




CiteLang: Modeling the Research Software Ecosystem

Vanessa Sochat ¹

¹ Lawrence Livermore National Laboratory, Livermore, CA, USA

DOI: [10.21105/joss.04458](https://doi.org/10.21105/joss.04458)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Fabian-Robert Stöter](#) 

Reviewers:

- [@gflofst](#)
- [@rmmilewi](#)

Submitted: 26 April 2022

Published: 05 September 2022

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

Understanding attribution of software is essential for understanding the research software ecosystem and evaluating the utility or value of any particular library. While substantial work has been done to discuss research software citation ([Smith et al., 2016](#)), there is not follow up work to provide libraries or methods to model this ecosystem for easier study, despite the need ([Glass et al., 2002](#); [Goble, 2014](#)). CiteLang is the first tool to afford this type of study, offering automated analysis and data extraction for open source software repositories, generation of summary analysis and graphs for single packages or groups of software, and maintaining a local database to store cached data. Using CiteLang it is possible to calculate different views of open source contributions for a repository ([Sochat, 2022b](#)), or to do analyses that describe an entire ecosystem of software ([Sochat, 2022c](#)). This tool should be of interest to research groups interested in studying research software engineering ([Rosado de Souza et al., 2019](#)) or the software ecosystem ([Cohen et al., 2021](#); [Glass et al., 2002](#)).

Statement of Need

Research software engineering ([Baxter et al., 2012](#)) is becoming a more established profession, providing best practices for software in research ([Cohen et al., 2021](#)) and fueling an entire economy of new jobs for research software engineers (RSEng). As this role has grown out of academia, the accepted practice of publication to derive value of an individual has followed tradition from academia, meaning that RSEng have not only the burden to write software, but to publish papers to prove their value. While writing a software paper may be appropriate for cases that warrant getting the attention of an academic community, writing papers should not be the primary way that RSEng are valued. While discussion of research software citation is not uncommon ([Smith et al., 2016](#)), what is uncommon is derivation of libraries and modeling software that can make it easy to study the ecosystem and propose new paradigms. Work in this area tends to focus on generating single citations and metadata for a specific project ([Druskat, 2019](#); [Jones et al., 2017](#)), to require manual derivation of details and discussion ([Katz & Smith, 2015](#)), or to make the assumption that the end goal or best idea is to still fit research software into the traditional academic citation system ([Group, 2022](#); [Katz et al., 2019](#)). Undeniably, efforts to capture more metadata around a software project are important, but they typically require extra work on the part of the researcher, and do not address the larger question of rethinking valuation of research software engineers. Notably, most of these papers that discuss software citation provide theoretical examples. There is a gap in work to provide software to better model the ecosystem, and a community initiative to think about ideas that go beyond a traditional citation.

Software for Modeling the Software Ecosystem

CiteLang grew from this need, and is the first library of its kind that aims to empower researchers to better study the ecosystem by way of providing methods and graph-based modeling of a single project or larger ecosystem. A research group using CiteLang can easily derive:

- How software depends on other software
- How software dependencies change over time
- How the value of a particular software library changes

And can customize an analysis to choose everything from how to distribute credit between software and dependencies, to setting a minimum credit value to stop parsing, or even to stop parsing after a particular level of dependency is traversed. CiteLang works by way of taking advantage of package manager APIs (when available) to automatically discover software metadata and dependencies, and generates a cache for saving all data and for re-use. A list of supported package managers is provided in Table 1.

