An Empirical Study of Popularity and Quality in NPM Packages

Ahmed Zerouali
Email: ahmed.zerouali@umons.ac.be
University of Mons
Bitergia

***Abstract*—Software systems, and open source software in particular, often leverage on libraries to reuse functionalities. Such libraries are readily available through software package management systems like *npm* for *JavaScript*. However, developers often struggle to identify the most appropriate library that fits their needs, due to huge amount of available packages, often with similar functionalities. In this paper, we empirically study the relationship between quality and popularity in a large dataset of 308k *npm* packages whose metadata was extracted from two open source datasets: *libraries.io* and *npms.io*. We found only a weak relation between popularity and quality. We also found the package popularity in terms of community interest to be moderately correlated with the package's usage by other repositories. We observed that maintenance effort (i.e., commit and release frequency and, opening and fixing issues) has little impact on package usage popularity.**

I. Introduction

In software systems, and open source software systems in particular, reusing code provided by external libraries is widely accepted as common practice by software developers. It allows them to reuse important and often complex functionality that these libraries offer, rather than needing to implement it from scratch.

In order to make external libraries available and easy to use, most of the programming languages come with at least one software package manager, such as the Node Package Manager (*npm*) for *JavaScript* packages. These package managers automate the distribution, installation and upgrading of thousands of different software packages.

If similar functionalities are provided by different packages, it is not always easy for developers to select the most appropriate package for their needs. Should one choose the most popular package? Should one prefer packages that have higher code quality, more tests and less unresolved issues? Should one avoid packages with less, or less active, contributors? Should one avoid packages that have too many dependencies?

In this empirical study, we focus on analysing the relation between software popularity and software quality of packages in a package-based software distribution. We will target *JavaScript/Node.Js* packages in *npm* because of its widespread use, and because it is by far the biggest package manager in terms of number of hosted packages (over 500k packages as of October 2017). Because of its size, it becomes much more likely to find "similar" packages, justifying the need for concrete guidelines for selecting the most appropriate package.

II. Research Questions

In this paper we focus on the following research questions:

***RQ1*: Is there a relation between package quality control and package popularity?** Given the availability of different packages providing similar functionalities, most developers often struggle to choose the right one with a good quality and they find themselves influenced by popular choices [2]. With this question we aim to verify if there is a quantitive evidence of a relation between quality and popularity.

***RQ2*: Is there a relation between the maintainability and popularity of packages?** Software maintenance is an integral part of a software life cycle because it eases the understanding and enhancement of the software. With this research question we aim to study the possible relations between maintenance effort and popularity.

III. Method

In this section, we motivate the selection of the packaging system and present the data extraction process.

A. Choice of Package Manager

In order to study popularity and code quality in software packages, it is important to choose a relevant software package manager. For the purpose of our study, we would like to have a popular ecosystem [3] that involves a big and very active developer community, and a large number of software packages that is increasing over time. The most obvious choice is to focus on a package manager for a popular programming language.

JavaScript is one of the most used programming languages and the most popular one on *GitHub*[1], the world's leading software development platform [1]. Thanks to a very active community and to popular frameworks that can be used to create software applications, such as *AngularJS* and *ReactJS* in the client-side and *Node.Js* in server-side, *JavaScript* developers can either write projects that can be used as regular software applications, or software packages that are intended to be reused (through explicit dependencies) by other projects.

The *npm* (Node Package Manager) has grown exponentially, and is now the largest package registry in the world. It is the

[1]For more information on the popularity of programming languages in GitHub, see http://githut.info and https://langpop.corger.nl

official package manager for *Node.Js* and it contains more than 523,000 packages[2]. Thus, we believe that *npm* packages are good candidates to focus on. Figure 1 shows the evolution of the number of newly created software packages that are still in *npm*, and the number of their releases. The packages and releases available in *npm* at the time of the data extraction were 516K and more than 3,5M respectively.

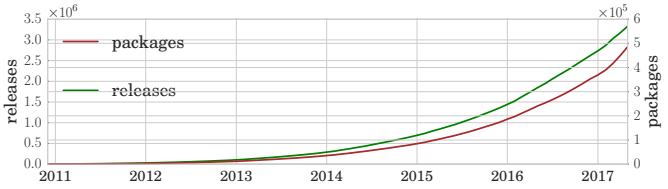

Fig. 1. Evolution of the cumulative number of newly created software packages (right y-axis) and package releases (left y-axis) in *npm*

B. Data Selection and Extraction

To extract data about *npm* packages, in particular with respect to package quality and popularity, we combined information from two different data sources.

libraries.io[3] is an open source repository containing metadata of package dependencies extracted from 23 package managers. The dataset is available as open access under the CC Share-Alike 4.0 license[4]. Based on the dataset of 15 June 2017 we extracted the following information for each *npm* package: the package name and size, the number of packages depending on it, the number of applications/repositories other than packages using this package, and the status, which is the attribute that describes if a packages is deprecated or not. Data from this source was basically about the age, size, status and popularity in terms of usage.

npms.io[5] is an online open source search engine for *npm* packages. For each package, the engine computes a score (i.e., a percentage) for the package's popularity, quality and maintenance.[6] To calculate these scores, many metrics are taken into consideration; these are summarised in Table I. Based on the names of all *npm* packages extracted from *libraries.io*, we searched in *npms.io* the recent available information concerning these scores, as well as the number of open issues for each package.

