial to use a system similar to MESH to annotate software tools for standardization of minimal and basic keyword annotations, a mechanism similar to NCBI’s current approach to annotate papers with MESH terms should prove useful for downstream analyses and categorizations.

4- Reproducibility is as important as discoverability of software and reproducibility depends on not only the software tool but many other factors [1], this index should aim to make it easier to reproduce results such as being able to index not only software but a whole frozen package that includes settings, random seeds , configuration etc. This way there will be a unique and persistent

identifier for a package that lets others to reproduce the results of a particular application of a software package.

5- Even though existing software repositories contain well established mechanisms for contacting developers and getting support it may be important for the proposed software index to have the contact information (email?) of the corresponding developer/author of the software package as part of the MIAS. This is not intended to replace the normal support and ticketing mechanisms but will serve as a quick and easy communication channel between interested users and developers. The developers may then direct the interested party to a richer channel of interaction.

References:

1- Sandve GK, Nekrutenko A, Taylor J, Hovig E (2013) Ten Simple Rules for Reproducible Computational Research. PLoS Comput Biol 9(10): e1003285. doi:10.1371/journal.pcbi.1003285

Reply

James Taylor

October 13, 2014 at 6:41 pm

Yes, from a reproducibility perspective, an archive is more important than an index. Even github repositories can be deleted. Archiving **everything associated with an analysis is crucial. I've always thought this would be a great service NCBI could provide, just a place to stick the tarball (or docker container) that goes with me paper. People now tell me institutional repositories are the answer…**

Reply

Peter Tonellato

October 10, 2014 at 11:55 am

Excellent report and recommendations. However, I agree with many of those that have already posted that the effort would leverage existing and standardized means of indexing . In particular, DOI has already dealt (to a degree) with the infrastructure required (see : dpi.org) in particular the metadata includes specifics concerning ‘media’ (including software). Examples abound in the discussion and DOI use in software including:

http://home.web.cern.ch/about/updates/2014/03/tool-developed-cern-makes-software-citation-easier

https://github.com/blog/1840-improving-github-for-science

http://software.ac.uk/so-exactly-what-software-did-you-use

All that said, the DOI system is not perfect and one can imagine a well thought group approaching DOI to more fully address some of the needs as described in the report and possible weaknesses in the current definition/scope/use of DOI in the software context.

↳ Reply

Todd Smith

October 12, 2014 at 6:04 pm

A point I do not see in the above discussion and probably should have some thought is related to workflows. We know that bioinformatics analyses are complex multistep processes where data are analyzed independently and in the

context of other data, aka databases. The version of the database, or any derivatives, is just as important to track to make research reproducible.

Presently these are things that are left to the researchers, but a registry concept could be beneficial.

↳ Reply

James Malone
October 13, 2014 at 6:14 am

I blogged a response to the document and ather than cut and paste the whole thing, links is here http://drjamesmalone.blogspot.co.uk/2014/10/it-needs-to-be-better-than-google.html

↳ Reply

Michel Dumontier
October 13, 2014 at 1:49 pm

I believe that the value proposition for a software discovery index is less about what software one should use to undertake a task, which is nevertheless useful, but should be more about ensuring the **reproducibility** of research results by providing:
i) *useful descriptions* of research software
ii) *enabling access* to these software

Given the difficulty in obtaining research software today, I think this report should go one step further and **recommend the establishment of a software commons** to archive scientific software. These software entries should contain a

description of how they can be used, along with sample datasets/unit tests to ensure they are functioning. It's important the kind of data being processed and produced (along with supported data formats) be included in the annotations. It should expand on software terminologies such as EDAM or the software ontology to do this, and enable the community to extend these and link them to standards such as those listed at biosharing.org.

I might also suggest that should be some description of approaches that will be taken to continuously harvest software from existing scientific software repositories (e.g.SBML.org). I also wondered whether web services, such as those listed at biocatalogue.org, would also be included. I believe that inclusion of services (which also provide adequate metadata about their version), would make a software discovery index even more powerful by enabling immediate execution. Finally, I worried about how additions to the index will be moderated? How will duplicates be found? How will forking of software be managed? What will the citation look like for that? You can imagine that there could be efforts to generate lightweight derivatives, just so they could be cited.

