Understanding Popularity Growth of Packages in JavaScript Package Ecosystem

Shi Qiu*, Raula Gaikovina Kula‡ and Katsuro Inoue*

*Graduate School of Information Science and Technology, Osaka University, Japan
Email: {qiujitsu, inoue}@ist.osaka-u.ac.jp
‡Graduate School of Information Science, Nara Institute of Science and Technology, Japan
Email: raula-k@is.naist.jp

***Abstract*—In open-source software (OSS) ecosystems, software popularity is valuable information to developers because they continually want to know whether their software is attracting and gaining acceptance. Meanwhile, software popularity is also an important indicator to suggest if a software is beating its competitors in an OSS ecosystem. Accordingly, it is important to understand the popularity growth of packages (i.e., how fast packages become popular). In this paper, we conduct an exploratory study on packages in the node package manager (npm) to understand: (1) the characteristics of popularity growth, and (2) the factors that could affect popularity growth. We propose a method to model popularity growth as a curve and find that popularity growth mathematically follows three models — accelerated growth model (i.e., quadratic model), steady growth model (i.e., linear model), and decelerated growth model (i.e., square root model). The results show that 51.56% of the studied packages depict steady growth model, followed by accelerated growth model and decelerated growth model, 40.02% and 7.20% respectively. Furthermore, we reveal that factors including age, dependents, new features and functionalities have impacts on popularity growth. Our study shows potential tips for helping practitioners on developing and evolving packages in a competitive OSS ecosystem.**

I. Introduction

A software ecosystem consists of software projects that are developed and evolve together in a shared environment [13]. Open-source software (OSS) ecosystem is software ecosystem consisting of open-source software (OSS) software projects. Usually, these OSS software projects are contributed by the users of this OSS ecosystem. The development of software ecosystems has resulted in an abundance of free software packages that are easily reused by both new and existing projects. Also, this kind of software reuse has long been proved to be a good method to increase software productivity [15], [16], [4]. One example is `npm`, which serves as a large repository of JavaScript-based software packages. It hosts over 650,000 JavaScript packages to become the largest software ecosystem, with millions of packages being installed from the `npm` repository on an everyday basis. In such a large number of packages, some packages are significantly more attractive than others, being downloaded more times by end users. Software popularity is used to measure this difference. It is a useful indicator of whether a package is successful and is attracting and gaining acceptance in the software ecosystem [5]. Understanding the popularity growth of packages in OSS ecosystem is very important because developers are continually wanting to know whether or not their software is attracting and gaining acceptance. Especially, the competition of packages in OSS ecosystem is becoming more and more fiercely nowadays. For example, it's reported by the homepage of `npm`[1] that about 200,000 new packages were uploaded onto `npm` during the last year, which account for nearly 30% of all the packages uploaded in the past 8 years. It's no doubt that the large and rapidly growing number of packages makes OSS ecosystem more competitive. Therefore, understanding the popularity growth will also help developers on improving their packages to survive in the competitive OSS ecosystem.

Studies of popularity are firstly conducted on the social platforms such as YouTube [1] and Twitter [12] aimed for recommendations on how to produce successful content. Unfortunately, to date, there have been few studies on software popularity growth. One exception is an effort on understanding the evolution of popularity by case study [17], but they only examine the top 5 popular packages in `npm`. Borges et al. aim to investigate common patterns of popularity growth of the projects on `GitHub` [5]. They use KSC algorithm [11] to achieve this, but due to the restriction of the algorithm, this approach can hardly be applied to a large and irregular dataset.

In this paper, we would like to understand how fast packages become popular (defined as `popularity growth`). We propose that popularity growth over time can be modeled as a mathematical equation, and is plotted visually as a growth curve. Actually, this kind of application of the mathematical models is very common in many fields of biology, medicine, economics and the social sciences [3]. Especially, growth curve models have been used in various disciplines as well, for example in biological sciences to study crop growth, population processes, and bacterial growth. They are often estimated to understand defining characteristics, including initial levels, rates of change, periods of acceleration and deceleration, and final or asymptotic levels [8]. In software engineering, different models have been applied in the context of software reliability (e.g., [2], [18]) and modeling the evolution of library usage [10].

