How to Not Get Rich: An Empirical Study of Donations in Op€n $our¢e

Cassandra Overney,[†] Jens Meinicke,[‡] Christian Kästner,[‡] Bogdan Vasilescu[‡]
[†]Olin College, USA [‡]Carnegie Mellon University, USA

ABSTRACT

Open source is ubiquitous and many projects act as critical infrastructure, yet funding and sustaining the whole ecosystem is challenging. While there are many different funding models for open source and concerted efforts through foundations, donation platforms like *PayPal*, *Patreon*, and *OpenCollective* are popular and low-bar platforms to raise funds for open-source development. With a mixed-method study, we investigate the emerging and largely unexplored phenomenon of donations in open source. Specifically, we quantify how commonly open-source projects ask for donations, statistically model characteristics of projects that ask for and receive donations, analyze for what the requested funds are needed and used, and assess whether the received donations achieve the intended outcomes. We find 25,885 projects asking for donations on GitHub, often to support engineering activities; however, we also find no clear evidence that donations influence the activity level of a project. In fact, we find that donations are used in a multitude of ways, raising new research questions about effective funding.

ACM Reference format:
Cassandra Overney, Jens Meinicke, Christian Kästner, Bogdan Vasilescu. 2020. How to Not Get Rich: An Empirical Study of Donations in Open Source. In *Proceedings of 42nd International Conference on Software Engineering, Seoul, Republic of Korea, May 23–29, 2020 (ICSE '20),* 13 pages.
DOI: 10.1145/3377811.3380410

1 INTRODUCTION

Open-source software is ubiquitous, but sustaining it is a challenge. Open source plays critical roles in our software infrastructure, and by extension in economic growth and almost every facet of modern life, far beyond only technical software projects: it is used in almost every product or in the process of creating products by companies big and small, often in ways invisible to open-source maintainers or without ever contributing back. Some argue that open source provides just as important infrastructure as roads and bridges do for the economy, yet its importance, and our dependence on it, are often not recognized [15].

As all software projects [41], open source also needs continuous effort to fix bugs and vulnerabilities and adapting to evolving technical and nontechnical environments and requirements to remain relevant. With increasing popularity, demands for maintenance and support work typically rise, including large numbers of support requests, feature requests, and reported issues. When open-source infrastructure is insufficiently maintained or even abandoned by their developers, this can raise significant costs and risks for users of such infrastructure, who might need to work around known bugs or make significant changes to find alternatives. How to supply all this needed maintenance and development work is an open and sometimes controversial question.

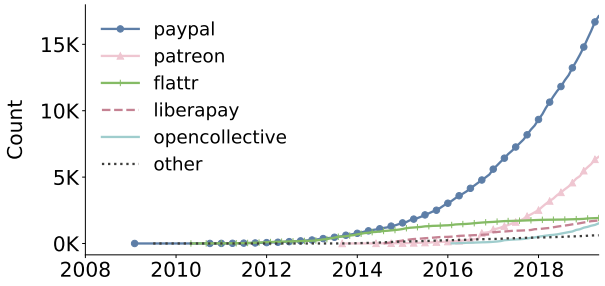


Figure 1: Adoption of donation platforms over time on GitHub (number of new non-fork repositories per month).

Traditionally, much work in free software and open source has been done by volunteers, but over the last decades many corporations got involved in open source, releasing their own open-source projects and paying developers to work on open source. Open source also provides business opportunities to found companies selling premium features, support, or hosting services.

Sustaining open source is another significant challenge that the community has identified and is discussing controversially [18, 32, 43, 45, 63, 65]. There is a patchwork of models to support open source [16], including reliance on often-overworked volunteers (many of which recently have raised concerns about stress and burnout in the community [1, 6, 23, 31, 40, 43, 48, 76]), sponsorship from corporations and foundations, selling premium versions, hosting services and support, raising money through books, consulting or speaking engagements, and simply asking for donations.

In this paper, we focus on the latter, *donation and crowd-funding platforms*, such as *PayPal*, *Patreon* and *OpenCollective*. Donations are gaining popularity in open source as a potential viable model for sustainability, as evidenced by the many community blog posts and podcasts [32, 42, 45, 63, 65, 80], GitHub's recent addition of a *Sponsors* feature to prominently and uniformly support requests for donations [81], and our own findings (see Fig. 1).