I also wondered about how we really need software to be stamped with a versioned identifier. While there are many ways that this can be achieved, I'll highlight our TrustyURI solution as one mechanism to obtain versioned identifiers from arbitrary distributions.

↳ Reply

Herbert Sauro
October 13, 2014 at 7:00 pm

A lot more emails since I last looked.

This is a bit of a side track but I'd like to reemphasize something Steven Salzberg mentioned in comment and that is the use of bioRxiv. Publishing software in journals is difficult. One reason is the expense. Of all the possible journals to publish bioinformatics software I only counted one that wasn't open access. You might be thinking, but open access is great, and it is until you get the $2000 invoice for publishing a 2 page description of your software. If a lab publishes 5 to 10 papers per years that 10 to $20k per year in expenses. I other words bioRxiv could serve as a cost effective repository for software papers, that part of the tool chain doesn't need reinventing.

Herbert Sauro recently posted…Systems Biology Group Supported by:

↳ Reply

Barbara Hill Meyers

October 14, 2014 at 2:48 pm

With Regard to the first use case in Appendix 2 & the dicussion of benchmarking "Developer: A developer registers their software, she is able to track and quantify all use of their software in scientific publications, through comprehensive and accurate citation of the index-associated identifier. With the ability to find similar types of software packages (e.g., other assembly programs), she would also identify benchmarking data sets, and other related software development efforts."

"As a secondary benefit, exposing the results of various measures may encourage developers to invest in documentation, unit tests, benchmarking, and other best practices. With benchmarking in particular, the Index has the potential to simplify benchmarking by providing gold standard datasets against which benchmarks could be run. Ultimately, if the Index captures multiple

measures of software utility and quality, it should be possible to offer certification levels for software to signal compliance with various best practices. Such recognition, already being examined[3], could both help users wishing to find software and encourage software developers to follow those best practices."

This begins to sound like something of a vetting process, which, while certainly useful and likely desirable, may fall outside the initial scope of this project. That sort of endeavor would need to be preceded by the curation of benchmark data sets, in addition to those data sets being viable for use by all software intending to use them for benchmarking. This would also suggest if not require that there be some standard or set of standards (akin to the standards put forth by the IEEE or IETF) against which to measure the performance of the software tools. That then asks for an unbiased method for measuring performance against standards and then comparing the performance among the tools based on comparable metrics.

All of these are, again, desirable and useful, but likely something best left to a later phase of the project.

As to reproducibility and maintenance of legacy code, might something like Docker be useful? Perhaps as a solution to Carl Boettiger's comments regarding CrossRef above. If developers were required to at least store an archive version of the code, which was proven to work in a virtual, but archived environment, this could aid in efforts to keep software accessible even after it ceases to be maintained. Though the virtual envs would need maintenance…

↳ **Reply**

Peter Murray

October 14, 2014 at 8:26 pm

This community may also be interested in a similar registry of software geared towards libraries and others in the cultural heritage community. Called foss4lib.org, it was developed using concepts from the NITRC system. One key difference is not hosting the projects themselves in the repository; rather just collect metadata about the projects and provide pointers to them. We are in the process of implementing a feature comparison utility (exemplified at ils.foss4lib.org) to aid in selecting between packages that have similar use cases.

FOSS4Lib is based on Drupal7, and we would be happy to share the code. Feel free to contact me at Peter.Murray@lyrasis.org.

Peter Murray recently posted…Umlaut – 4.0

↳ Reply

ian mulvany

October 15, 2014 at 10:04 am

When it comes to citing software with the given UID, how do we find and verify that UID. As a publisher having to hit multiple endpoints represents a point of friction in our workflows. The proposed API should take into account a facility for people querying it to be able to send in a variety of identifiers to the API.

Scoring of software – one thing that I personally use to check this is installed base, and freshness of the code. Repositories such as Drupal.org and github have managed this question quite well, and their metrics might be ones to adopt for this service. For example the rolling number of bug reports is a very

interesting aspect of information available about drupal modules on drupal.org

I don't think that it is impractical to measure aspects such as test coverage. CI systems like Jenkins and Travis CI do this quite simply. It might be a challenge to gather this information on all software, but for some software it should be trivial, and software that provides this information will probably be more valuable software, in the long run.