For an empirical evaluation of our popularity growth mod-

[1]https://www.npmjs.com/

TABLE I
MATHEMATICAL MODELS

Model	Equation
accelerated growth model	$Popularity = at^2 (a > 0)$
steady growth model	$Popularity = at (a > 0)$
decelerated growth model	$Popularity = a\sqrt{t} (a > 0)$

TABLE II
SUMMARY STATISTICS OF THE COLLECTED DATASET

	Dataset statistics
observation period	2010-Oct to 2017-Apr
# packages	152,812
total size of projects	365 GB

els, we conducted an exploratory study of packages in npm to understand: (1) the characteristics of popularity growth, and (2) the factors that could affect popularity growth. This paper aims at answering two research questions:

(RQ1) Do packages in npm share common characteristics of popularity growth? If so, what are these characteristics? The goal is to find different kinds of characteristics of popularity growth and the proportion of them. The answer to this question will provide a general view of how fast the popularity grows for packages in npm.

(RQ2) Are there some factors which could affect the popularity growth of packages in npm? If so, what are the effects of these factors? This investigation can reveal the factors that could be utilized to accelerate popularity growth or to prevent the deceleration of it.

To answer RQ1, we first proposed a method of modeling popularity growth as a curve. To answer RQ2, we select some main factors including total downloads count, age, the number of contributors, dependencies, dependents, versions, and functionalities. We reveal the impact of these selected factors by examining their significant difference across proposed models. We also give some suggestions based on the results we observed.

II. MODELING POPULARITY GROWTH AS A CURVE

In this paper, we are interested in the popularity growth of the packages in npm. Kula et al. have succeeded in modeling the evolution of library usage as a curve to study the library aging [10]. Inspired by this work, we proposed a method of modeling popularity growth as a curve. We assume that popularity growth has the following three types: (1) grow slowly at first but accelerate over time, (2) grow steadily, with no sign of accelerating or decelerating, (3) grow rapidly at first but gradually decelerate.

We define these three kinds of growth as the accelerated growth model, steady growth model and decelerated growth model. When popularity growth is modeled as a curve, the accelerated growth can be represented by a convex while the decelerated growth can be represented by a concave. The convex on curve indicates the speed of growth is accelerating over time while the concave indicates the speed is decelerating. Similarly, the steady growth can be represented by a straight line, indicating that the speed is steady over time. Furthermore, we use three mathematical equations to represent the characteristics of the proposed three growth models. The linear growth model is the most commonly fit growth curve to describe a steady growth. For nonlinear change, many researchers turn to the quadratic growth model when a linear change model does not fit well or when a nonlinear trend is seen in the longitudinal plot [8]. When we limit the coefficient to be positive, the quadratic growth model becomes a good choice to describe an accelerated growth. Another reason we select quadratic equation is that its rate of growth increases slowly and gently. So if popularity growth has a tendency to accelerate, even not dramatically, it will still be well fitted by this equation. For the same reason, we select square root equation as the opposite equation to describe the decelerated growth.

To summarize, we use three models as shown in Table I and Figure 1 for our curve fitting. Thus, for the relationship between the popularity and time *t*, key characteristics of each model are described below:

- **Accelerated growth model**. The quadratic equation is depicted in Figure 1(a) as having a convex toward the lower right corner. The curve of this model indicates that popularity grows slowly at first but accelerates over time.
- **Steady growth model**. The linear equation is depicted in Figure 1(b) as having the single linear line and no convex and concave. It indicates that the popularity grows steadily, with no sign of accelerating or decelerating.
- **Decelerated growth model**. The square root equation is depicted in Figure 1(c) as having a concave toward the lower right corner. The growth of popularity fitting this model grows rapidly at first but gradually decelerates.