Name	Project Count	Homepage	Default Language
Alcatraz	464	http://alcatraz.io	Objective-C
Bower	69517	http://bower.io	CSS
Cargo	81429	https://crates.io	Rust
Carthage	4515	https://github.com/Carthage/Carthage	Swift
Clojars	24291	https://clojars.org	Clojure
CocoaPods	87311	http://cocoapods.org/	Objective-C
Conda	16297	https://anaconda.org	
Cpan	39086	https://metacpan.org	Perl
Cran	22051	https://cran.r-project.org/	R
Dub	2404	http://code.dlang.org	D
Elm	2605	http://package.elm-lang.org/	Elm
Github		https://github.com	
Go	365289	https://pkg.go.dev	
Hackage	16460	http://hackage.haskell.org	
Haxelib	1703	https://lib.haxe.org	Haxe
Hex	12946	https://hex.pm	Elixir
Homebrew	7503	http://brew.sh/	C
Includer	228	https://include.org/	C++
Julia	3048	http://pkg.julialang.org/	Julia
Maven	469374	http://maven.org	Java
Meteor	13410	https://atmospherejs.com	JavaScript
Nimble	1902	https://github.com/nim-lang/nimble	Nim
Npm	2324490	https://www.npmjs.com	JavaScript
Nuget	375186	https://www.nuget.org	C#
Packagist	355691	https://packagist.org	PHP
Pub	30141	https://pub.dartlang.org	Dart
Puppet	6923	https://forge.puppet.com	Puppet
Purescript	582	https://github.com/purescript/psc-...	PureScript
Pypi	437955	https://pypi.org/	Python
Racket	2193	http://pkgs.racket-lang.org/	
Rubygems	178224	https://rubygems.org	Ruby
Spack	6375	https://spack.github.io/packages	
SwiftPM	4207	https://developer.apple.com/swift/	Swift

Table 1 Package managers supported by CiteLang. A subset use their own APIs, while others use libraries.io. The “GitHub” package manager looks at dependencies parsed from the [GitHub dependency graph](https://github.com).

Along with methods to derive data to model the ecosystem, CiteLang provides a suite of command line tools to generate graphs that can be displayed in the terminal, and other formats for popular graphing software (dot, Cypher for Neo4j, and Gexf for NetworkX). By default, CiteLang will use a model that attributes 50% of credit to a main package, and then 50%

evenly distributed to all dependencies, and applied recursively up to a minimum credit limit (e.g., 0.001) or to a specific level of parsing (e.g., three generations of dependencies). CiteLang also provides a badge command for a repository to generate a sunburst badge (Figure 1).

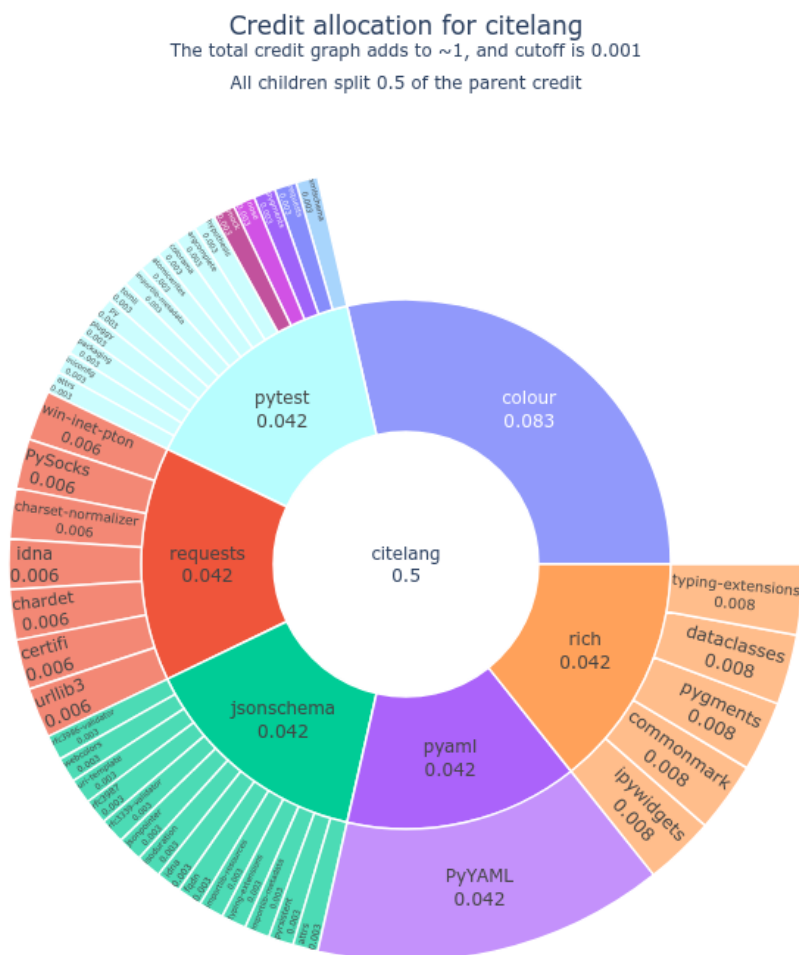


Figure 1: Figure 1: A repository badge generated with CiteLang.