Even though donations seem to gain traction to support open-source activities, little is known about their prevalence, success, and impact. Through an exploratory mixed-methods empirical study,

ICSE '20, Seoul, Republic of Korea

978-1-4503-7121-6/20/05…$15.00
DOI: 10.1145/3377811.3380410

we investigate the goals, reality, and use of donations in open source, contributing a deeper understanding of this new phenomenon, and provide data, where there were previously often only opinionated and controversial discussions. In summary, we contribute: (1) a study design to identify open source projects requesting donations and a census of donation requests on GitHub, (2) an analysis of observable characteristics of projects requesting and receiving donations, (3) a list of commonly-stated reasons for asking for donations, (4) a time-series analysis of donations' effects on project activity, and (5) an analysis of how donations are commonly spent.

We find 25,885 projects asking for donations on GitHub, most commonly with *PayPal* and *Patreon*, typically with the goal of supporting engineering activities. Many of these projects receive donations but rarely enough to fund a full-time engineering position. While we do not find strong evidence that received donations associate with higher levels of activity in a project, we find a multitude of different patterns of how received donations are spent. For example, many projects that are successful at fund-raising do not spend their funds. In addition, funds are often spent on non-engineering community activities (*e.g.*, travel), web hosting, and personal expenses. By providing insights into what some open-source projects are currently doing, our results (1) can lead to more realistic expectations regarding donations, (2) may shape future fundraising activities, and (3) raise new questions for future research about sustaining open-source funding.

2 BACKGROUND AND RELATED WORK: OPEN SOURCE AND MONEY

Open-source software has a long and varied history [12, 50, 51, 74], seeing increasing commercialization and professionalization in the last decade. Open source has been recognized to have enabled significant productivity gains [15, 29, 52] and is now broadly adopted as fundamental infrastructure in most software projects [30, 49, 67]. However, practitioners and researchers have recognized sustainability challenges, as a quickly growing amount of open-source infrastructure needs to be maintained [7, 15, 42, 45, 48, 71].

Funding of open-source software has long been an issue, raising questions about sustainability. Open source can be considered as a *public good* or *common-pool resource* [55], which is free to use, but therefore also potentially challenging to produce in a market setting. There are many different approaches to fund the creation and maintenance of open source [16], including selling support, consulting, and services around a project (*e.g.*, the 'RedHat model'), adopting a combined open/closed source strategy (*e.g.*, dual license, 'open core'), companies committing full-time or part-time developers to open source (*i.e.*, 'corporate open source') [4], or just asking for donations. In many cases, companies and open-source projects band together and create foundations to support open-source projects by providing a governance framework, centralizing fundraising, paying for infrastructure, and (sometimes) paying for engineering work [36]. Yet, much of current open-source work seems to be still performed by volunteers: according to GitHub's recent representative survey [27], only 23 percent of the respondents indicated that they contribute to open source as part of their job description.

Researchers have long studied why individual developers and corporations participate in open source. Developers participate both because of intrinsic (*e.g.*, enjoyment or sense of obligation) and extrinsic (*e.g.*, pay, reputation, or own use) reasons [9, 26, 39, 58, 64, 73]. Economic motivations such as signaling one's value in a labor market have also been theorized [44] and empirically supported [13, 46, 47, 58]. Not all developers accept money for open-source work, mirroring their motivations [37]. At the same time, companies often aim to gain synergistic advantages either as the center of an ecosystem [35] or by soliciting third-party contributions and cooperation with others on non-strategic parts of their business [19, 75].

With the increasing popularity of open source and its use as critical infrastructure in many projects, many developers perceive rising demands on their time in terms of a constant stream of feature requests, support questions, and bug reports [e.g., 23, 40, 43, 48, 70]. Developers who volunteer their time for open-source work increasingly report stress and even burnout [e.g., 31, 40, 76], and turnover can be high, threatening the sustainability of critical projects [11, 17, 33, 34, 48, 59, 62, 71, 77, 79]. Some developers thus hope that donations are a path to sustain their own open-source activities.