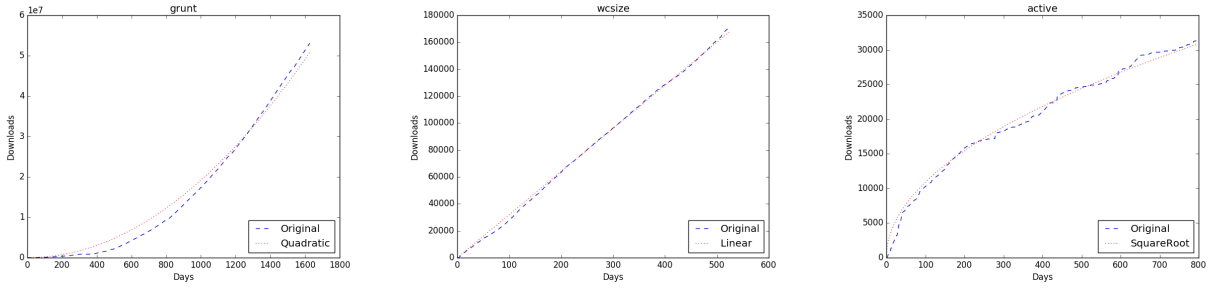
For the metric to measure popularity, we choose downloads count. Downloads count is incremented every time a package is installed from npm in any cases — redistribution, test or development. We use downloads count because it is a value to show how many times the package is downloaded, indicating how popular the package is used by end users intuitively. Note that, as shown in Figure 2, when a package is installed from npm, those packages which are in the dependency chain of this package are also installed. In this case, downloads count is incremented for all these packages. We obtained the historical data of downloads counts for each package on a daily basis through the web API of npm[2]. Finally, we accumulate the downloads counts to represent popularity growth.

III. EMPIRICAL EVALUATION

A. *(RQ1)* Do packages in npm share common characteristics of popularity growth? If so, what are these characteristics?

1) Research Method: Our research method comprises of two steps. In the first step, we need to collect empirical data

[2]source: https://api.npmjs.org/downloads/range/2010-10-1:2017-04-07/packageName



(a) accelerated growth model (b) steady growth model (c) decelerated growth model

Fig. 1. The popularity growth for three packages representing the three proposed models. In each plot, the blue curve is the one created with the original data while the red curve is the model that fits best. Note that Figure 1(a) represents the accelerated growth model (`grunt`), Figure 1(b) represents the steady growth model (`wcsize`) and Figure 1(c) illustrates the best-fits for decelerated growth model (`active`).

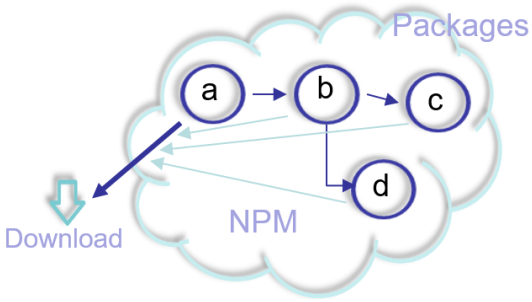


Fig. 2. The effect of dependencies on downloads in `npm`. When package a is installed, package b, c and d is downloaded at the same time.

TABLE III
BEST FITTING RESULTS FOR THE 102,341 TARGET PACKAGES.

Model	**# Fitted**	**% of Studied Packages**
accelerated growth model	40,953	40.02%
steady growth model	52,769	51.56%
decelerated growth model	7366	7.20%
Not Fit	1,253	1.22%
Total	102,341	100%

that represents popularity and other main software ecosystem factors for `npm`. Table II shows a summary of collected `npm` packages. The observation period is from October 1st, 2010 to April 7th, 2017, and all data we collected only cover this range as well. At last, we collect 152,812 packages. Then we filter out packages whose age is younger than 1 year or total downloads count is less than 1,000. This step ensures the reliability of the following curve fitting because the popularity growth of packages with few downloads or short lifetime is meaningless. For example, a package was published two days ago and downloaded once for each day. In this extreme case, this package will still be fitted by the steady growth model, which is not expected. We end up with 102,341 packages in total. Then for the second step, we run the experiment by which popularity growth is fitting against the three proposed growth models. Firstly, for the curve fitting, we rely on a Python-based package called `scipy`[3] to fit popularity growth against the proposed three growth models. Secondly, We use the widely-used the coefficient of determination [14], denoted by R^2, to evaluate the goodness of fit for each growth model. The one with the largest R^2 value among three proposed models is determined as the best-fitted model. Additionally, no matter which model is determined as the best-fitted model, the R^2 value of it must be larger than 0.7. The value is decided as 0.7 because if the R^2 value is larger than 0.7, this value is generally considered strong effect size [7]. For the case that all R^2 values of three proposed models are less than 0.7, the package is not well fitted by any models. By this, we ensure that the best-fitted model fits the popularity growth curve enough well. Meanwhile, we can also examine whether our three proposed models are effective or not. If a large number of packages could not be well fitted by any proposed models, our three proposed models are obviously not good enough to model popularity growth.