On the topic of delivering this index via a website and an API, one can think of the website as being the API into the google search index, so SEO should be a significant concern of the development of this portal.

In terms of defining what software goes in, some publications already look at software contributions (such as the Journal of Reproducible Software), and locations such as these might be a good place to find seed content for this index.

A possibly useful addition to MIAS might be the language that the software is developed in.

↳ Reply

Mihai Pop
October 15, 2014 at 4:35 pm

I would like to elaborate a bit more on the point about the need to capture "what the software does" in a software index. This is a non-trivial proposition that warrants a much deeper discussion in our community. One could easily equate Bowtie and Mummer, for example, as both align sequences against a genome. The specific details of what exactly the different tools do, however, are

critical to deciding which tool one needs to use. In some cases even subtle differences such as between Bowtie and BWA, or between different versions or revisions of the same tool might have a significant impact in terms of their use. We currently do not have a good mechanism for formalizing, in a concise and broadly understandable way , what a piece of software does, yet such a description is a critical prerequisite to any automated discovery system.

↪ Reply

Pete White
October 15, 2014 at 5:45 pm

In reading this report, I started to wonder about the underlying goal. After some thought, I came to the conclusion that “funding agencies struggle to make informed funding decisions about which software projects to support” is likely NIH’s driving interest. More specifically, what may be of greatest interest is how NIH can maximize ROI for their investments, both in terms of positively impacting the scientific community and being able to demonstrate better outcomes.

If this assumption holds true, then the effort described in the report seems necessary but not nearly sufficient as a value proposition. What I think may have much greater impact is a discussion about how the development community, funding agencies, publishers, and end users can maximize tool adoption. Major challenges I see revolve around usability, promotion of a useful development community, and incentive to participate as a user or developer. While it is important to be able to find software, such software is only successful if it provides sufficient user incentive and trust for a critical mass of users to overcome adoption barriers. These barriers clearly include user subjectivity and

lack of knowledge, which an indexing project can certainly help with. However, in my experience, additional barriers such as lack of bandwidth or reluctance to significantly change a current environment (uncertainty) are often more difficult to overcome. As tools seek larger and less informed audiences (and thus typically have greater potential impact), these barriers tend to grow exponentially.

My recommendation is to consider this project as part of a larger effort that incorporates principles of implementation science, community building, and awareness. There are many examples of biomedical software that have had enormous impact on our collective research. Most of these efforts have been successful because the developers have acted tirelessly as change agents far beyond the initial release of the software. There are many more examples of well-meaning software that are not widely adopted due to a lack of effort or incentive in creating sustainability and a lasting impact. In my opinion, an NIH-led discussion of how to recommend and begin to promote best practices regarding effective software utilization would be most informative, and that it could have great yield if pursued vigorously.

↳ Reply

Mohammed Eslami
October 15, 2014 at 6:08 pm

I'd like to take a step back as it seems like many of the comments are focused on finding the appropriate unique identifiers (or indices) to label/find software. It makes it sound like the software is becoming the central piece of the indexing scheme, when clearly this is not the purpose of indexing. Software indexing, or even indexing in general for that matter, is merely a tool for the user to be able to

identify the most relevant item of interest when queried in a very timely manner. It is therefore that I believe this standard needs to take a much more human centered approach rather than a software centered one. What makes NIH's problem of data driven or software driven indexing very interesting is that it is a domain specific index/search problem. This limits the items to be indexed to a specific domain thereby allowing for an adaptable task-specific index that can be catered to individual (or groups of) users.

DARPA has very recently begun to focus one program's efforts in the domain of human trafficking on this very subject (see link at the end of this post on Memex). Through efforts as such these, it is clear that a "one-size-fits-all" solution for indexing is not the answer and that we are moving towards having a decentralized indexing scheme tailored to a specific knowledge group. If the NIH standard could expand from an indexing/cataloging of datasets or software to a system that brings the user at the center of the effort, it may prove to position itself much better in an already evolving domain. For example, a researcher focused on neuro-oncology would be more interested in seeing information focusing in that specialized area rather than oncology in general. The indexing scheme for that user should then be adapted for these sorts of individuals based on past searches, page visits, clicks, duration of stay, and other features. In this way, the system is not solely designed to find the specific response that best fits the search query through a common indexing scheme, but it will be a tool that aids the user in quickly retrieving the response through many related responses that may not be of interest to the individual.