While donations are often discussed by practitioners [e.g., 32, 63, 68], little is known about their actual prevalence or effectiveness. Krishnamurthy and Tripathi [38] studied donations to SourceForge (that is, donations to the hosting platform, not to the hosted projects) back in 2009, finding among others that developers who were on the platform longer were more likely to donate. Nakasai et al. [53, 54] analyzed donations to the Eclipse foundation, finding that benefits to donors and displaying badges can encourage donations, that donations can lead to preferential treatment when closing issues of donors, and that releases often trigger donations. Of course, there is a vast amount of research on donations, philanthropy, and altruism more broadly, ranging from studies on what influences philanthropic giving (e.g., awareness of need and reputation) [2] to *effective altruism* [66], which could inform many recommendations of how to effectively raise donations for open source; however, at this point our goal is to first understand the current landscape of open-source donations, including their prevalence, goals, and surrounding practices.

3 STUDY OVERVIEW

We adopt an *exploratory mixed-methods research design* [14], in which we incrementally explore publicly available resources on open-source projects and donations. We proceed iteratively, with results from prior phases informing subsequent phases of research (*e.g.*, inspiring additional research questions to explore unexpected results), and we frequently interleave quantitative and qualitative methods to explore phenomena and seek explanations.

Ethical considerations. Given the sensitive topic, where money and donations are often framed in the context of larger discussions on fairness, stress, or even burnout, and the unequal distribution of donations focused on few projects and individuals, we explicitly made a decision to *avoid interviews and surveys with a potentially vulnerable population* that already receive many survey requests from researchers. For example, when developers ask for donations as a "cry for help" because they have too little resources and time to work on their open-source projects (see Sec. 6.2), we would rather not add to their stress by asking for more of their time. Instead, our research analyzes only *public artifacts*, including descriptions

on GitHub repositories and project web pages, public information on donation and crowdsourcing platforms, metadata from GitHub activities and package releases, as well as blog posts and other grey literature [22]. By aggregating data from different sources, our research may make behavior and patterns more transparent (*e.g.*, projects spending little of their donations, see Sec. 8), but we only analyze donation amounts and spending for those projects that chose to use donation platforms that explicitly make such information public in the first place, allowing anybody to perform these kinds of analyses.

Scope. We study projects in two corpora: all 537,640 GitHub repositories corresponding to *npm* packages and all 77,934,441 repositories on GitHub by May 23, 2019. The former is a subset of the latter, but allows us to focus explorations first on a smaller subset, that is known to be innovation friendly [3], with which we are more familiar, which we can download in total, and for which we can gather additional metadata (download counts and dependencies). We typically start our exploration with the *npm* corpus and then attempt to generalize to the GitHub corpus. We do not consider projects that are not on GitHub, but given GitHub's dominance for open-source hosting (even projects hosted on other platforms are often mirrored on GitHub and thus included in our corpus, *e.g.*, the Linux kernel), we expect that the GitHub dataset is characteristic of open-source development more broadly, though projects on GitHub may differ from the global population of open-source projects in ways we do not account for.

Research steps. We report our research in five steps, centered around five themes (see Fig. 2 for an overview):
(1) Frequency of donation requests: Using large-scale repository mining, we start with a census of how many open-source projects ask for donations (RQ 1) and which donation platforms they commonly use (RQ 2). Results lead us to subsequently focus on *Patreon* and *OpenCollective*, two popular and transparent services.
(2) Characteristics of projects asking for and receiving donations: Next we explore, using multiple regression modeling, characteristics of projects that are more likely to ask for donations (RQ 3) and of those that are successful in raising donations (RQ 4).
(3) Expectations for donations: To understand expectations, we analyze reasons developers give for asking for donations (RQ 5), using qualitative text analysis techniques to generate testable hypotheses about the expected effects of donations.
(4) Observable outcomes of donations: To test whether donations associate with observable outcomes such as increase activity, we model the longitudinal effects of donations on a sample of projects that successfully raised donations, using interrupted time series designs (RQ 6).
(5) Use of donations: After finding only weak results regarding donation outcomes, we use qualitative analysis to explore this difference between expectations and outcomes, focusing on how donations were used in those projects (RQ 7).

4 STEP 1: FREQUENCY OF DONATION REQUESTS

As a first step, we ask **RQ 1: How common is asking for donations in open source?** and **RQ 2: Which donation platforms are typically used to collect donations?** We try to get a census of donation requests among all projects in our *npm* and GitHub corpora and identify popular donation platforms for subsequent research. We additionally explore changes over time (when donation platforms are adopted and dropped) and common combinations of donation platforms to understand trends, dynamics, and potential confounds for subsequent research.