2) Findings: Table III lists the percentage of packages that are best-fit by each model. Notice that the percentage of not fit is only 1.22%. It suggests that only a small part of all packages could not be fitted by any proposed growth model. The percentage proves that our method is effective in modeling the popularity growth of packages. The result shows that 51.56% of the studied packages depict steady growth model, followed by accelerated growth model and decelerated growth model, 40.02% and 7.20% respectively. The most important finding is that only 7.20% of the studied packages are best-fitted by decelerated growth model, which indicates that `npm` is still very active and the number of packages installed from `npm` is still growing with no sign of deceleration. Specifically, 40.02% of the studied packages are best-fitted by accelerated growth model, which interprets that a large number of packages in `npm` are gaining popularity in accelerating speed. The result suggests that the reuse of packages in `npm` is still active, with more and more packages being installed from the `npm`.

Hence, we answer RQ1:

[3]https://www.scipy.org/

Packages in npm do share common characteristics of popularity growth. Specifically, 51.56% of the studied packages depict steady growth model, followed by accelerated growth model (40.02%) and decelerated growth model (7.20%). The distribution suggests that the reuse of packages in npm is still active.

B. *(RQ2)* Are there some factors which could affect the popularity growth of packages in npm? If so, what are the effects of these factors?

The result from RQ1 indicates that most studied packages depict steady growth model and accelerated growth model while only a few packages depict decelerated growth model. Hence our motivation for RQ2 is to make use of a quantitative approach to examine the effect of some main software ecosystem factors on popularity growth.

1) Research Method: To solve this research question, we select some main software ecosystem factors and examine whether or not these factors are significantly different among the packages fitted by three proposed growth models. By this way, we aim to find the impact of these selected factors on popularity growth. The main software ecosystem factors we selected include *total downloads count, age* and *the number of contributors, dependencies, dependents, versions*.

The study on total downloads count could interpret if the packages with accelerated popularity growth are also the popular ones. Note that the downloads counts are accumulated on a daily basis. So the downloads count of the last day is the total downloads count. The investigation on age answers the question that whether or not the early packages are more likely to gain popularity in accelerated speed than the new ones. The age is represented by the number of passing through days after the first publication of a package. The number of contributors is the indicator of the scale of the development team. The contributors refer to any developer who ever contributed to any version of a package. A package in npm is usually maintained and contributed by an individual developer or some developers working as a team. While the numbers of dependencies and dependents reveal that if reusing or being reused by other packages have effects on popularity growth. The study on the numbers of dependencies and dependents will suggest the impact of relationships among packages on popularity growth. At last, the study on the number of versions interprets the impact of new features on popularity growth. Additionally, the larger number of versions not only means frequent update and more new feature but also suggests whether or not the package is continuously maintained.

For these selected factors, we make use of statistic analysis to examine whether or not there is a strong relationship between them and the proposed growth models. Specifically, we randomly pick 1,000 packages from each growth models and run Kruskal-Wallis H test [9] on them to re-check whether or not the selected factors are significantly different across three proposed growth models. Kruskal-Wallis H test is an effective and widely used method to test whether two or more samples of equal or different sample sizes originate from the same distribution [6]. Additionally, We investigate if the functionalities of packages play roles in popularity growth as well. For this purpose, we collect the keywords of every package fitted by each growth models and draw the word-cloud graph using these keywords. The keywords are extracted from the meta-files of packages and we believe that they interpret the functionalities of packages to some extent.

2) Findings: Table IV shows the summary of the statistic analysis result on *total downloads count, age* and *the number of contributors, dependencies, dependents, versions* for each growth model.

Total download count: The result shows that the packages with accelerated popularity growth are also the popular ones, which also conforms to our general impression.

Age: The packages fitted by accelerated growth model are definitely older than other two models, and the ones fitted by steady growth model are also a bit older than decelerated model. It suggests that the early packages in npm are much more easily to get popularity in accelerated speed with the time passing by while the new packages are not.

