The Scope and Impact of Open Source Software as Intangible Capital: A Framework for Measurement with an Application Based on the use of R Packages

March 11, 2019[1]

Carol A. Robbins*(1), Gizem Korkmaz (2), José Bayoán Santiago Calderón (3),
Daniel Chen (2), Aaron Schroeder (2), Claire Kelling (4), Stephanie Shipp (2), Sallie Keller (2)

Abstract

Open source software allows free access to digital tools and constitutes a part of intangible investment with the qualities of public goods. Open source software (OSS) provides users with an unknown amount of freely modifiable software tools and other useful products; they are created both within the business sector and outside of it. Better accounting for the contribution of public spending to investments in OSS, a vital component of science activity, motivates this paper. We develop a bottom-up approach to document the scope and impact of OSS created by all sectors of the economy by collecting data on OSS languages R and Python, as well as from the Federal Government's Code.gov.

Using lines of code and a standard model to estimate package developer time, we convert lines of code to resource cost. We estimate that the resource cost for developing R and Python packages exceeds three billion dollars, based on 2017 costs. We find that the base software for R was downloaded over 1 million times in 2018, with more than 400 thousand of those from the United States. Add-on components, or "packages" were downloaded over 600 million separate times worldwide in 2018, and over 256 million times in the US. We analyze downloads and reuses between software packages as measures of relative impact. We find that by either measure ggplot, a graphics package, and Rccp, a package with C++ tools, are among the highest impact packages. This methodology provides the first step to developing a set of prototype statistics for the contribution of public investment in open source software; the bottom-up approach makes it possible to begin identifying the sector of the contributors based on publicly available data. This can provide an additional indicator of the outcomes of research dollars, currently characterized primarily by patents and bibliometric indicators.

Key words: Open Source Software, Intangibles Measurement, National Accounts

1) National Center for Science and Engineering Statistics, National Science Foundation; 2) Social & Decision Analytics Division, Biocomplexity Institute & Initiative, University of Virginia; 3) Claremont Graduate University; 4) Pennsylvania State University

[1] Earlier versions of this paper were presented August 21, 2018 at the International Association for Research on Income and Wealth. [http://www.iariw.org/copenhagen/robbins.pdf] and November 19, 2019 at the 6th IMF Statistical Forum: Measuring Economic Welfare in the Digital Age: What and How? (https://www.imf.org/en/News/Seminars/Conferences/2018/04/06/6th-statistics-forum)

This work was supported by U.S. Department of Agriculture (58-3AEU-7-0074).
The views expressed in this paper are those of the authors and not necessarily those of their respective institutions. CR and GK designed and performed the analysis and took the lead in writing of the paper. GK, JC, DC, AS, and CK collected the data and contributed analysis tools. All authors contributed to the writing of the paper. The authors also thank the 2018 Data Science for the Public Good program students Keren Chen, Hannah Brinkley, and Eirik Iversen at Virginia Tech for their contributions.
*Corresponding author: crobbins@nsf.gov

Introduction and Contribution

Open source software (OSS) is everywhere, both as specialized applications nurtured by devoted user communities and as digital infrastructure underlying platforms used by millions daily. It is computer software shared with a license in which the copyright holder provides the rights to study, change, and distribute the software to anyone and for any purpose. OSS is developed, maintained, and extended both within the private sector and outside of it, through the contribution of independent developers as well as people from businesses, universities, government research institutions, and nonprofits.

Many OSS projects create long-lived tools that are often outputs of public spending, a kind of freely share-able intangible asset that in many cases have been developed outside the business sector and subsequently used within the business sector. The scale and use of these modifiable software tools highlight an aspect of technology diffusion and flow that is not captured in market measures. Measures of creation and use of OSS would complement existing science and technology indicators on peer-reviewed publications and patents that are calculated from databases covering scientific articles and patent documents. Many well-developed methodologies and extensions exist, and a research community continues to grow, invigorated by improved computing power and algorithms. We are motivated to better account for both the scale of OSS overall and the contribution of public spending to investments in open source software, a vital component of science activity.

In this paper, we explore non-survey data to see how they may be used to measure the scope and impact of OSS, focusing on the open source languages R and Python as prototypes. Python is among the most common OSS languages and R has features for statistical analysis and visualization that make it a no-cost substitute for proprietary software like SAS.

Our data collection strategy combines data from multiple sources. We collect OSS data that is disseminated online from archives, repositories and from the projects themselves. We start with the location with the broadest coverage of the target languages, registries and package managers, that host and distribute OSS, such as the Comprehensive R Archive Network (CRAN) and the Python Package Index (PyPI). These online sources provide individual package files containing metadata such as author, license, version, and dependencies (required packages to function). We collect variables such as the number of complete R and Python packages available, downloads per package, and reuses in other packages. The information includes source code hosting platforms, such as GitHub, where these projects are developed and maintained. We identify R and Python packages that are hosted on GitHub and collect data also on their development activity (e.g., lines of code, contributors, code changes over time).

We have a two-pronged approach to measuring the scope and impact of OSS. First: we estimate the resource cost associated with creating these packages and projects to indicate the scale of the project. Following Boehm (1981, 2000), we use lines of code in a package as the measure of effort to estimate the time spent on software development. Average compensation for computer programmers and computer system analysts from Bureau of Labor Statistics wage data and other costs based on national accounts methodologies allow us to estimate per package resource costs for the R and Python packages on GitHub. Second: we use methods developed for bibliometrics and patent analysis to study the impact of these projects. This involves counting OSS software packages and projects as discrete units of output, analyzing the downloads, and calculating the relationships between packages that imply re-use across packages. The production and delivery of OSS through publicly accessible websites provide harvestable count and linkage data for software languages and packages.

This paper is organized as follows. First, we explain how software is measured in the national economic accounts and describe the project's motivation through the landscape of open source software and the platforms where it is shared. We then describe our approach to data collection and preparation. We describe and present both counts and order of magnitude estimate for the resource costs of OSS packages for R and Python. We rank packages by two different types of impact and finish our paper with a look at the internet domains that are associated with package maintainers.

Measurement of Software in the National Accounts.

Annual investment in software in the US as a nation is estimated at $381 billion a year in 2017, according to the BEA (2018a, 2018b).   About $38 billion of this is public sector investment (federal labs and facilities, public universities, state and local government entities). Most investment accounted for, more than $350 billion dollars, is for private sector investment in software (Figure 1).

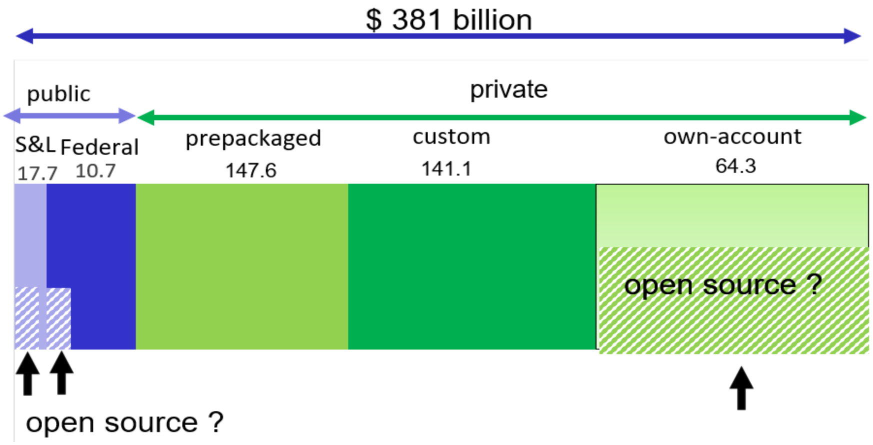


Figure 1 BEA's Software Investment Measures for the US in 2017

Private investment is published in three categories, prepackaged, custom, and own-account. OSS created with public funding would be part of the $17.7 billion in S&L investment if funded by a public university, and part of the $10.7 billion if funded by the Federal government. If funded by a private, nonprofit university or institution, it would be part of private investment. What we don't know is how much of any of this is for open source software. Macro-level statistics don't have the resolution to tell us.

As measured in the national accounts, software investment has three types, prepackaged, custom, and own-account (inhouse work). For counting investment, prepackaged and custom software are purchased inputs, product revenues should be an available data source. However, production activity within firms and organizations also bring forward new software tools. As a freely-shared software tool, OSS can be custom software or own-account. Own-account software is not purchased or sold: it is new, or significantly-enhanced software created by business enterprises or government units for their own use and its value is estimated based on in-house expenditures for its creation (Parker and Grimm, 2000).

This software investment drives a wide range of economic and value creation activity that challenges current measurement. Many digital products are used by consumers without a direct payment: similar to network television programming, their costs are supported by advertising. This kind of free content that is bundled with advertising can be understood as a barter transaction, content in exchange for being exposed to the advertising. In the absence of a direct price, this content created in the business sector can be valued based on its production cost (Nakamura, Samuels, and Soloveichik, 2016 & 2017).

