Measuring the Cost and Value of Open Source Software

Hannah Brinkley (VT), Keren Chen (VT), Eirik Iversen (VT), and Bayoán Santiago Calderón (Claremont Graduate University)
SDAL: Gizem Korkmaz, Daniel Chen, and Stephanie Shipp
Sponsor: Carol Robbins, The National Center for Science & Engineering Statistics (NCSES) at the National Science Foundation (NSF)

Objective

The project aims to test the feasibility and to develop methods to measure the cost and value of Open Source Software (OSS).

- To quantify cost, we focus on 4 major programming languages (i.e., R, Python, Julia, and JavaScript) and collect information on the development activities of all of the individual packages that use these languages. We use cost modeling approaches from software engineering, in particular, the Constructive Cost Estimation Model [1] (COCOMO) to quantify cost per package using the lines of code and contributor information.
- We use reverse dependencies (reuse) of packages to measure the impact. We generate the dependency network to study the connection between the structural features of the network and the cost of the OSS projects.

Data and Methods

DATA COLLECTION STRATEGY

OSS Definition → Registry → Manifest File → Repository → Project Code (Github)

- Open Source Initiative (OSI)-Approved License
- Production Ready Release for Current Ecosystem

- Language-specific package managers (e.g., CRAN, PyPi)
- Basic information about each package is web scraped

- Contains metadata
  - Author, license, dependencies, version, etc.
- Some registries parse this information.

- Information about author, maintainer, license, package status, etc.

- For the latest release of the package
- Top 100 contributors by commits
- Timestamped lines of code (LoC)
- Number and time of commits
- Profile of contributors
- Continuous integration

Data Collection

Language	Package manager	Number of packages	OSI-approved & production ready	Packages on Github
R	CRAN	12,614	11,886	3,396
Python	PyPi	143,047	7,392	3,804
Julia	Pkg.jl	2,040	1,324	1,324
JavaScript	CDNJS	3,367	3,367	3,213

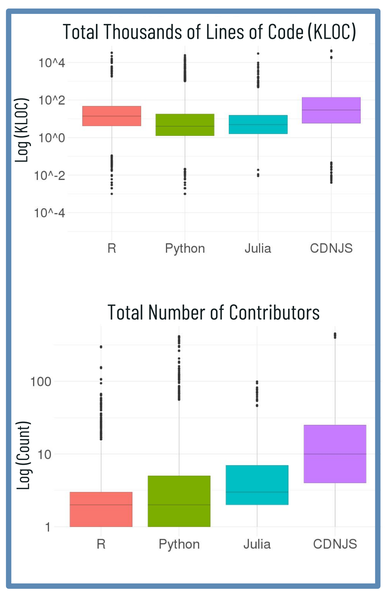
Cost of Open Source Software

The creation cost of packages is estimated using data collected from Github on the *lines of code* added and deleted by each *contributor* to each completed package. These distributions are given on the right.

The COCOMO [1] calibration factor was used to estimate person-months spent per package using kilo (thousands) of lines of code (KLOC). Here, we use the formula (given below) for the *organic software class* which consists of software dealing with a well-known programming language and a small, but experienced team of contributors.

$$Effort = 2.4(KLOC)^{1.05} \quad Person\text{-}months$$
$$Nominal\ development\ time = 2.5(Effort)^{0.38} \quad Months$$
$$Development\ cost = Monthly\ wage \times Nominal\ development\ time$$

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 data [2]) plus additional *intermediate input and capital services costs* (from Bureau of Economic Analysis (BEA) 2007 data [3]). The per person month cost for OSS contribution is obtained as $18,096, which is the amount used in our cost calculations.