The number of contributors: The number of contributors interprets the scale of development team. The result suggests that there is no significant difference across packages fitted by the proposed three growth models. It interprets that the scale of development team have no definite impact on popularity growth.

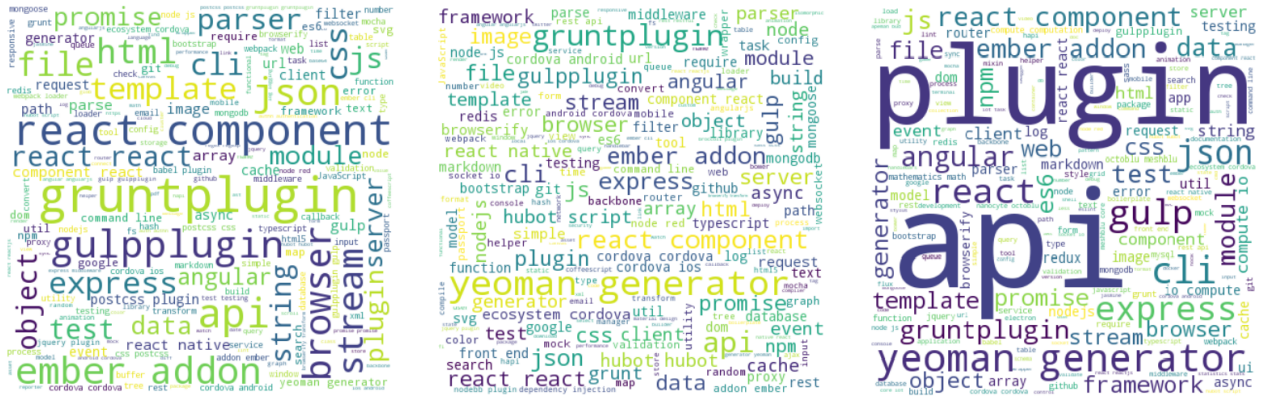
The number of the dependencies and dependents: In this investigation we want to answer the question whether reusing or being reused by other packages have effects on popularity growth or not. The result suggests that the numbers of the dependencies are similar across the three proposed growth models while the numbers of the dependents are significantly different. The packages fitted by accelerated growth model attract a lot of dependents while steady growth model and decelerated growth model attract few. It illustrates that being reused by other packages plays a significant role in accelerating popularity growth while the number of dependencies does not. This can be explained by the download mechanism of npm — when a package is installed from npm, those packages which are in the dependency chain of this package are also installed.

Versions: The p-value tells that the numbers of versions are significantly different across the three proposed growth models. The result suggests that packages fitted by accelerated growth model and decelerated growth model have a tendency to maintain more releases and add new features. On the contrary, the releases of packages fitted by the decelerated growth model are less. This illustrates that adding new features is a double-edged sword to popularity growth: both of acceleration and deceleration are possible due to adding new features. On the contrary, less changes make popularity grow steadily.

Functionalities: Figure 3 shows the the word-cloud graphs created with the keywords of every package fitted by three proposed growth models. The result shows that the functionalities have a large variation across the three proposed growth models. The top 3 keywords of packages in accel-

TABLE IV
SUMMARY STATISTICS OF THE 101,088 FITTED PACKAGES.