Software and databases can provide revenue in an additional way. In use, online platforms collect data about users as well as transaction fees. These data are part of the value that the platform provides. Li and co-authors (2018) describe several different types of online platforms, including E-commerce, online resource sharing, e-financial services, and online social network services, where data collection provides high value to the business. In these cases, the cost and market-based approaches underestimate the value of data. Using an income approach, they argue, better captures the variety of ways that firms monetize software and data.

Beyond these categories, Corrado, Hulten, and Sichel (2005) provide a framework for consistent accounting for a larger set of intangibles that generate future benefits, including brand equity and investments in human and organizational capital. Further arguing that public expenditures yielding long-lived returns should be understood as investment, Corrado, Haskell, and Jona-Lasinio (2015) propose a public investment category: information, scientific, and cultural assets. They argue that better accounting of public investment in intangibles would provide a more complete picture of economic growth.

Open Source Software Related Definitions

Open source software (OSS): Computer software with its source code made available with a license in which the copyright holder provides the rights to study, change, and distribute the software to anyone and for any purpose. For this paper, we treat as open source any software language or package with an Open Source Initiative (OSI)-approved license.

Own account software: A category of software investment in national economic accounts. Own-account software is long-lasting software created for internal use, rather than as a market product.
Production Ready Release: A software package that is ready for production in its current ecosystem.
Registry: A location that hosts, manages, and distributes OSS. The Comprehensive R Archive Network (CRAN) and PyPI for Python are examples.

Repository: An online hosting facility for maintaining versions of software programs. Source code hosting facilities such as GitHub, SourceForge, Bitbucket, and GitLab are commonly used to develop, download, review, and publish OSS projects and computer code.

Commits: With respect to OSS development, a commit is an incorporated improvement to existing code.

The Landscape of Open Source Software (OSS)

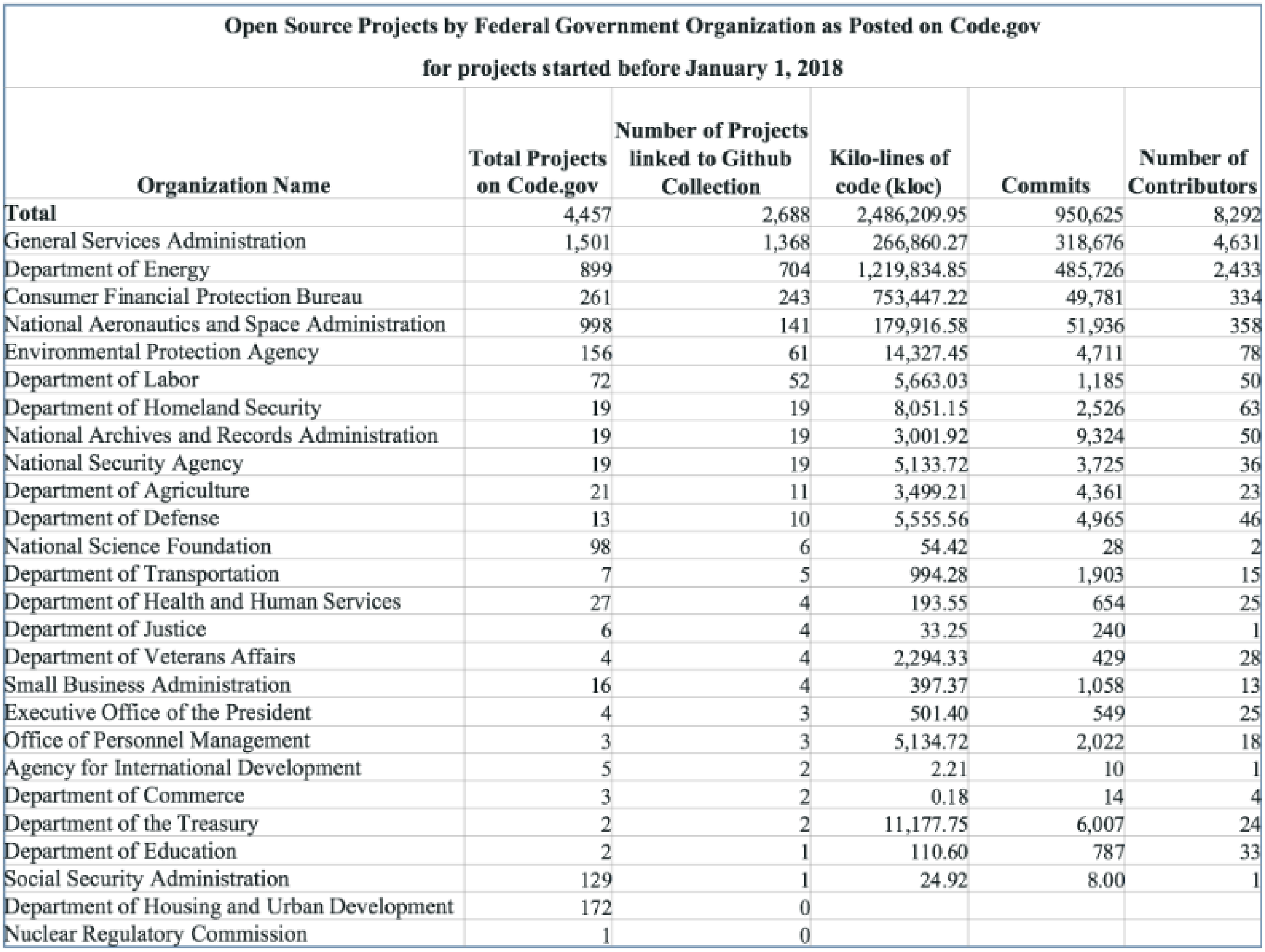
Beginning in the early 1980s, OSS projects have provided users with zero-dollar cost and freely modifiable software tools. Table 1 lists some of the widely-used OSS projects and the year of their initial release. LaTeX is typesetting software popular for its ease with mathematical symbols, introduced in 1983 by the nonprofit research organization SRI. The Linux operating system is the basis for many applications, including the most widely-used operating system globally-- the Linux-based Android operating system (GlobalStats statcounter, 2018). Apache is server software developed with federal and state funds at the National Center for Supercomputing Applications in Illinois. As of July 2018, Apache is the most frequently used HTTP server on the internet (W3Techs, 2018). Greenstein and Nagle (2014) estimated the value of capital stock of Apache software in use in 2013 at between $2 and $12 billion.

Table 1. Major OSS Projects

Name	Type	Initial Release
LaTeX	Typesetting	1983
Linux	Operating System (OS)	1991
Apache HTTP Server	Web server	1995
GIMP	Raster graphics editor	1996
VLC media player	Media player	2001
Mozilla Firefox	Web browser	2002
QGIS	Geographic information system (GIS)	2002
LLVM	Compiler	2003
Mozilla Thunderbird	Email client; Personal information manager	2003
WordPress	Content Management System (CMS)	2005
Bootstrap	Front-end framework	2011
LibreOffice	Productivity Suite	2011
OpenBLAS	BLAS implementation (Linear algebra)	2011
React	JavaScript library	2013
TensorFlow	Machine learning library	2015
Project Jupyter	Shell	2015
Atom	Text Editor; Source code editor	2016
Hugo	Static Site Generator	2017

A lot of reuseable code is also created as part of the ongoing work of the Federal Government. As of late July 2018, more than 4,000 separate software projects are shared for reuse on the website, Code.gov, as part of an effort to make custom-developed code broadly available across the Federal Government.

Table 2. Contributions to Code.gov by lines of code.



Open Source Projects by Federal Government Organization as Posted on Code.gov
for projects started before January 1, 2018

Organization Name	Total Projects on Code.gov	Number of Projects linked to Github Collection	Kilo-lines of code (kloc)	Commits	Number of Contributors
Total	4,457	2,688	2,486,209.95	950,625	8,292
General Services Administration	1,501	1,368	266,860.27	318,676	4,631
Department of Energy	899	704	1,219,834.85	485,726	2,433
Consumer Financial Protection Bureau	261	243	753,447.22	49,781	334
National Aeronautics and Space Administration	998	141	179,916.58	51,936	358
Environmental Protection Agency	156	61	14,327.45	4,711	78
Department of Labor	72	52	5,663.03	1,185	50
Department of Homeland Security	19	19	8,051.15	2,526	63
National Archives and Records Administration	19	19	3,001.92	9,324	50
National Security Agency	19	19	5,133.72	3,725	36
Department of Agriculture	21	11	3,499.21	4,361	23
Department of Defense	13	10	5,555.56	4,965	46
National Science Foundation	98	6	54.42	28	2
Department of Transportation	7	5	994.28	1,903	15
Department of Health and Human Services	27	4	193.55	654	25
Department of Justice	6	4	33.25	240	1
Department of Veterans Affairs	4	4	2,294.33	429	28
Small Business Administration	16	4	397.37	1,058	13
Executive Office of the President	4	3	501.40	549	25
Office of Personnel Management	3	3	5,134.72	2,022	18
Agency for International Development	5	2	2.21	10	1
Department of Commerce	3	2	0.18	14	4
Department of the Treasury	2	2	11,177.75	6,007	24
Department of Education	2	1	110.60	787	33
Social Security Administration	129	1	24.92	8.00	1
Department of Housing and Urban Development	172	0			
Nuclear Regulatory Commission	1	0			

Table 2 shows the number of projects listed by each Federal Government agency on Code.gov. These projects include both code developed within the Federal Government and code created through contracting. The table shows number of projects for each agency as well as lines of code, commits (incorporated improvement to existing code), and contributors. Based on lines of code, the Department of Energy (DOE) contributed the most, two large DOE projects are Raven, statistical software for risk analysis in nuclear reactor systems, and Qball, which uses molecular dynamics to compute the electronic structure of matter. The projects also include applications built from existing OSS projects; for example, the US government's data portal, www.data.gov, is an OSS project built from WordPress.