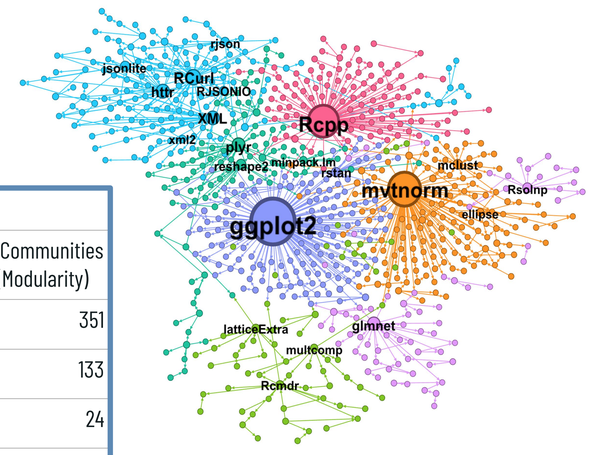
Value of Open Source Software

We use reverse dependencies of packages as the measure of impact/value. We obtain the dependency information from manifest files to generate the dependency network where a directed edge $i \longrightarrow j$ indicates that the package j requires i to be installed to function. Packages with no edges (i.e., dependency links) are removed from the network.

Dependency Network Characteristics

Language	#Nodes	#Edges	Average degree	Diameter	# Connected Components	Largest Component Size	# Communities (Modularity)
R*	6,684	22,024	3.29	7	334	6,303 (94.3%)	351
Python	2,419	3,020	1.25	4	102	2,113 (87.3%)	133
Julia	1,711	5,440	3.18	7	11	1,691 (98.8%)	24
JavaScript	1,279	2,591	2.02	8	13	1,255 (98.1%)	25

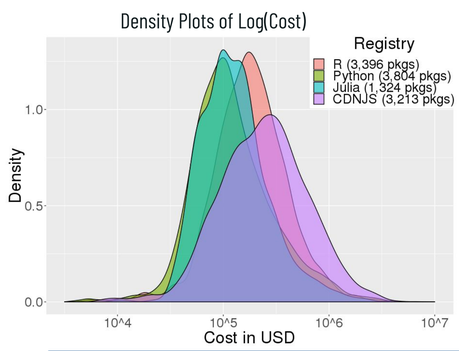
*Includes both dependencies and imports. Standard libraries that are supplied with R (e.g., stats, utils, graphics) are removed.
Average degree (indegree and outdegree) indicates the average number of packages that they depend on (and are used by).
Diameter is the shortest distance between the two most distant nodes in the network.
(Weakly) Connected components are subgraphs such that there is an undirected path from every pair of nodes in the subgraph.
Communities are detected using modularity algorithm [4] that identifies densely connected subgraphs of the network.

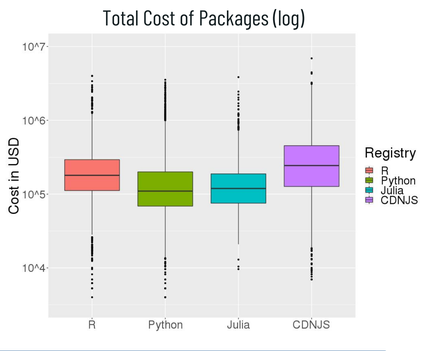


A subgraph of the R dependency network. The size of the node is proportional to the out-degree, i.e., number of packages that reuses the package, and the different colors indicate communities (different uses of R) identified using the modularity algorithm [4] implemented in Gephi [5]. The small communities (less than 5%) are removed for illustration.

Results

We estimate the cost of all packages that use R, Python, Julia and JavaScript. We obtain ***power-law distributions*** for all languages, which is also found in bibliometrics and patents. The density plots and the boxplots are given below.





Top 5 packages with the highest total cost (in USD)

CRAN (R)		PyPi (Python)		Julia (Julia)		CDNJS (JavaScript)	
Package	Cost	Package	Cost	Package	Cost	Package	Cost
googleAnalyticsR	4.0M	Nupic	3.5M	GeoStatsImages	3.8M	Webkit.js	6.9M
Archivist	4.0M	Django-workon	3.3M	PSPlot	2.4M	Phaser-ce	4.5M
Quanteda	3.4M	D1_python	3.1M	MITоS	2.2M	Phaser	4.3M
CollessLike	3.0M	Selenium	3.0M	PDSampler	2.0M	Ag-grid	3.2M
Readtext	2.8M	Senaite.core	3.0M	GeoIP	1.9M	Libsodium.js	3.1M
TOTAL R	854M	TOTAL Python	747M	TOTAL JULIA	239M	TOTAL CDNJS	1,199M