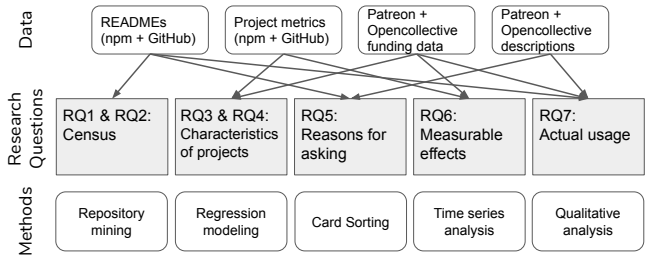


Figure 2: Overview of our research.

4.1 Research Methods

Our primary strategy to identify which repositories ask for donations, and with which donation platform, was to analyze `README.md` files of GitHub repositories for links or mentions of specific donation platforms. We arrived at this method after several iterations of refinement and validation.

Corpora. As mentioned in Sec. 3, we use a smaller *npm* corpus and a larger GitHub corpus. The former includes all packages published on *npm* that link back to a valid GitHub repository (needed for analysis) as of May 23, 2019 (537,640 repositories). The latter contains all 77,934,441 GitHub repositories indexed by GHTorrent [28] at that date, that are not forks of other repositories[1] and not marked as deleted. The former corpus is a subset of the latter.

We chose May 23, 2019 for taking a snapshot and as the end date for all longitudinal analyses, because GitHub announced its own donation platform feature that day [81], which may affect our results but is too recent to reliably study its effect.

Identifying donation requests. We started with a popular and manually curated list of funding models for open source [16] to seed a list of 11 donation and crowdsourcing platforms, including *PayPal*, *Patreon*, and *Liberapay*. We then manually sampled projects that used these services and inspected their GitHub repositories for signals indicating that they were doing so, observing that the funding platforms were almost always referenced as links in the repositories' `README.md` files.

Next, we iteratively curated a list of donation platforms and corresponding search strings that would identify them in a markdown file. To go beyond the initial 11 ones, we used two strategies: First,

[1]Requests for donations are automatically copied with the fork, adding noise. During our manual analyses (details below) we also noticed that some repositories are copies of other repositories without being labeled as forks on GitHub / in GHTorrent; the repository name and contents are typically clear indicators of the duplicate status. The duplicate repositories also tend to be abandoned. To automatically identify and exclude such fork-like cases from our corpora, we developed and manually validated a simple heuristic: among repositories with the same name but hosted under different GitHub accounts, all of which link to the same donation service profile, only keep the repository with the most GitHub stars and remove all others.

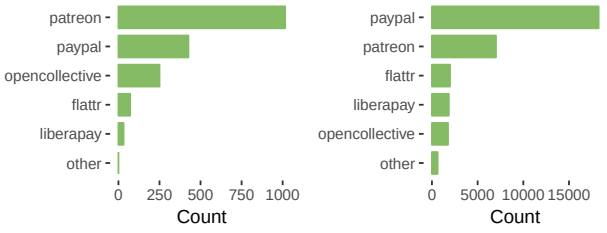


Figure 3: Most popular donation services in *npm* (left) and across GitHub (right) as of May 23, 2019.

we used online searches for donation platforms. Second, we conducted a large-scale *grep* search for funding-related terms across all markdown files in all 537,640 repositories in our *npm* corpus, clones of which we had available locally on disk, using the terms "donate," "sponsor," and "donation;" we then manually investigated a random sample of 367 of the 8093 *npm* repositories with such terms (representative sample at 95 % confidence level for ±5 % confidence interval) for mentions of platforms that we missed. We found two forms of donations that we could not accurately detect automatically and thus excluded from our analysis: mentioning *sponsors* but without an explicit platform to manage sponsorship and *Bitcoin* were occasionally used but stated in many different ways so that all search patterns we tried would either only recognize a small subset of these repositories or yield too many false positives to be useful. Our final list contains 12 donation platforms, all automatically detectable in markdown files: *Bountysource*, *Flattr*, *IssueHunt*, *Kickstarter*, *Liberapay* (including *Gittip* and *Gratipay*), *OpenCollective*, *Otechie*, *Patreon*, *PayPal*, *SALT*, *Tidelift*, and *Tip4Commit*.