Repositories and Source Code Hosting

While some government software is controlled through restrictive access requirements on government-controlled repositories, as Table 2 shows, many of the Federal government's OSS projects are shared on GitHub. GitHub, SourceForge, and Bitbucket are the most widely-used source code hosting platforms that allow OSS projects to be shared. These platforms are used to develop, download, review, and publish OSS projects and computer code. They host both private repositories and free accounts and provide access control and several collaboration features such as bug tracking, web-hosting, feature requests, task management, and wikis for every project. They use version control systems, such as Git, for tracking changes and coordinating work on files among multiple developers. GitHub is by far the largest hosting facility, with 31 million users and developers worldwide (Octoverse, 2018a). GitHub is shown with other sharing platforms and their scale in users and projects in Table 3.

Table 3. Source Code Hosting Platforms and Users

Platform	Company	Users/Developers	Number of Projects
GitHub.com	Microsoft	31 million	96 million
Bitbucket.org	Atlassian	5 million	N/A
SourceForge.net	Slashdot Media	3.7 million	500 thousand
GitLab.com	GitLab	100 thousand	546 thousand

Notes:
GitHub.com: Octoverse, 2018a. "The State of the Octoverse." https://octoverse.github.com/
Bitbucket.org: Bitbucket, 2017. "Bitbucket Cloud: 5 million developers and 900,000 teams." https://bitbucket.org/blog/bitbucket-cloud-5-million-developers-900000-teams. Retrieved 2017-03-25.
SourceForget.net: Alexa, 2018. "How popular is SourceForge.net?" alexa.com. Retrieved 2018-12-25.
GitLab.com: GitLab, 2016. "2015 was a great year at GitLab!" https://about.gitlab.com/2016/02/11/gitlab-retrospective/. Retrieved 28 July 2016.

Counting Packages, Downloads, and Linkages

Public investment in the US in research, technology, and the tools needed for this work comes at the cost of other national priorities, thus policymakers are interested in data that can help them evaluate progress and impact. Neither progress nor impact is easy to measure in dollars. In comparing science and engineering activity in the US and China, the National Science Board highlights numbers of peer-reviewed science and engineering publications in addition to R&D expenditures (NSB, 2018). Numbers of publications are an indicator of S&E activity, and citations from other work to a focal publication are indicators of impact. For patents, Hall, Jaffe, and Trajtenberg (2001) released an electronic data set for US Patent and Trademark Office (USPTO) patents along with methodological information for its use in the NBER patent citation file. An intersecting literature drives increasingly sophisticated analysis and visualization of data sets released from patent offices across the world. Our methods build on these foundations as well as the methods of extraction, interpretation and analysis of empirical data from software source code described by Ghosh et al. (2002). These approaches have also been applied to assign credit to OSS contributors.[2]

Data and Methods

We define OSS as computer software with its source code made available with a license in which the copyright holder provides the rights to study, change, and distribute the software to

[2] Depsy is a proof of concept project introduced in 2015 with NSF-funding that tracked the impact of research code using citations and other impact measures (Piwowar and Priem, 2016). More recently, the IEEE uses similar methods to create an indicator of the prevalence of different programming languages in the references of conference and journal articles, taking advantage of their digital library, IEEE Xplore (IEEE Spectrum, 2018b).

anyone and for any purpose. For this paper, we treat as open source any software language or package with an Open Source Initiative (OSI)-approved license.[3]

Table 4. Licenses

Name	License	Initial Release
Perl	Artistic-2.0	1987
Python	Python-2.0	1990
Haskell	BSD-3-Clause	1992
R	GPL-2.0	1993
LUA	MIT	1993
PHP	PHP-3.0	1995
MySQL	GPL-2.0	1995
Ruby	BSD-2-Clause	1995
Scala	BSD-3-Clause	2004
OpenJDK	GPL-2.0	2007
Go	BSD-3-Clause	2009
Rust	MIT	2010
Julia	MIT	2012
Swift	Apache-2.0	2014

Notes:
BSD: Berkeley Software Distribution
MIT: Massachusetts Institute of Technology
GPL: GNU Public License
PHP: A special license for the PHP scripting language

Table 4 lists the type of licenses and year of initial release for several popular OSS languages. BSD (Berkeley Software Development) and MIT licenses were developed in these universities. GPL is the GNU Public License, developed by Richard Stallman and the Free Software Foundation (Tozzi, 2017). Artistic, PHP, Python, and Apache are project-specific licenses that conform to the OSI standards.

There are hundreds of programming languages used for various purposes, with constantly changing popularity rankings. Table 5 summarizes the most widely used rankings and the data sources and methods used. We observe close (not perfect) alignment in the language rankings based on the various methods.

[3] The Open Source Initiative is an organization that reviews software licenses for compliance with this definition.

Table 5. Comparison of Software Language Rankings

Popularity Ranking	**Language Coverage**	**Data Sources**	**Method**	**Top 10 languages (2018)**
The IEEE Spectrum Top Programming Languages interactive app covers contexts that include social chatter, open-source code production, and job postings (IEEE Spectrum 2018a).	Starting from a list of over 300 programming languages gathered from GitHub, they remove languages with a very low number of search results in Google and eventually track 47 languages. (IEEE Spectrum 2018b)	9 sources: Google Search, Google Trends, Twitter, GitHub, Stack Overflow, Reddit, Hacker News, CareerBuilder, and IEEE Xplore Digital Library (with over 3.6 million conference and journal articles).	The popularity is calculated by searching the name of the language in the data sources, e.g., number of hits on Google Search, mentions in the journal articles.	Python, C++, Java, C, C#, PHP, R, Scala, Go, MATLAB
TIOBE index is created monthly and maintained by the TIOBE Company based in the Netherlands (TIOBE 2019a).	TIOBE tracks over 256 programming languages that satisfy certain requirements. such as having an own entry in Wikipedia and having at least 5,000 hits on Google (TIOBE 2019b).	25 most popular search engines including Google, MSN, Yahoo!, Wikipedia and YouTube.	The ranking scores are calculated by counting hits, i.e., the number of web pages with the language name.	Java, C, Python, C++, VB.Net, JavaScript, C#, PHP, SQL, Objective-C
Stack Overflow Developer Survey (Stack OverFlow 2018). Stack OverFlow is a forum-based tool primarily used to help solve coding problems.	The ranking lists top 25 languages (the survey question is not provided)	Survey of 101,592 software developers from 183 countries around the world. Recruited primarily through channels owned by Stack Overflow.	Survey question asks respondents to select all languages that apply; 78,334 responses	JavaScript, HTML, CSS, SQL, Java, Bash/Shell, Python, C#, PHP, C++, C
Octoverse report by GitHub Data Science Team provides annual trends and insights into GitHub activity (Octoverse 2018a).	337 unique programming languages on GitHub (Octoverse 2017)	GitHub repositories, contributors and pull requests (contributions to an open development project)	Based on the opened pull requests and by number of unique contributors to public and private repositories tagged with the primary language (Octoverse 2018b)	JavaScript, Java, Python, PHP, C++, C#, Typescript, Bash/Shell, C, Ruby
The PYPL Popularity of Programming Language **Index** (PYPL 2019)	The index is currently limited to 22 languages	Google Trends	Created by analyzing hits of language ‘tutorials’ on Google.	Python, Java, JavaScript, C#, R, C/C++, Objective-C, Swift, PHP, MATLAB
OpenHub is a public directory that provides statistics on different free/libre and open-source software (FLOSS) projects (OpenHub 2018).	Provides statistics on 112 languages	Active OSS projects on OpenHub	Based on commits, contributors, lines of code changes, and the total number of new projects	Python, Bash/shell, HTML, JavaScript, C, C++, Java, PHP, Ruby, Perl

Next we show more detail for two of these rankings, the IEEE Rating and the TIOBE Index. As Table 5 shows, the IEEE Spectrum ranking is a weighted index of 11 metrics, Google

Trends, Twitter, GitHub, Stack Overflow, Reddit, Hacker News, CareerBuilder, Dice, and the mentions of each language in IEEE Xplore Digital Library. TIOBE rankings (TIOBE stands for "The Importance of Being Earnest") are a measure of popularity of programming languages, created monthly and maintained by the TIOBE Company.