Table II presents a summary about the metrics extracted from *libraries.io* and *npms.io*. After combining the data from both sources, from the 516,705 packages on *libraries.io*, we found 308,777 of them also on *npms.io*. We observed that all packages on *npms.io* are hosted on *GitHub*, which is of great value for our research since our purpose is to analyse packages that evolve in the same technical environment.

Table III presents a summary about the considered ecosystem and the used dataset.

[2]http://www.modulecounts.com
[3]www.libraries.io
[4]https://zenodo.org/record/808273
[5]www.npms.io
[6]See https://npms.io/about

TABLE I
SUMMARY OF THE METRICS USED BY *npms.io* TO CALCULATE SCORES

Score	Characteristic	Metric
Popularity	community Interest	number of stars number of forks number of subscribers number of contributors
	downloads	number of downloads downloads acceleration
	dependencies	number of dependents
Quality	carefulness	has license has readme? linters configured? has .gitignore and friends? changelog
	tests	has tests test coverage % build status
	health	# outdated dependencies # outdated deps with vulnerabilities
	branding	has badges? has custom website?
Maintenance	commits	commit frequency most recent commit
	releases	release frequency
	issues	ratio of open vs total issues time to close issues

TABLE II
SUMMARY OF THE METRICS EXTRACTED FROM *libraries.io* AND *npms.io*.

Source	Metrics
libraries.io	age, keywords, releases, dependent repositories, dependent packages dependencies, size, status
npms.io	popularity, quality , maintenance

TABLE III
DESCRIPTIVE STATISTICS OF THE CONSIDERED DATASET

Characteristics	NPM
URL	npmjs.com
Language	JavaScript
Packages from *libraries.io*	516,705
Packages from *libraries.io* in *npms.io* and *GitHub*	308,777
Extraction date	15 June 2017

IV. FINDINGS

A. Is there a relation between package quality control and package popularity?

- Package community interest and its usage by other repositories, are moderately correlated.
- 38% of all packages are not used in any package or other repository.
- The most hot topics in *npm* are related to *react, jquery* and *test*.
- Most packages are not popular but most of them have good quality control.
- There is no correlation between package branding quality and popularity.
- *npm* developers care more about the basics of a package, such as the README, license, stability, then tests.
- Software quality and popularity, are weakly correlated.

B. Is there a relation between the maintainability and popularity of packages?

- 48% of packages have zero or disabled issues in their repository.
- Maintenance quality has less impact on the package usage popularity.
- Packages that have frequent commits have also frequent releases.
- Package with zero or disabled issues have less number of releases and their size is smaller than the size of the other packages.

V. Threats to Validity

We used *libraries.io* and *npms.io* to measure various quantitative metrics related to quality, popularity and maintenance. Our measurements are only as accurate as these two tools, the motivation and full derivation of their metrics is beyond the scope of this paper. However, given that we checked the reliability of data manually for few packages and that they both continuously analyze the *npm* ecosystem, we feel confident about their metrics and measurements.

Our results may not be generalizable to other *JavaScript* packages, or other packages that are published in other software package managers.

The particular metrics that we have chosen for carrying out the study may bias the results. Our results could be different when considering and relying on different metrics that have been measured in a different way for quantifying quality or popularity.

VI. Discussion and Future Work

We quantitatively analyzed the quality of development and popularity in terms of usage and community interest of a large number of *npm* packages, using metrics provided by two open source data platforms *libraries.io* and *npms.io*. We found that most *npm* packages are not popular, however, they have good quality control. This latter metric is weakly correlated with popularity.

We also analyzed the relationship between maintenance and popularity and we found that maintenance quality has less impact on the package usage. Packages that have frequent commits have also frequent releases.

As future work, we would like to extend our study to other ecosystems, such as *PyPI*, to compare and better understand the relation between popularity and quality across ecosystems.

Acknowledgment

This research is part of SENECA[7] and SECOHealth project.

References

[1] H. Borges, M. T. Valente, A. Hora, and J. Coelho. On the popularity of github applications: A preliminary note. *arXiv preprint arXiv:1507.00604*, 2015.

[2] M. J. Lee, B. Ferwerda, J. Choi, J. Hahn, J. Y. Moon, and J. Kim. Github developers use rockstars to overcome overflow of news. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 133–138. ACM, 2013.

[3] D. G. Messerschmitt, C. Szyperski, et al. Software ecosystem: understanding an indispensable technology and industry. *MIT Press Books*, 1, 2005.

[7] http://senecaproject.github.io