In conclusion, an initial indexing scheme is necessary, but to meet the challenges of an evolving area, the pillars of an adaptive one need to be addressed.

Memex: https://www.fbo.gov/index?s=opportunity&mode=form&id=426485bc9531aaccba1b01ea6d4316ee&tab=core&_cview=0

↳ Reply

Pingback: Software DIscovery Index and Metadata for Software Minimal information about software (MIAS) | Laurie N. Taylor

Bruno Vieira
October 19, 2014 at 12:30 pm

More examples of registries for bioinformatics:

http://registry.biojs.net/client/#/
http://sciencetoolbox.org
https://www.npmjs.org/browse/keyword/bioinformatics
http://biogems.info

↳ Reply

James Howison
October 22, 2014 at 11:41 am

Great discussion. I wanted to share a pieces of research that might be useful to putting some numbers on some of the claims discussed above (such as “it’s easy to find the software” and “everyone publishes papers that we can cite, we should just cite those”).

We studied of how software is currently mentioned in the biology literature,

together with an examination of the ability to find and re-use that software. We took a random sample of 90 biology journal articles, stratified by impact factor, developed a reliable content analysis scheme and analyzed the articles.

Howison, J., and Bullard, J. (Under review). The visibility of software in the scientific literature: how do scientists mention software and how effective are those mentions? Working paper submitted to Journal of the Association for Information Science and Technology.
http://dx.doi.org/10.6084/m9.figshare.1146366 Data and analysis at:
https://github.com/jameshowison/softcite

Headline findings: only 37% of mentions are formal citations, 78% of mentions attempt some form of crediting (higher than I'd imagined). Most of the code is identifiable, given the information in the papers, about 94% overall. Overall about 87% of the software could be found, at least to package level. Versions fare much, much, worse: only 20% of mentions provide any version data (including dates) and we could only find the specific version mentioned in 1 of 4 of those cases (making only 5% of mentions able to be found at a version level). Back at the package level, there is a lot of code that is inaccessible: 20% is inaccessible in any form, only 37% has source code access, and only 20% has explicit permission to modify (usually an explicit open source license). Things don't vary much even in the higher impact factor journals, in fact some are worse there (more mentions of unidentified software).

We also looked at the journals in the sample, seeing if they had a policy for citing software. Only 24% did overall, with the higher impact factor journals being more likely to have one. Request for citation are relatively rare and not that effective: We found that only 18% of packages made a specific request to be cited in a particular way, but only 68% of the mentions of those pieces of

software actually used the preferred citation (32% did not, including one where citing was a license requirement of using the software).

So I think these results show the scale of the problem; of course knowing what exactly to do is much harder 🙂 Identifiers for software are important, as is being able to track it in publications (so the goals of this report make great sense). Getting publishers (and funders!) on-board to ensure that software mentions are done properly is crucial.

We address some of this in another recent study taking a 360° view of reasons and techniques to measure software in science, and a set of long-term goals and policy possibilities:

Howison, J., Deelman, E., da Silva, R. F., McLennan, M., & Herbsleb, J. D. (under review). Metrics for insight into scientific software ecosystems (Working paper for submission to Research Evaluation). http://james.howison.name/pubs/SoftwareMetricsPaper-WorkingPaper.pdf.

↳ Reply

Susanna-Assunta Sansone
October 23, 2014 at 12:51 pm

Great discussions and also thanks to the NIH BD2K team for such useful workshops around key topics!

Echoing other comments and linking to other discussions, such those from last year's "NIH BD2K – Frameworks for Community-Based Standards Efforts" event and ELIXIR activities: we need to ensure that this software discovery index is

designed as part of the growing ***ecosystem of registries***, holding metadata descriptions of others research outputs (such as content standards, databases, methods, datasets, publications, people etc). To achieve this we also need to harmonize the ***MIAS metadata descriptions*** for the software with more***generic metadata descriptions and terminologies*** for all these registries in order to cross link them, e.g. according to data types, areas of study, type of licenses etc.