Table 6. IEEE Rating and TIOBE Index for Programming Languages

Name	IEEE spectrum	TIOBE Index	Developer	Initial Release	Open vs. Proprietary
Python	100	8.29%	Python Software Foundation (Non-profit organization)	1990	Open
C++	99.7	8.16%	Bell Labs (Private company)	1985	Open
Java	97.5	16.90%	Oracle Corporation	1995	Open
C	96.7	13.34%	Bell Labs (Private company)	1972	Open
C#	89.4	3.28%	Microsoft	2000	Open
PHP	84.9	2.68%	Zend Technologies (Private company)	1995	Open
R	82.9	1.33%	R Foundation (Non-profit organization)	1993	Open
Scala	82.6	n/a	École Polytechnique Fédérale de Lausanne (University)	2004	Open
Go	76.4	1.12%	Google	2009	Open
MATLAB	72.8	1.50%	MathWorks	1984	Proprietary
JavaScript	72.1	3.30%	Mozilla Foundation (Non-profit organization)	1995	Open
Ruby	71.4	1.10%	Yukihiro Matsumoto, et al.	1995	Open
HTML	71.2	n/a	W3C (Int'l Standards Organization)	1993	Open
Bash/Shell	66.1	n/a	Brian Fox	1989	Open

Table 6 summarizes the top 15 languages based on IEEE Spectrum (IEEE Spectrum 2018a) and TIOBE rankings (TIOBE 2019a), sorted by the former. The TIOBE Index includes languages with more than 5,000 search hits on Google. The top 15 languages of TIOBE index listed in

Table 6 account for 73% of the search hits in Google for all languages, i.e., the webpages that contain the name of the language. For example, Python accounts for 8.3% of the hits.

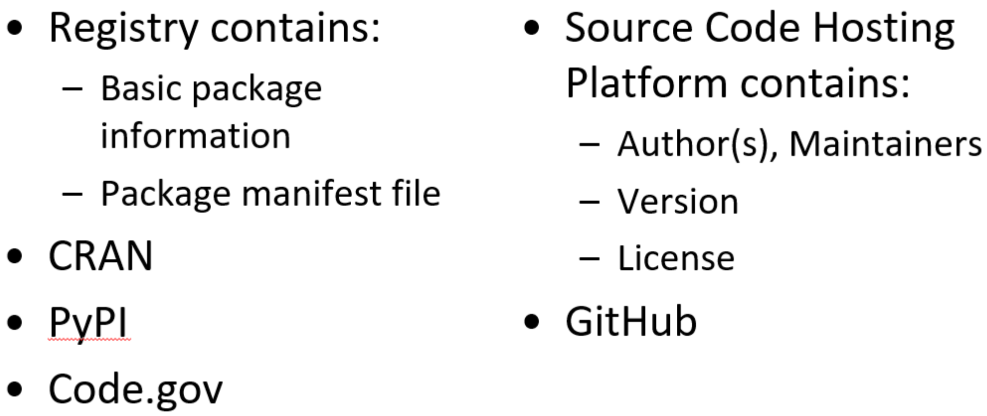
We focus on two programming languages, R and Python, that are themselves open source and are also used by others in further development. The R language is a set of tools for statistical analysis and mathematical modeling that has grown rapidly in the last decade. It is functionally a substitute for statistical software such as SAS and SPSS, except that it can be used free of charge. Since its release in 2000, users around the world have developed packages for it that are shared with the whole R community. It was developed at the University of Auckland in New Zealand by academics for use in their teaching laboratory; over time it had extended development by others, including Hadley Wickham, at Rice and Stanford Universities (Ihaka, 1998, Wickham, n.d.). As additional contributors extend R's functionality through additional packages. the Comprehensive R Archive Network (CRAN) emerged at the Technical University of Vienna in Austria; production ready packages are hosted from this site.

Python is one of the most widely-used programming languages mainly due to its simple syntax that makes its code easy to learn and share, and its flexibility. Like R it is also used for data analysis and visualization, however compared to R, Python's simplicity makes it a good general-purpose language for other purposes, including scripting and web development. Some of the most popular packages include *django* which is a tool for building web applications, *pandas* that provides high-performance data structures and analysis tools; and *scikit-learn* used for machine learning. Python is the successor to the ABC language (ABC, 2018), was developed by a Dutch programmer at National Research Institute for Mathematics and Computer Science in Netherlands (Centrum, 2018) in the late 1980s (Rossum, 2009). PyPI is an indexed repository for Python packages. There are over 166K projects and 299K contributors on PyPI as of February 2019 (PyPI, 2019).

Data Collection and Preparation

Keller et al. (2018) describes the overall approach used here to explore data sources beyond surveys to improve and extend indicators of science and engineering activity and of innovation. This approach includes structured processes to discover, acquire, profile, clean, link, explore the fitness-for-use, and statistically analyze the data. Here we gather and use publicly available metadata about individual packages and their contributors, as well as information within the code.

The natural way to obtain the information about the development of an OSS project is to inspect the repository that hosts the code for that package or application. The first step is to catalog all projects available to the programming language. This information is stored by a ***registry***, where all the packages are stored (the universe of packages for the language). For example, Comprehensive R Archive Network (CRAN) and Python Package Index (PyPI) are registries used to distribute the R and Python packages, respectively. Every programming language has the ability to install additional packages from the registry by using that particular languages' ***package manager***. Package managers take the package that the user wants to install and finds the package in the registry. The package manager obtains all the information that is needed by using the ***metadata*** stored on the registry, such as a unique identifier (usually the package name), a release version to identify what version of the package should be retrieved, and the ***repository*** location (where to find the actual code). The metadata stored on the registry also holds information such as name, author, maintainer, license, description, dependencies, and project status. These are shown in Figure 2.



- Registry contains:
  - Basic package information
  - Package manifest file
- CRAN
- PyPI
- Code.gov

- Source Code Hosting Platform contains:
  - Author(s), Maintainers
  - Version
  - License
- GitHub

Figure 2. Package Managers and Repositories

We use the registries of R and Python: (i) CRAN at https://cran.r-project.org/web/packages/available_packages_by_name.html, and (ii) PyPI at https://pypi.org/simple/. We use the information provided in the registries to identify the list of packages that are production-ready, i.e., not in development stage (different heuristics were used for different registries), and that have Open Source Initiative (OSI)-approved licenses (which is given in the package registry metadata). Using the information in the metadata, we find the set of packages that have their code bases on GitHub to obtain development activity and perform a lines-of-code count for analysis. Table 7 shows the number of R and Python packages we collected from their respective registries, the number of OSI-approved packages that are production ready, and the subset of these packages that are also on GitHub. The number of packages used in our analysis for each language is given in the final column.

Table 7 Data Collection Summary

Data Collection						
Language	Package manager	Number of packages	Production ready	OSI-approved & production ready	Packages on GitHub	Packages on GitHub (analysis)
R	CRAN	13,719	13,350	13,143	4,407	4,358
Python	PyPI	164,836	17,482	15,043	11,016	9,773

We collected information about 13,719 packages on CRAN on February 16, 2019, successfully downloading 13,350 packages.[4] We use license information and the production status of the release versions of these packages provided on CRAN to obtain 13,143 production-ready packages with OSI-approved licenses. Of these, we subset those also on GitHub, and obtain 4,364 R package repositories.

The initial list of 164,836 Python packages was obtained on January 23, 2019 from PyPI (PyPI, 2019). The latest package source of each package was downloaded (between 2019-01-28 and 2019-01-30) and user-reported production status in the metadata source was used to identify production-ready packages. We obtain a list of 17,482 production ready/stable and mature Python packages. The license information of these packages is obtained using the libraries.io (a web-service compiling and providing information on Python packages), resulting in 15,043 production ready packages with OSI-approved licenses. We use the package source to obtain the GitHub repository locations of these packages (if exists), and we download 9,775 repositories on March 1, 2019. Using the downloaded repositories of R and Python packages, we obtain development information including the number of contributors, the lines of code (added and deleted), number and time of commits (incorporated improvement to existing code from contributors),

[4] In the time it takes us to download, as of February 28, 2019, we lose 244 packages that had an update during the 12-day period which led to the change in their URL's on CRAN; hence they could not be downloaded.

and profile of contributors. The lines of code can be used as a measure of output, as we will describe.

Labor time, Cost and Impact Estimates

We are interested in resource cost estimates for OSS software that will be comparable with expenditures for R&D and with fixed investment data in intellectual property products. The US National Accounts production costs for own account software include those for analysis, design, programming and testing, and exclude maintenance and repair (Parker and Grimm, 2000). As originally described, cost of production is the sum of labor costs and intermediate inputs (such as materials and supplies and overhead). In US economic statistics, these costs are estimated based on hours worked by computer programmers and system analysts in each industry or government entity. The underlying assumption is that own-account software is created as a fixed proportion of the work activity of these occupations. Mean wage rates, adjusted for compensation costs that include fringe benefits are multiplied by the number of computer programmers and system analysts in each industry.[5]

Constructive Cost Model (COCOMO)

The challenge of keeping large software projects on schedule and within budget motivates a literature in cost estimation within software engineering (Sharma, Bhardwaj, Sharma, 2011). Experience has shown that while costs can be estimated as a function of the number of instructions, as software projects grow, effort increases nonlinearly. We observe development of cost models that account for complexity, reliability, and scale in a variety of

[5] Wage, employment and compensation are from the U.S. Bureau of Labor Statistics Occupational Employment Survey data. To account for time spent on tasks other than software development, BEA uses an adjustment ratio from a survey of software developers' time. Non-labor costs for OSS development are estimated with industry production ratios (Parker and Grimm 2000).

ways based on characteristics of the product, the platform, the contributors, and the project. Examples of these estimation models include Constructive Cost model COCOMO II, the Putnam Software Life Cycle Management model, and models based on function points (Boehm and Valerdi, 2008). This is the approach that we use here.