All of these actions can be accomplished from within Python, from the terminal with the CiteLang client, or via an automated workflow and the GitHub actions (Sochat, 2022a) that CiteLang provides. In summary, the software enables research to better understand our research software ecosystem, and hopefully implement ideas to improve it.

Community Initiative

A large part of innovation is cultural. On a cultural level, CiteLang is the first tool of its kind to suggest an alternative way to value software beyond the traditional academic practice of publication. It suggests a model where RSEng are not required to generate separate DOIs (digital object identifiers) associated with papers, or supplementary metadata files about the

software. This model uses already established ways to publish and distribute software, package managers.

CiteLang is part of a larger vision that is needed for an initiative of tooling to empower research groups to study software and imagine new paradims to supplement or replace citation. Beyond citation, the software offers a set of tools to enable any kind of graph- or metadata- based study of a software ecosystem. If researchers do not have analysis tools that make it easy to model and understand the space, change will come slowly, if at all. If research software engineers do not proactively develop and champion tools for this kind of study, it will be less likely to happen, and we will choose decision through indecision – valuing research software engineers based on a broken publication system – only because it's the way we have always done things, and nobody has been inspired or empowered to try anything different.

Conclusion

CiteLang makes it possible to better study the ecosystem of research software. It supports study, visualization, and data extraction to enable more applied research around software citation. You can read more about CiteLang at the GitHub repository (<https://github.com/vsoch/citelang>) or the [documentation user guide](#).

References

- Baxter, R., Hong, N. C., Gorissen, D., Hetherington, J., & Todorov, I. (2012). The research software engineer. *Digital Research Conference, Oxford*, 1–3.
- Cohen, J., Katz, D. S., Barker, M., Chue Hong, N., Haines, R., & Jay, C. (2021). The four pillars of research software engineering. *IEEE Software*, 38(1), 97–105. <https://doi.org/10.1109/MS.2020.2973362>
- Druskat, S. (2019). *The citation file format (CFF): Why, what, how?*
- Glass, R. L., Vessey, I., & Ramesh, V. (2002). Research in software engineering: An analysis of the literature. *Information and Software Technology*, 44(8), 491–506. [https://doi.org/10.1016/S0950-5849\(02\)00049-6](https://doi.org/10.1016/S0950-5849(02)00049-6)
- Goble, C. (2014). Better software, better research. *IEEE Internet Computing*, 18(5), 4–8. <https://doi.org/10.1109/MIC.2014.88>
- Group, F. (2022). *FORCE11 software citation implementation working group*. <https://github.com/force11/force11-sciwg>
- Jones, M. B., Boettiger, C., Mayes, A. C., Slaughter, P., Gil, Y., Chue Hong, N., & Goble, C. (2017). *CodeMeta*.
- Katz, D. S., Bouquin, D., Hong, N. P. C., Hausman, J., Jones, C., Chivvis, D., Clark, T., Crosas, M., Druskat, S., Fenner, M., Gillespie, T., Gonzalez-Beltran, A., Gruenpeter, M., Habermann, T., Haines, R., Harrison, M., Henneken, E., Hwang, L., Jones, M. B., ... Zhang, Q. (2019). *Software citation implementation challenges*. arXiv. <https://doi.org/10.48550/ARXIV.1905.08674>
- Katz, D. S., & Smith, A. M. (2015). Transitive credit and JSON-LD. *Journal of Open Research Software*, 3(1), 7. <https://doi.org/10.5334/jors.by>
- Rosado de Souza, M., Haines, R., Vigo, M., & Jay, C. (2019). What makes research software sustainable? An interview study with research software engineers. *2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, 135–138. <https://doi.org/10.1109/CHASE.2019.00039>
- Smith, A. M., Katz, D. S., Niemeyer, K. E., & Group, F. S. C. W. (2016). Software citation principles. *PeerJ Computer Science*, 2, e86. <https://doi.org/10.7717/peerj-cs.86>

- Sochat, V. (2022a). *CiteLang GitHub actions*. https://vsoch.github.io/citelang/getting_started/user-guide.html#github-action
- Sochat, V. (2022b). *Open source contributions with CiteLang*. <https://vsoch.github.io/2022/citelang-contrib/>
- Sochat, V. (2022c). *The research software ecosystem*. <https://vsoch.github.io/2022/rsepedia/>