For example, linking software to data/metadata standards – hosted by BioSharing – is very useful, but ultimately cross linking **standards**, **databases**, **datasets** and **tools** is the ultimate goal, along with linking these to additional resources, such as **training material**,**scholarly profiling** and tools to create data management **plans** etc.

Researchers, *developers* and *curators* lack support and guidance on how to best navigate and select these appropriate tools, standards, etc., understand their maturity, or find databases that implement them;
But also *funders*, *journals* and *librarians* do not have enough information to make informed decisions on which tool, content standards or database to recommended in policies, or funded or implemented.

In conclusion, designing this software discovery index as part of the growing ecosystem of interconnected discovery indexes is pivotal to deliver a system that is really meeting the users' needs.

↳ **Reply**

Desfeux Arnaud
October 25, 2014 at 12:42 pm

Dear all,

This is a great initiative and we totally share your goals.

We have developed OMICtools (omictools.com) a freely available metadatabase that provide more than 6000 software tools and databases to the scientific community. OMICtools is a workflow for genomic, transcriptomic, proteomic, and metabolomic data analysis. All tools have been classified by omic technologies (NGS, microarray, PCR, MS, NMR).

An interface has been established to allow anyone to report a problem, write a comment or rate a given website, in an attempt to develop an interactive community. 20 volunteer curators are involved with us in this project.

Reproducibility is a priority for us. In collaboration with the Neuroscience Information Framework (NIF), OMICtools has already added a unique Research Resource Identifiers (RRIDs) for each software tool and database.

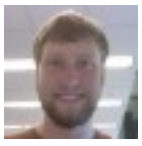
We would be happy to help the Software Discovery Index.

Best

Arnaud

OMICtools coordinator

↳ Reply



Spencer Bliven

October 28, 2014 at 6:49 am

This is a great initiative, one which will really help the bioinformatics community (and hopefully extend to other fields too). A few scattered comments:

– Several of the features listed would require some method for invoking the software (benchmarking, reproducibility). Perhaps a solution to this would be to include (optional) metadata for incorporating the software into some workflow manager.

– I support the incorporation of existing standards, rather than the creation of new standards

– I like the idea of multiple parallel Indices, with the NIH hosting the main one but other fields or entities hosting compatible indices with different software. This parallels the linux software repository system, where most software comes from the official distribution but additional packages can be distributed separately where needed.

– The metadata should be fairly easy to specify, and have an easy editor for developers.

↳ **Reply**

Charlie Ma
October 30, 2014 at 11:39 am

I read with great interest the Software Discovery Index Meeting Report. As a librarian who works with front-line medical researchers, I am glad to see people in this community have the vision to create an index for public-funded biomedical software. From the framework of the Software Discovery Index

described in the report, the benefits of Software Discovery Index are very clear. I can see researchers will find related software tools easily and software can find its way to reach target users efficiently.

I would like give some inputs form a librarian perspective.
– The needs to connect the Software Discovery Index to existing biomedical literature database (e.g. PubMed). If the Software Discovery Index is a stand-alone product, it will limit its power to disseminate the software information. As the report pointed out the Software Discovery Index should work with other repositories, I think the Software Discovery Index must work with biomedical literature database developers to create a field (link) to present software information for the research literature in the database search result.
– The data field for the Software Discovery Index in biomedical literature database could be similar to the DOI. This will allow user easily access the detailed information in the Index.
– Ideally there will be a link resolver type of tool to link the “software citation” to the software, making the software easy to access.
– To establish a mechanism collecting feedback from users and keeping the software information up-to-date.
– A detailed guideline to define the scope of public-funded biomedical software inclusion.
– A relevant ranking system to help researcher evaluating the software. The Software Discovery Index will not only answer the question – “What software can researcher use?”, but also provide information on software ranking to help researcher choosing the software.

↳ Reply

John Greene
October 30, 2014 at 2:52 pm

This just in to stir the pot more…. <a href="http://www.genomeweb.com/blog/share-code-0"

↳ Reply

John Greene
October 30, 2014 at 2:54 pm

Share the Code

↳ Reply

Pingback: Open Science & Altmetrics Monthly Roundup (October 2014) - Impactstory blog

James Miller
October 31, 2014 at 9:55 am

It is probably about time that software have resource identifiers, particularly if the identifier links to information about the specific version (and perhaps platform) that was used. This seems like the “right thing to do”.