The logic of the constructive cost model is that:

$$Production\ time\ in\ PersonMonths =$$

$$Calibration\ factor\ x\ lines\ of\ code\ x\ effort\ multipliers\ .$$

The calibration factor represents the person months needed for a set number of lines of code, unadjusted for effort factors. The effort multipliers account for complexity, reliability, and scale for these models; they lead to increased cost. Translating this approach to our data on OSS, the package-specific data we collected provides lines of code for each completed package.

In our use of this model, we multiply lines of code by a COCOMO II calibration factor (Boehm et al., 2000) to estimate person-hours per package or project. The effort multipliers from COCOMO II are parameters that we selected for the organic software class which consists of software dealing with a well-known programming language and a small, but experienced team of contributors. While we held these consistent across all packages, the model allows for these parameters to be adjusted based on additional data.

$$\textbf{\textit{Effort = 2.4(KLOC)}}^{\textbf{\textit{1.05}}}$$

$$\textbf{\textit{Nominal development time = 2.5(Effort)}}^{\textbf{\textit{.38}}}$$

$$\textbf{\textit{Development cost = Monthly wage x Nominal development time}}$$

KLOC stands for kilo (thousand) lines of code. With these person-month calculations per OSS package, we estimate a resource cost by multiplying by monthly wages for programming occupations. Appendix Table 1 shows the steps and data sources for the estimation. To summarize our method, we assume that the input time of contributors is roughly equivalent to the average wage for computer programmers (from Bureau of Labor Statistics (BLS) (2017). Occupational Employment Survey data) plus additional intermediate input and capital services costs (from Bureau of Economic Analysis (BEA 2014). The per person month cost for OSS contribution is obtained as $19,963, which is the amount used in our cost calculations.

Lines of Code

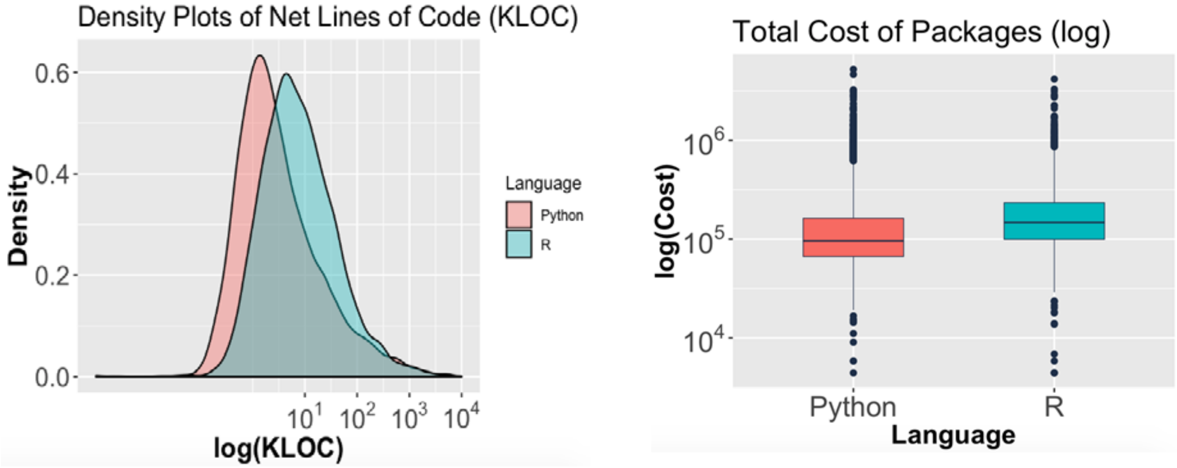
Lines of code track the scale of development activity for OSS packages. For each of the 13,143 R packages in Table 7, we counted the total number of lines in all the files that are included in the CRAN package source. These files include source code, description files, manuals, data files, citations, and images. With this method, we estimate the cost of almost all of the R packages that are on CRAN, providing a broad measure of the number of production ready OSS packages for the R language.

For a subset of these R packages, and for almost 10,000 Python packages, we collect more detailed data from GitHub. The packages' GitHub repositories provide development history including information about line insertions and deletions for accepted change or commit. Using the history of the repository, we obtain details of each commit which gives us the total number of lines edited, thus capturing the resources and the effort that were put in the development of each package. We can calculate costs with both the total number of lines edited (gross) and with the net lines of code (insertions minus deletions). We focus on the net lines of code as this measure is similar to the method (counting the number of lines in the package source files) used on the dataset collected from CRAN on R packages.

Using net lines of code as a measure of effort and the parameters for a well-known programming language and small, experienced team of contributors, we estimate the cost of all packages that use R and Python, which we view as order of magnitude estimates. We obtain power-law distributions for all languages. The density plots and the boxplots are given in Figure 3.

.

Figure 3 Distribution Costs across languages



Order of Magnitude Resource Cost Calculations for OSS Languages

Using the widest set of production-ready and OSI-approved R packages on CRAN and summing across number of lines in the packages' source files, we calculate a resource cost in 2017 dollars of $1.58 billion for 12,901 R packages. The packages with the highest cost estimates are given in Table 8. *Mapdata*, the package with the highest number of lines, is a supplement to other map packages, providing a high-resolution mapping tool. The *mapdata* package source files include geometry files involving points and lines to draw maps that results in a high count of lines. *Hunspell* is the spell-checker library used by other open-source applications including web-browsers such as Google Chrome, Mozilla Firefox, and the package includes dictionaries in English, among other languages. *EdgarWebR* is a package used for accessing and parsing Securities and Exchange Commission (SEC) filings, and the source files of the package include a lot of test code and data, which are very large files. The R packages that are written in C language for efficiency needs (faster computation) such as the *igraph* package used for network analysis, also have a large number of lines due to the structure (commonly used coding practices) of this language.

Table 8. CRAN Packages' Kilo lines and Estimated Resource Cost

Package Name	Klines	Estimated Cost in Thousands of 2017$
All packages	100,216.787	1,579,689
mapdata	2,257.20	1,516
hunspell	756.9	980
edgarWebR	456.8	801
TCGA2STAT	376.9	742
igraph	364	732

Limiting our estimates to the packages on GitHub and using net lines of code, we find a resource cost of $0.88 billion dollars for 4,364 R packages, and $1.56 billion for 9,775 Python packages based on 2017 US wage rates (though, as we will show, contributors come from many countries). The R and Python packages with the highest cost estimates are given in Table 9. Among the top R packages, *Archivist* is used to store data artifacts, *CollessLike* for analyzing genetic trees, *readtext* used for text files and *ptwikiwords* is a dataset with words used pages from the Portuguese Wikipedia, and *nasapower* is used for global meteorology, surface solar energy and climatology data. The high number of changes in lines of code is due to the database files, images and datasets uploaded that are used for testing. Similarly, the top Python packages, such as *libsass* (style files used for web-development), *py3-ortools* (research tools developed at Google including programming algorithms), *LSD-Bubble* (used in astronomy), *IotPy* (used to develop applications using sensors an social media data), and *openquake.engine* (computing earthquake hazard and risk) include datasets and documentation.

Table 9. R and Python Packages on GitHub: Kilo lines of code (KLOC) and Estimated Resource Cost

R Packages on GitHub			Python Packages on GitHub		
Package Name	KLOC	Estimated Cost in Thousands of 2017$	Package Name	KLOC	Estimated Cost in Thousands of 2017$
All packages	282,167.871	883,209	All packages	611,601.568	1,560,374
archivist	28488.639	4,169	libsass	50340.53	5,233
CollessLike	15844.721	3,299	py3-ortools	37412.424	4,648
readtext	13888.309	3,130	LSD-Bubble	15270.398	3,251
ptwikiwords	11452.965	2,898	IotPy	14899.252	3,219
nasapower	10613.638	2,812	openquake.engine	13841.578	3,126

Packages on Code.gov

Using the same approach, we estimate the cost for the OSS projects on Code.gov that are hosted on GitHub, shown earlier in Table 2. Since many of the projects contributed by Federal Government organizations have been developed by contractors as custom software, this resource cost is not an estimation of what the government actually paid for these software projects. Rather, it gives an order of magnitude cost estimate consistent with own-account software that allows comparison with the OSS language packages described in the previous section. We estimated a resource cost for these 2.5 billion lines of code at about $1.1 billion dollars, calculating all contributions at the rate of 2017 costs. This is a partial estimate of all the contributed projects, because our calculation is only for those projects on GitHub.

To sum up, we calculate between about a billion dollars and a billion and a half (0.88 billion and 1.58) resource cost for R, another 1.5 billion for Python and more than a billion for Code.gov. These lines of code provide an ongoing set of snap shots of the scale of development activity for OSS packages. As a unit of output has the virtue of be being comparable across packages and projects, as long as it is counting the same objects within files. Our current method does not yet distinguish between lines of computer instructions, documentation, generated output, and data files.