	Fitting Model	Min.	1st Qu.	$\mu = median$	3rd Qu.	Max.	$\bar{x} = mean$	p-value
# Total downloads	accelerated growth model	1001.0	3091.0	8312.0	48718.0	800785605.0	2505140.14	
	steady growth model	1001.0	1511.0	2430.0	4876.0	92901071.0	49305.13	1.22e-09
	decelerated growth model	1001.0	1619.0	2820.0	6039.75	15443288.0	24208.82	
Age (# days)	accelerated growth model	366.0	759.0	1179.0	1507.0	17263.0	1169.85	
	steady growth model	366.0	585.0	781.0	1017.0	2300.0	817.89	1.12e-54
	decelerated growth model	366.0	423.0	533.0	796.0	2291.0	640.79	
# Contributors	accelerated growth model	0.0	1.0	1.0	1.0	134.0	1.53	
	steady growth model	0.0	1.0	1.0	1.0	64.0	1.27	0.63
	decelerated growth model	0.0	1.0	1.0	1.0	55.0	1.29	
# dependencies	accelerated growth model	0.0	1.0	2.0	4.0	106.0	3.79	
	steady growth model	0.0	1.0	3.0	5.0	114.0	4.24	0.24
	decelerated growth model	0.0	2.0	3.0	6.0	122.0	4.81	
# dependents	accelerated growth model	0.0	1.0	3.0	10.0	32130.0	38.13	
	steady growth model	0.0	1.0	1.0	2.0	3456.0	3.31	9.38e-49
	decelerated growth model	0.0	1.0	1.0	2.0	61.0	2.29	
# versions	accelerated growth model	0.0	3.0	6.0	13.0	4204.0	12.95	
	steady growth model	0.0	2.0	3.0	7.0	742.0	6.55	0.005
	decelerated growth model	0.0	4.0	7.0	14.0	1813.0	12.58	



(a) accelerated growth model (b) steady growth model (c) decelerated growth model

Fig. 3. The word-cloud graph using the keywords of every package fitted by each growth models.

erated growth model are *gruntplugin, gulpplugin* and *react-component*, which are all related with one particular popular package. Among them, *grunt*[4] package is a JavaScript task runner, which can free developer from repetitive tasks like minification, compilation, unit testing, linting, etc. *gulp*[5] package is a toolkit that helps developer automate painful or time-consuming tasks in the development workflow. *react*[6] package is a JavaScript package for building user interfaces, a traditional client-side application on web development. It illustrates that *grunt*, *gulp*, and *react* are probably most widely used packages in `npm`, so the packages related to them can easily gain popularity in accelerated speed. Additionally, the packages fitted by decelerated model also get two significant hot keywords — *plugin* and *api*. This might suggest that some packages of this kind suffer from higher risk of losing popularity as time passes by.

Hence, we answer RQ2:

[4]https://gruntjs.com/
[5]http://gulpjs.com/
[6]https://github.com/facebook/react

Some main factors including age, dependents, new features, and functionalities have effects on popularity growth. Among them age and dependents have a positive impact on popularity growth while the new features have possibility of both acceleration and deceleration. We also reveal some possible effects of functionalities on popularity growth.

IV. DISCUSSION

In this section, we aim to give some suggestions to practitioners in `npm` community to help them on developing and evolving their packages.

- `npm` community is health and well maintained. Therefore, practitioners in such community could possibly get chances to publish successful packages gaining a lot of acceptance.
- The popularity also complies with the Matthew Effect - the popular and rapid growing packages gain more and more acceptance easily. It's not wise for developers to have their packages compete with these packages.

- The early packages have strong advantage. The packages tested by time are proved to be of good quality and much more easily to be accepted. To compete with these early packages, developers should aim to introduce their packages with the distinctive features and good reliability to persuade other packages to make a change.
- Being widely reused by other packages will make a package popular. Easy-to-use features and well-maintained documents will help on that. Especially, being merged into the dependency chain of a popular and rapid growing package is a convenient way to accelerate the popularity growth of a package. So developers can find some popular and rapid growing packages which could possibly reuse their packages and provide more reliable and effective functionalities to them.
- Adding new features is a double-edged sword. On one hand, frequent changes of versions usually refer to the package is well maintained. On the other hand, the new features will also possibly result in dependency chain breakages and confuse the re-users. Developers should pay attention to the balance between them.
- Packages related to some popular packages also easily become popular in `npm`. Developing packages related to popular packages may be a good choice for practitioners in `npm` community.

V. Challenges and Future Work

In this paper, we model the popularity growth of packages as curves and fit it against the three proposed growth models to understand the common characteristics of the popularity growth of packages in `npm`. We found that 51.56% of the studied packages depict the steady growth model, followed by accelerated growth model (40.02%) and decelerated growth model (7.20%). The result suggests that the reuse of packages in `npm` is still active, with more and more packages being installed from the `npm`. Also, we select and examine some main software ecosystem factors to understand their impacts on popularity growth. Our study shows that age, dependents, new features, and functionalities play significant roles in popularity growth. Based on these findings, we give some suggestions to practitioners in `npm` community. We hope our results provide valuable insights to help practitioners in developing and evolving packages in a competitive OSS ecosystem.