However, I am not sure that having resource identifiers for software actually addresses he need for reproducibility. Citing a particular piece of software does not capture how that software was utilized in producing the claimed results. This is certainly true when citing an end-user application, where sequences of mouse clicks and operations on data are not captured. But it is also true when the

software is the library or toolkit, and citing the software does not capture which classes and algorithms were utilized nor the order of their application and parameter settings.

While resource identifiers for software is a good idea, it does not completely address one of the key needs.

James Miller recently posted…Staying on Track for a Career in R&D

↳ Reply

Carole Goble
October 31, 2014 at 3:18 pm

Thanks for the initiative and the great comments. I wholeheartedly back the aims, and the comments above have really been insightful and informative.

Packaging, portability and execution
————————————————-

I entirely endorse the comments by Paul Groth and a few others. If we are serious about reproducibility, dependencies, versions etc then you have to address packaging (packaging with data is a further step).

Open platforms for distributed applications like DOCKER (http://www.docker.com) are now available, usable and widely adopted.

Research Object idea – see http://www.researchobject.org – addresses the principle of:
(a) treating research outputs like software as first class citizens to be managed, credited and tracked in a common way and

(b) bundling and relating multi-hosted (digital) resources of a investigations using standard mechanisms & uniform access protocols,

The RO model systematically addresses the metadata needed for dependencies, checklists (aka MIAS), versioning and provenance, using standard metadata models from the digital library community for aggregation (OAI-ORE) and annotation (W3C OAM), using progressive annotation profiles.

See http://www.slideshare.net/carolegoble/goble-keynote-vivoscits2014 for a recent talk on this.
We have been working with Chuck Vardeman Notre Dame on describing a DOCKER manifest using the RO model, for example.

It would be great if these ideas could be aligned/incorporated into the Software Discovery Index and its linkage to the Data Discovery Index.

Other registries

———————

To add to the pool above:
The EU ELIXIR program has an initiative to build and manage a tools and data services registry for Life Sciences. Again it would be good to harmonise efforts.
We are involved through our management of the BioCatalogue (http://www.biocatalogue.org) which is not a catalogue of software tools but of web services.

There are a couple of ontologies that would be useful:
EDAM: http://edamontology.org/page
SWO: http://softwareontology.wordpress.com/,
http://www.jbiomedsem.com/content/5/1/25/abstract

Software Citation

———————————

Some thought about software citation should also accompany the work: notably the difference between citing software as proof (this is what I used, where version fixivity is critical) and citing software as signpost (this is where to get it and get the most recent version for your use). The Software Discovery Index, if used as a prime journal citation platform, needs to handle both.

Digital Libraries & Journals

———————————————

PLEASE do not just look to the Life Science and Biomedical field for expertise. The Digital Library community know quite a lot about indexing and registries, and had quite a few standard protocols for metadata harvesting (OAI-PMH), search etc. The same for journals. In that vein you might consider extending NISO/JATS http://jats.niso.org/ to inform the MIAS from the journal citation perspective.

↳ Reply

Brent Shambaugh

November 13, 2014 at 8:02 pm

I like your unique identifier idea.

” In this effort, controlled vocabularies and ontologies may prove to be useful, but should not be the primary focus of the initial effort.” –

While it is in my brain, here is a link to the Description of a Project Ontology: http://lov.okfn.org/dataset/lov/details/vocabulary_doap.html

↳ Reply

Sweitze Roffel
November 14, 2014 at 6:13 am

Dear Dr Owen White at al.

This is a very interesting initiative and I could not agree more with your aims. I feel that software discovery is important beyond the domain of biomedical research, and vital across all domains of science. How can one check the results of any science without being able to access and understand how these results were computed?

WRT the discussions here it might be interesting to share that at Elsevier we've been quietly working on treating 'software' as an equal academic citizen for quite some time. Being an 'equal citizen' naturally means proper indexing of these important research artefacts, a nontrivial issue for "born digital' research output.