Impact

OSS is generally distributed without cost, and so standard market measures of revenue cannot provide an impact measure. Piwowar and Priem (2016) use downloads and citations to software as measures of impact for Python and R packages. When software code is assigned a digital object identifier (DOI), it can be cited along with other reference sources, providing an indicator of impact to the scientific community. However, few academic papers actually cite software (Piwowar and Priem (2016).

Downloads provide an indicator of impact that is broader than scientific impact. Using downloads as a measure of impact, Korkmaz, et al. (2018) analyze factors that affect the impact of R and Python packages. They find that three network measures, outdegree, closeness centrality, and pagerank significantly affect impact of packages. Outdegree is a count of reuses across packages, while closeness centrality and pagerank assess more complex network relationships.

Estimating Impact through Reuse: From the perspective of the users of OSS as a set of tools, ‘reuse of packages’ is a measure of the impact and value through the network. The greater the reuse by other packages, the greater the value. When an OSS package requires the code of a second package to do its work, the first package is dependent on the second. For example, an R package for statistics as well as an R package for inventory may be both be dependent on the same visualization package, such as ggplot2. This reuse, through multiple dependencies and imports, increases the visualization package’s impact. The reverse dependency, reuse, is used as a measure of the impact. Table 9 show the R packages (OSI-approved and production ready) with the highest number of reuses.

Table 9 Top packages (Impact Based on Reuse)

Package	# Reuse
ggplot2	925
Rcpp	838
dplyr	626
stringr	398
plyr	395
magrittr	393
data.table	311
sp	308
reshape2	271
foreach	266

Note: The dependency information is obtained from the manifest files described earlier (it is given as “depends” and “imports”). We consider both “imports” that load a package and “depends” that attach a package for our dependency measure. Standard libraries that are supplied with R (e.g., stats, utils, graphics) are removed.

Here we are considering only the number of packages that reuse a particular package as its impact measure. When we consider these dependencies, a more complete analysis should account for the interactions between all of these packages (not only the bilateral relationship between two packages in isolation) and the structural features of these interactions. Network analysis allows us to use various centrality measures (e.g., betweenness, pagerank) in addition to the degree centrality (basic counts of dependencies) as impact measures and provides a fuller picture of impact (see Korkmaz et al. 2018).

Estimating Impact through Downloads: Downloads is a measure of end-user impact. Figure 5 shows total downloads of Base R from CRAN between 2013 and 2018; in 2018 base R was downloaded over a million times over 400,000 of them in the US.

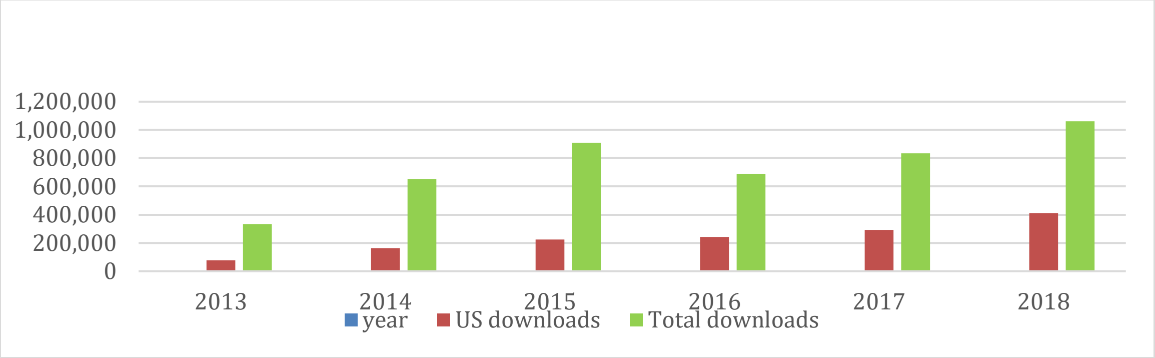


Figure 6. Downloads of Base R from CRAN 2013 – 2018

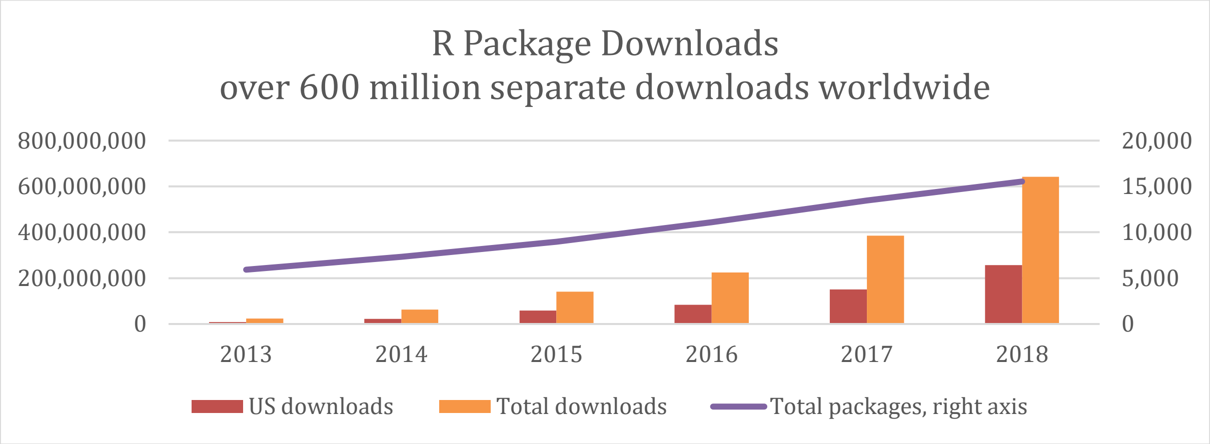


Figure 7. Downloads of R Packages 2013 – 2018

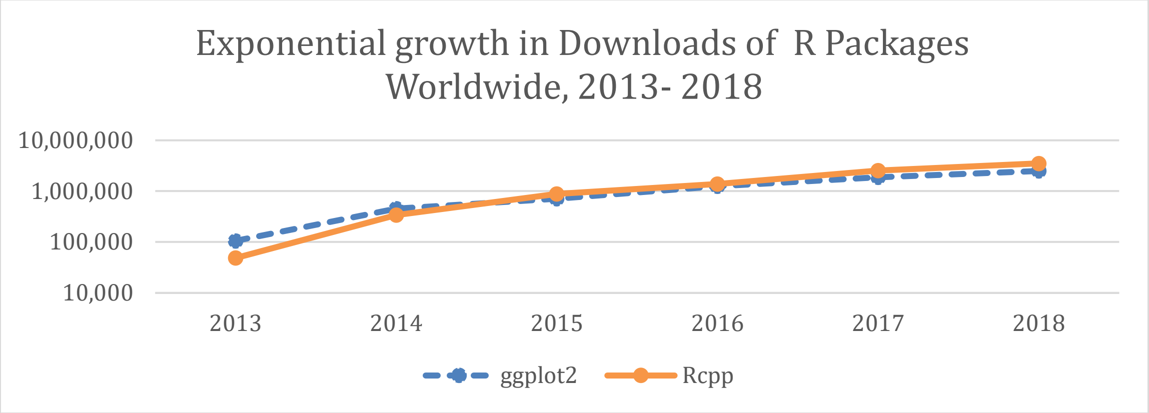


Figure 8. Downloads of R Packages 2013 – 2018

The package downloads below are data collected from between 2013 and 2018. They show the rapid growth overall in the use of these packages. Note that ggplot2 was the most downloaded package in 2013 with 105,774 downloads (Table 10). In 2018 *ggplot2* was downloaded an order of magnitude more often, almost 2.5 million times.

Most Frequently Downloaded R Packages, 2013 and 2018

Package	2013	Package	2018
ggplot2	105,774	Rcpp	3,519,510
plyr	101,596	rlang	2,893,889
digest	99,774	stringi	2,610,184
stringr	98,086	stringr	2,511,011
colorspace	93,590	ggplot2	2,495,315
RColorBrewer	81,448	digest	2,453,958
reshape2	81,350	glue	2,296,688
scales	73,385	tibble	2,242,376
proto	71,698	pillar	2,222,364
munsell	71,483	yaml	2,207,621

Table 10. Top R Packages, Based on Downloads

Downloads is a direct, though noisy measure of impact. A single user may reload packages several times for use on different platforms. However, downloads provide a clear measure of the diffusion of packages.