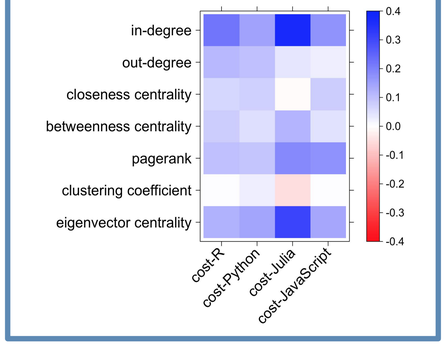
Selected network features and costs of top packages with the highest out-degree

	Package	Outdegree	Indegree	Closeness	Between-ness	Eigen-centrality	Cost ($)
R	ggplot2	925	7	0.73	19.5K	0.07	984K
	Rcpp	838	0	0.50	0	0	871K
	dplyr	626	10	0.76	6.5K	0.06	793K
Python	requests	735	0	0.92	0	0	583K
	setuptools	182	0	0.71	0	0	503K
	scipy	131	0	0.96	0	0	2.34M
	Django	103	0	0.82	0	0	1.85M
Julia	Compat	596	0	0.60	0	0	213K
	Distributions	147	7	0.76	1.7K	0.07	480K
	StatsBase	136	4	0.56	856	0.02	277K
CDNJS	mocha	468	0	0.62	0	0	629K
	gulp	438	2	0.93	801	0.02	214K
	chai	258	0	0.86	0	0	573K

Definitions: Outdegree (indegree, respectively) is the total number of outgoing (incoming) links, i.e., the number of packages that depend on (are required by) the package. Closeness centrality measures how close a node is to every other node. Betweenness is a measure of being connected to other nodes that are not connected to each other (as a bridge). PageRank depends on (i) the number of links the node receives, (ii) the number of links given out by its neighbors, (iii) the centrality of its neighbors. Clustering coefficient quantifies the degree to which a node's neighbors are connected. Eigencentrality of a node takes into account the centrality of its neighbors.

- We do not observe a consistent pattern between cost and value.
- The heatmap also shows that the correlation between the log(cost) of packages and network characteristics is low and positive for most of the features. In-degree (the number of used pkgs) has the highest positive correlation with cost for all the languages.

Heatmap of the Pearson correlation between the log(cost) and network characteristics of packages



Next Steps

- Collect author information (e.g., organization, country) to categorize OSS investment by sector and country.
- Analyze the contributor network to identify influential developers.

References

[1] Barry Boehm, Chris Abts, A. Winsor Brown, Sunita Chulani, Bradford K. Clark, Ellis Horowitz, Ray Madachy, Donald J. Reifer, and Bert Steece. *Software Cost Estimation with COCOMO II* (with CD-ROM). Englewood Cliffs, NJ:Prentice-Hall, 2000.
[2] Bureau of Labor Statistics, U.S. Department of Labor, Occupational Employment Statistics, 2017. [www.bls.gov/oes/].
[3] Bureau of Economic Analysis, Input-Output Accounts Data: The Use of Commodities by Industries, Before Redefinitions (Purchasers' Prices), 2007. [https://www.bea.gov/industry/io_annual.htm].
[4] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, Etienne Lefebvre, Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment 2008 (10), P1000
[5] Bastian, M., Heymann, S., & Jacomy, M. (2009). Gephi: An open source software for exploring and manipulating networks. *ICWSM* 8(2009), 361-362.

BIOCOMPLEXITY INSTITUTE

bi.vt.edu/sdal

Stay Connected: facebook.com/biocomplexity
twitter.com/biocomplexityvt
youtube.com/c/biocomplexity
google.com/+biocomplexity

VIRGINIA TECH