Our first roll out in this space is support for a novel class of scientific publication called Original Software Publications (OSP). These OSP's describe major/significant software and code in full, including post publication updates (versions) to capture and organize all metadata needed to expose this 'body of work' to interested readers and users.

The Original Software Publications and their subsequent updates (versions) will be peer reviewed and considered "one body of work" for citation and indexing purposes. One could say it's main purpose is to be the canonical and verified academic reference point for "software"

See more at: http://www.elsevier.com/about/content-innovation/original-software-publications

To achieve these goals we modified current (human, organizational and IT) infrastructures & practices to create the groundworks for further innovation (start with what we have as opposed to what we'd like to have). To enable further scaling and enrichment this new class of content has been purposely designed to be business model neutral on the software & publishing side, as well as being systems agnostic.

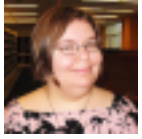
As no single organization controls all the required infrastructure, this project is also an interesting exercise in better organizing content across a distributed ecosystem. Or in short: we need (and like) to collaborate with relevant stakeholders to better expose "software as a body of academic work'.

its unlikely any plan survives direct contact with reality unchanged. To test and learn before attempting to scale across 'all sciences' we are rolling out versions of this new OSP content class in a number of selected publications.

The very first journal testing this OSP in the real world can be found here: http://www.journals.elsevier.com/neurocomputing/call-for-software/a-new-software-track-on-original-software-publications/

I'd be very interested in your general comments wrt this project, and also interested in any practical comments you may have after you've tried submitting an OSP to any of the venues supporting this new class of content.

↳ Reply



Jodi Schneider

November 25, 2014 at 2:17 pm

This is a substantive report. I have some minor comments, selected from the scribbles on my printed copy which I share (belatedly) in case that's useful. I strongly agree that supporting reproducibility is a critical need and that identifiers and software indices are a key part of improving the situation. Particularly that "one of the major contributions of the Index may be in improving the reproducibility of published analyses."

I also strongly agree that there is a niche for a software repository, analogous to PubMed: consistency and selection add to the utility here. "automated indexing of software, the integration with multiple registries, and the provision of APIs enabling the creation of community-specific user interfaces."

——

Criticism/suggestions

It's not immediately clear who mints RRID's — that could be clarified in the paragraph about that.

"Properly citing" is value-laden. That may be want you want, but if this is intended for those same authors, a different phrasing might be called for.

"Pilot projects have shown that both journals and authors recognize the need for such a system…" — any examples to include?

Consider referring to the "Software Discovery Index" as the "planned Software Discovery Index". Note that "data" is vague in this context, it's worth specifying further when using this phrase in the "use cases" section.

“Ability to install the code” — by whom?
“understand the impact of changing run-time variables”–again, by whom

“Moreover, other resources such as Synapse, GitHub, Zenodo, figshare, SciCrunch/NIF, and others may wish to expose their software to this index, and the API should enable this as well.” — this “wish” would have to come from upstream demand (users/funders/etc). So I agree in principle but I think that for success, the incentives need to be there. Better when you say “This will require strong relationships with multiple existing repositories and a willingness to work with new repositories that contain relevant software.”

Citations/links to other efforts should be given (BioSiteMaps, NITRC, NIF) as these are unlikely to be known outside of the biosciences; this will make your report more valuable to cognate communities (e.g. library & information sciences; digital preservation; digital libraries; escience) exploring the state of the art.

“Engage effectively with the Data Discovery Index” — this is underspecified. Can you be more specific about how to measure this or achieve it?

—-
“Publisher: A publisher can associate software with their publications during & for peer review and upon publication for citation. They can also pull & display metrics related to all the research objects surrounding the article, including software based on the software identifier.” — Please link to your definition of research object. Is there any evidence that this is happening or feasible?

For metrics: links/annotation — is this presence/absence or the number? How do you propose to measure the use of the APIs to repackage the data for specific

use cases?

“supports various package management functions” — not clear what you mean here.

“error rate to date is ~7%” — any way to contextualize this by comparing to other similar initiatives?

↳ Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

NAME *

EMAIL *

WEBSITE

COMMENT

You may use these HTML tags and attributes: <a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code> <del datetime=""> <i> <q cite=""> <strike>

POST COMMENT

commentluv