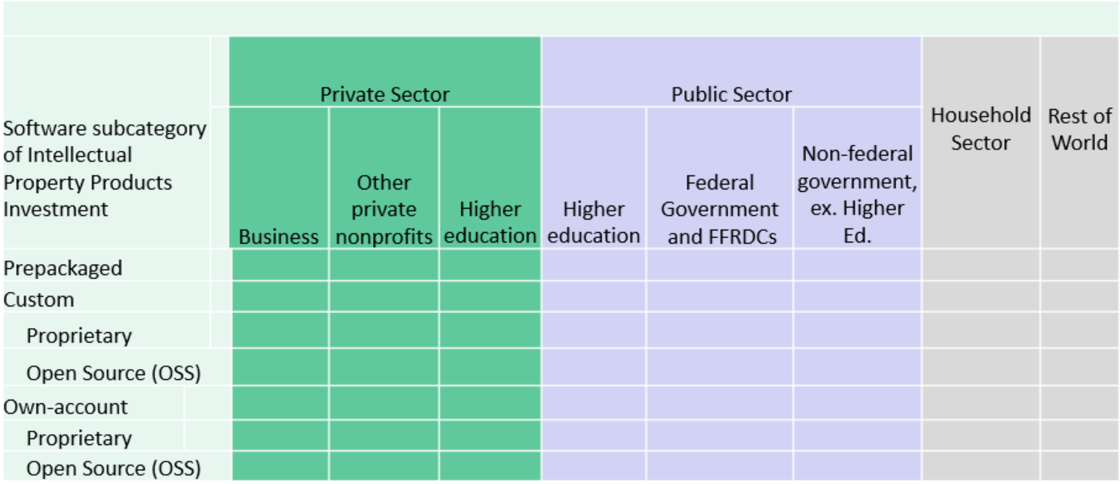
Sectoral Story for R

OSS software is created as tools for the developer’s own use, as custom software created in the market, and also as means for developers to signal their skills to potential employers. Firms develop OSS as well; allowing them to sell complementary hardware or consulting services (Lerner and Tirole, 2002). For example, for-profit vendors may offer a free download of OSS along with commercial support or dedicated servers. Companies investing heavily in OSS include Microsoft, Red Hat, and Oracle, Google, General Electric, Dell, Sony, Nokia, Ericsson as well as non-technology specific companies like Volkswagen, Bosch, and BMW

(Nagle 2018). Microsoft’s 2018 purchase of GitHub shows industry sees further potential in OSS.

However, LaTeX, Linux, Apache, and R were all developed in university, government, or nonprofit institutions, some outside of the US. To fully understand the inputs to the digital economy that come from public funding, we need to understand how well the national accounts are tracking OSS in all sectors. Figure 10 shows a sectoral framework that embeds OSS within the scope of software investment. It starts with the BEA framework, then adds two rows and two columns to the BEA categories. In the rows we add OSS as a subcategory of custom software it is developed by a market producer, and as a subcategory of own-account software when it is developed in house. Conceptually, BEA’s 2017 estimate of about $381 billion in software investment should include OSS produced by businesses, government entities, and nonprofit institutions. As described earlier, these BEA investment measures are based on receipts for software sold in the market and, for own-account software, based on employment counts and wage rates for computer programmers and systems analysts.

So how much OSS is created or funded by governments and nonprofits? While BEA does not publish the composition of public investment in software, it’s methodology for public investment is the same as for private, that is to say, software transactions and the count and wages of computer programmers and system analysts. Since BEA the subcomponents of government and nonprofit software investment are not released, we don’t actually know. However, our analysis suggests to us that it may be substantially undermeasured. In a 2014 survey of United Kingdom research software engineers, defined as academics who write software used by researchers, found that these software writers had their highest degree in physical science (39.4%) compared with computer science (23.6%) (Philippe, et al, 2015).



Software subcategory of Intellectual Property Products Investment	Private Sector			Public Sector			Household Sector	Rest of World
	Business	Other private nonprofits	Higher education	Higher education	Federal Government and FFRDCs	Non-federal government, ex. Higher Ed.		
Prepackaged								
Custom								
Proprietary								
Open Source (OSS)								
Own-account								
Proprietary								
Open Source (OSS)								

Figure 9. Software Investment Framework, Augmented to Show OSS Producers

To identify the sectors where developers are contributing to OSS, we take a closer look at the manifest data we collected from CRAN for R packages. We obtain 11,886 OSI-approved production-ready packages from CRAN[6] published between 2005-10-29 and 2018-06-18, and collect information about these packages including the license, published data, authors and their roles (creator or maintainer, contributor, copyright holder), and the email address of the maintainers, dependencies (imports, suggests, depends), and URL's to the repositories.

Every package has at least one author and one maintainer listed; they may be the same person. The creator or maintainer is the person to be contacted if there are problems, hence they must provide an email address. We use the email addresses to obtain information about the location and organization of the creators. This approach gives us a lower boundary on university contributions, and a substantial share of email addresses that are insufficient to identify sector.

In our dataset, we obtain 6,697 unique maintainers associated with 6,871 unique email addresses (378 have more than one email addresses). There are 2,261 unique domains (e.g.,

[6] https://cran.r-project.org/web/packages/available_packages_by_date.html

gmail.com, yahoo.com, outlook.com), and 103 top-level domains (e.g., *.com, .edu, .org, .uk*). One third (32%) of maintainers have Gmail accounts; these can be from any sector. To get a sense of geographical distribution, we obtain a complete list of all country top-level domains maintained by the Internet Assigned Numbers Authority (IANA)[7] Table 11 summarizes the number of projects and number of creators for the most common top-level domains. The largest share of package maintainers has *.com* email addresses; these provide little information about geography or economic sector. However, 17% of both packages and maintainers have *.edu* email addresses, and more than a third have email addresses that are country specific.

Table 11 Top-level Domains of R Package Maintainers on CRAN

Domain	Packages	Percent	Maintainers	Percent
Total	11,886.0		6,697	
.com	4,964	42%	2,770	40%
.edu	1,981	17%	1,202	17%
.org	481	4%	184	3%
.net	168	1%	89	1%
.gov	69	1%	43	1%
.name	33	0%	3	0%
.info	8	0%	6	0%
.biz	6	0%	3	0%
.(country)	4,124	35%	2,495	36%
Germany (.de)	687	6%	427	6%
United Kingdom (.uk)	434	4%	267	4%
France (.fr)	398	3%	235	3%
Canada (.ca)	335	3%	160	2%
Australia (.au)	198	2%	109	2%
Italy (.it)	198	2%	129	2%
Switzerland (.ch)	172	1%	102	2%
Spain (.es)	166	1%	102	2%
Netherlands (.nl)	151	1%	89	1%
Austria (.at)	123.0	0.0	56.0	1%

[7] https://www.worldstandards.eu/other/tlds/ IANA is responsible for the global coordination of the DNS Root, IP addressing, and other Internet protocol resources.

A country domain is associated with 35% of contributions (packages) and 36% of the maintainers. Table 11 also shows the top 10 countries (out of 88 countries) that have the highest number of contributions (packages) and the number of maintainers. These exclude projects with an associated .com email addresses (42% of all projects) Finally, we analyze the creators' email domains with .edu., we obtain 1,981 packages (16.7%) created by 1,202 maintainers.

Although we cannot parse the Gmail addresses by sector, we find almost 17% are associated with university domains, and more than one third come from country domains. We view these shares as a lower boundary on the sector's contribution to OSS. We add two columns on the right that show that a full accounting will need to include OSS created outside of work and OSS created internationally.

From Resource Cost to Investment

Annual Investment: Resource cost estimates presented here for R and Python are aggregated across several years and treated as though all expenditures were made in 2017. To move to annual investment measures, each year's resource costs will need to be deflated with prices for the appropriate year.

Deflators: GDP measures economic output after removing the impact of overall inflation. This is done product by product using deflation with price indexes which reflect the movement of prices separate from quality or volume. BEA deflates expenditures for prepackaged software with their producer price index for software publishing. Custom software and own-account software are deflated with a weighted average of the prepackaged software price and of an input-cost index based on BLS data on wage rates for computer programmers and systems analysts and on intermediate input costs associated with the production of software (BEA 2017).

Depreciation: Given that their useful life exceeds a year, expenditures for capital assets lead to accumulation over time, less depreciation, which accounts for loss over time in utility. Physical capital loses its value as its useful declines with wear and tear, and well as through obsolescence, as newer alternative assets emerge. Intellectual property products, such as software, R&D and entertainment originals lose their value through obsolescence. This can happen as costs for maintain the asset rise over time, or through a crease in the value of services provided, despite a constant flow in the quantity of these services (Oulton and Srinivasan, 2003). BEA uses an estimated 3-year service life for prepackaged software and a 5-year estimated service life for custom and own-account software (Soleveichik and Wasshausen, 2013). Li and Hall (2016) argue that for R&D investment, depreciation through obsolescence and competition can be estimate through the decline in firm profits received from prior R&D investments.

Conclusion

We have described the open source software ecosystem and showed how data gathered freely from ISS registries and repositories can be used to estimate the resource costs and impact of OSS. Data collected data from CRAN, PyPI. Code.gov and GitHub provide information about the development activity of OSS projects. Cost models developed in software engineering, and own-account investment measurement methods like those used by national accounts estimate the resource cost of these projects/packages. We find that the resource cost for developing packages for two well-known OSS programming languages, R and Python, exceeds $3 billion dollars. Applying this approach to OSS projects available on Code.gov results in an estimated value of more than $1 billion, based on 2017 costs.