For the future work, although our approach of modeling the popularity growth of packages is proved to be effective because of the low percentage of packages not fitted by any growth models, we still don't know why these packages could not be fitted. By observing several packages, we find that the popularity of some packages not fitted is changing dramatically in a short time. It will be very interesting to find out why it happens. Secondly, although we have found the common characteristics of popularity growth and some software ecosystem factors having an impact on the growth, the detailed mechanism of how these factors impact popularity growth is still not very clear. Additionally, we presume that there are other factors that can have an impact on the popularity growth of packages since our selected factors are only a portion of all the factors of a software ecosystem. We consider these as future work. Our future work also includes extending the research to different ecosystems such as `RubyGems` or `CRAN` R ecosystem. Also, it would be interesting to do predictions on popularity growth based on the factors we observed.

VI. Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 18H04094.

References

[1] M. Ahmed, S. Spagna, F. Huici, and S. Niccolini. A peek into the future: Predicting the evolution of popularity in user generated content. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 607–616, New York, NY, USA, 2013. ACM.

[2] V. Almering, M. van Genuchten, G. Cloudt, and P. Sonnemans. Using software reliability growth models in practice. *Software, IEEE*, 24(6):82–88, Nov 2007.

[3] G. Annadurai, S. Rajesh Babu, and V. R. Srinivasamoorthy. Development of mathematical models (logistic, gompertz and richards models) describing the growth pattern of pseudomonas putida (nicm 2174). *Bioprocess Engineering*, 23(6):607–612, 2000.

[4] B. W. Boehm. Improving software productivity. *Computer*, 20(9):43–57, Sept. 1987.

[5] H. Borges, A. C. Hora, and M. T. Valente. Understanding the factors that impact the popularity of github repositories. *CoRR*, abs/1606.04984, 2016.

[6] W. Daniel. *Applied nonparametric statistics*. Houghton Mifflin, 1978.

[7] S. M. David, I. N. William, and A. F. Michael. *The Basic Practice of Statistics*, page 138. W. H. Freeman, 2013.

[8] K. J. Grimm, R. Nilam, and H. Fumiaki. Nonlinear growth curves in developmental research. *Child Dev.*, 82:1357–1371, Sep 2011.

[9] W. H. Kruskal and W. A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

[10] R. G. Kula, D. M. German, T. Ishio, A. Ouni, and K. Inoue. An exploratory study on library aging by monitoring client usage in a software ecosystem. In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 407–411, Feb 2017.

[11] R. Langone, R. Mall, C. Alzate, and J. A. K. Suykens. *Kernel Spectral Clustering and Applications*, pages 135–161. Springer International Publishing, Cham, 2016.

[12] J. Lehmann, B. Gonçalves, J. J. Ramasco, and C. Cattuto. Dynamical classes of collective attention in twitter. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 251–260, New York, NY, USA, 2012. ACM.

[13] M. Lungu, M. Lanza, T. Grba, and R. Robbes. The small project observatory: Visualizing software ecosystems. *Science of Computer Programming*, 75(4):264 – 275, 2010. Experimental Software and Toolkits (EST 3): A special issue of the Workshop on Academic Software Development Tools and Techniques (WASDeTT 2008).

[14] L. Magee. R 2 measures based on wald and likelihood ratio joint significance tests. *The American Statistician*, 44(3):250–253, 1990.

[15] M. D. McIlroy, J. Buxton, P. Naur, and B. Randell. Mass-produced software components. In *Proceedings of the 1st International Conference on Software Engineering (ICSE1968)*, pages 88–98, 1968.

[16] T. A. Standish. An essay on software reuse. *IEEE Transactions on Software Engineering*, SE-10(5):494–497, Sept 1984.

[17] E. Wittern, P. Suter, and S. Rajagopalan. A look at the dynamics of the javascript package ecosystem. In *Proceedings of the 13th International Conference on Mining Software Repositories*, MSR '16, pages 351–361, New York, NY, USA, 2016. ACM.

[18] S. Yamada, M. Ohba, and S. Osaki. S-shaped reliability growth modeling for software error detection. *Reliability, IEEE Trans.*, R-32(5):475–484, Dec 1983.