While our work does not propose or imply any change the definition of Investment in BEA's GDP accounts, from the perspective of identifying sources of innovation and technology diffusion, we see value in quantifying OSS created both in the market and outside of it, as well as the international contribution. The value of our work for GDP measurement is in the use of alternative methodology and source data that focuses on a subset of an investment category in the national economics accounts. The bottom-up method we use may reveal some currently unaccounted-for software investment—and that may be of broader interest because it would affect the level and composition of software investment.

References

ABC. 2018. "ABC programming language." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 30 Jul. 2018. https://en.wikipedia.org/w/index.php?title=ABC_(programming_language)&oldid=852622792 (last visited Feb. 4, 2019).

Boehm, Barry. 1981. Software Engineering Economics (Englewood Cliffs, NJ: Prentice-Hall, 1981): 533-35, 548-50.

Boehm, Barry, Abts, C., Brown, W. Chulani, S, Clark, B. Horowitz, E., Madachy, R, Reifer, D. and Steece.B., 2000. Software Cost Estimation with COCOMO II (with CD-ROM). Englewood Cliffs, NJ:Prentice-Hall

Boehm B., and R. Valerdi, 2008. Achievements and Challenges in Cocomo-Based Software Resource Estimation. IEEE Software 25(5): 74-83 (2008)

Bureau of Economic Analysis. 2013. Preview of the 2013 Comprehensive Revision of the National Income and Product Accounts: Changes in Definitions and Presentations. Survey of Current Business, March 2013.

Bureau of Economic Analysis. 2014. "2007 Input-Output Tables," IOUse_Before_Redefinitions_PUR_2007_Summary

Bureau of Economic Analysis. 2017. Concepts and Methods of the U.S. National Income and Product Accounts, November 2017.

Bureau of Economic Analysis. 2018a. Intellectual Property Products Fixed Asset Tables (private), last updated November 28, 2018.

Bureau of Economic Analysis. 2018b. Table 7.5B. Investment in Government Fixed Assets, Last Revised November 20th, 2018

Bureau of Labor Statistics, 2017. Mean Annual Wage Series, https://www.bls.gov/oes/2017/may/oes_nat.htm#15-0000

Centrum Wiskunde & Informatica. 2018. *Wikipedia, The Free Encyclopedia*. 23 Nov. 2018. https://en.wikipedia.org/w/index.php?title=Centrum_Wiskunde_%26_Informatica&oldid=870200085 (last visited Feb. 4, 2019).

Corrado, C. Haskel, J. & Jona-Lasinio, C. 2015 Public Investment in Intangible Assets. EPWP #15 – 01 Economics Program, [https://www.conference-board.org/pdf_free/workingpapers/EPWP1501.pdf]

Corrado, Carol, Charles Hulten, and Daniel Sichel, 2005, "Measuring Capital and Technology: An Expanded Framework," in Measuring Capital in the New Economy, Carol Corrado, John Haltiwanger, and Dan Sichel, editors, University of Chicago Press.

Ghosh, R. A. R. Glott, B. Krieger, and G. Robles. 2002. *Free/Libre and Open Source Software: Survey and Study Report. Part IV.* [http://www.infonomics.nl/FLOSS/report/]

GlobalStats statcounter, 2018. "Operating System Market Share Worldwide Operating System Market Share Worldwide - June 2018," (http://gs.statcounter.com/os-market-share) Accessed July 20, 2018.

Greenstein, S. and Ackermann, K. 2018. "The State of Open Source Server Software," Working Paper.

Greenstein, S. and Nagle, F. 2014. "Digital Dark Matter and the Economic Contribution of Apache" Research Policy, 43(2014) 623-631Hall, B. H., A. B. Jaffe, and M. Trajtenberg, 2001. "The NBER Patent Citation Data File: Lessons, Insights and Methodological Tools." NBER Working Paper 8498

IEEE Spectrum. 2018a. "Interactive: The Top Programming Languages 2018." https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2018

IEEE Spectrum. 2018b. "IEEE Top Programming Languages: Design, Methods, and Data Sources." https://spectrum.ieee.org/static/ieee-top-programming-languages-2018-methods

Keller, S., Korkmaz, G., Robbins, C., & Shipp, S. (2018). Opportunities to observe and measure intangible inputs to innovation: Definitions, operationalization, and examples. *Proceedings of the National Academy of Sciences*, *115*(50), 12638-12645.

Korkmaz, G., Kelling, C., Robbins, C. and Keller, S.A. 2018. Modeling the Impact of R Packages Using Dependency and Contributor Networks. In 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM) pp. 511-514.

[http://cmaptools.cicei.com:8002/rid=1203015502582_177114975_1313/BCG-HACKERSURVEY.pdf]

Lerner, J. and J. Tirole, 2002. Some Simple Economics of Open Source. The Journal of Industrial Economics 50 (2), 197-234.

Li, W, and Hall, B., 2018. The Depreciation of Business R&D and Capital, Review of Income and Wealth, https://doi.org/10.1111/roiw.12380

Li, Wendy C.Y. ,Makoto Nirei and Kazufumi Yamana, 2018. “Value of Data: There’s No Such Thing As A Free Lunch in the Digital Economy.” Paper prepared for the 6th IMF Statistical Forum, Washington DC.
https://www.bea.gov/papers/pdf/USSoftware.pdf. Accessed July 26, 2018.

Nagle, Frank, 2017. Open Source Software and Firm Productivity, Harvard Business School Research Paper No. 15-062.

Nakamura, Leonard Jon Samuels and Rachel Soloveichik, 2016. Valuing “Free” Media Across Countries in GDP BEA Working Paper WP2016-3

Nakamura, Leonard, Jon Samuels, and Rachel Soloveichik, 2017. “Measuring the “Free” Digital Economy within the GDP and Productivity Accounts.” BEA Working Paper WP2017-9

National Science Board, 2018. The Rise of China in Science and Engineering, two page fact sheet: https://nsf.gov/nsb/sei/one-pagers/China-2018.pdf

Octoverse. 2017. “The State of the Octoverse 2017: The Fifteen Most Popular Languages on GitHub.” https://octoverse.github.com/2017/

Octoverse. 2018a. “The State of the Octoverse.” https://octoverse.github.com

Octoverse. 2018b. “The State of the Octoverse: Top Programming Languages of 2018.” https://github.blog/2018-11-15-state-of-the-octoverse-top-programming-languages/

OECD and Eurostat, 2005. *Oslo Manual: Guidelines for Collecting and Interpreting Innovation Data, third edition.* OECD Publishing.

OECD 2010. *Handbook on Deriving Capital Measures of Intellectual Property Products*. OECD Publishing.

OECD/Eurostat, 2018, Oslo Manual 2018: *Guidelines for Collecting, Reporting and Using Data on Innovation*,4th Edition, The Measurement of Scientific, Technological and Innovation Activities, OECD. Publishing, Paris/Eurostat, Luxembourg.
https://doi.org/10.1787/9789264304604-en

Open Source Initiative, 1998. (https://opensource.org/osd).

OpenHub. 2018. “Compare Languages.” https://www.openhub.net/languages/compare

Oulton, Nicholas and Sylaja Srinivasan, 2003. Capital stocks, capital services, and depreciation: an integrated framework. Bank of England Working Paper no. 192

Parker R. and Grimm B. 2000. “Recognition of Business and Government Expenditures for Software as Investment: Methodology and Quantitative Impacts, 1959-98.” (https://www.bea.gov/papers/pdf/software.pdf)

Piwowar, Heather and Jason Priem, 2016. Depsy: valuing the software that powers science. https://github.com/Impactstory/depsy-research/blob/master/introducing_depsy.md

Philippe, O., N. Hong, and S. Hettrick. 2015. Preliminary Analysis of a Survey of UK Research Software Engineers, http://ceur-ws.org/Vol-1686/WSSSPE4_paper_19.pdf

PyPI. 2019. “Python Package Index.” https://pypi.org/

PYPL. 2019. “PYPL Popularity of Programming Language.” http://pypl.github.io/PYPL.html

Rossum, V. Guido. 2009. “A Brief Timeline of Python.” https://python-history.blogspot.com/2009/01/brief-timeline-of-python.html

Sharma, T., Bhardwaj, A, and Sharma, A. 2011. “A Comparative Study of COCOMO II and Putnam models of Software Cost Estimation. International Journal of Scientific and Engineering Research,” Volume 2, Issue 11, November 2011.

Soloveichik, Rachel, and David Wasshausen, 2013. Copyright-Protected Assets in the National Accounts. https://sites.nationalacademies.org/cs/groups/pgasite/documents/webpage/pga_063401.pdf

TIOBE. 2019a. “TIOBE Index for January 2019.” https://www.tiobe.com/tiobe-index/

TIOBE. 2019b. “TIOBE Programming Community Index Definition.” https://www.tiobe.com/tiobe-index/programming-languages-definition/

Tozzi, Christopher, 2017. For Fun and Profit, A History of the Free and Open Source Software Revolution, MIT Press, Cambridge.

W3Techs. 2018. “Usage statistics and market share of Apache for websites.” https://w3techs.com/technologies/details/ws-apache/all/all. Accessed July 23, 2018.
Wickham, Hadley. n.d. http://hadley.nz Accessed July 26, 2018