Some open-source projects request donations in multiple places. For example, it is common to also link to donation platforms from the project's web page or to create a specific `SPONSORS.md` file (or something similar). Doing a full analysis of the *npm* corpus, we found that it is not uncommon to link donation platforms in other markdown files, but in nearly all cases (98.2 %) those links are also included in the `README.md` file; in addition, typically the link in the `README.md` file was added earlier. Similarly, a manual analysis of a random sample of projects for each donation platform showed that links on a project's web page are also usually redundantly included in the `README.md`. Hence, we decided to search only in the `README.md`, which is sufficiently accurate and more computationally efficient.

Collecting snapshot frequency data. For the 537,640 repositories in the *npm* corpus, we performed a *grep* search over the `README.md` files in the default branch and the latest revision on or before May 23, 2019. The GitHub corpus is too big to download all repositories or even just all `README.md` files, hence we relied on the GitHub API to search for projects that contained the donation-service-specific terms in their `README.md`s. To avoid exceeding GitHub's API limits, we partitioned the search space using custom file size queries. We then cloned the identified candidate projects and ran exactly the same *grep* query on their May 23rd revision as we did for the *npm* corpus.

Collecting longitudinal frequency data. For both corpora we identified when each donation platform was first introduced in the `README.md` file and when it was removed (if ever). From the *npm* corpus we analyzed all projects longitudinally; from the GitHub corpus we analyzed all projects identified in the static snapshots, *i.e.*, at the scale of the GitHub corpus, we cannot identify projects that previously asked for donations but no longer do now. Technically, we iterated over all revisions to identify the timestamp of the commits that changed the result of our donation-platform detector for each donation platform.

Threats to validity. As discussed, we cannot capture donations outside of common donation platforms that are added as sponsors, and we cannot capture donation requests through *Bitcoin*. Despite careful validation, we may miss some donation platforms. Our results should be considered as lower bounds, given that the detector has high precision (we are not aware of any instances where a detected link to a donation service was not used for requesting donations), but also given that we may miss some repositories requesting donations (a) when requests are made outside the `README.md` file or (b) due to limitations of the GitHub API. The distribution of results for missed repositories may differ from the one for detected repositories, though analyzing the 1.8 % repositories of the *npm* corpus missed by focusing on the `README.md` file instead of all markdown files revealed no obvious deviations.

4.2 Results

Overall, there are many projects asking for donations, but they make up only a small fraction of our corpora (RQ 1): we found that on our cutoff date, 1,145 out of the 537,640 repositories of our *npm* corpus (0.2%) ask for donations. In our GitHub corpus the proportion of projects asking for donations is even lower, with 25,885 out of 77,934,441 repositories (0.04%).

As summarized in Fig. 3, *Patreon* is the most common donation platform by number of repositories in *npm*, followed by *PayPal* and *OpenCollective*, then with much lower adoptions come other services (RQ 2). Across the larger GitHub corpus, *PayPal* ranks first, which suggests that the *npm* corpus is not representative of all GitHub when it comes to fundraising; this divergence should be further explored by future work.

Developers slowly started to ask for donations (in the form of links to donation platforms) around 2012, with a significant increase in recent years, as visible in Fig. 1; the only stagnating platform is *Flattr*, likely related to the controversy around its sale in 2017.

While most repositories adopt only a single donation platform, 10.9 % of repositories ask for donations with multiple platforms (*PayPal* + *Patreon* is the most common combination). Transitions between donation platforms are relatively rare after the initial adoption so is the removal of donation links.

5 STEP 2: CHARACTERISTICS OF PROJECTS ASKING FOR AND RECEIVING DONATIONS

After identifying that only a small percentage of projects ask for donations, we ask **RQ 3: What are the characteristics of repositories asking for donations?** and **RQ 4: What are the characteristics of repositories receiving donations?** For the latter, we only analyze the repositories receiving funds through the funding platforms *Patreon* and *OpenCollective*, which are popular and make the funding amounts publicly available. For both questions, using

multiple regression modeling, we check which publicly observable project characteristics, such as popularity, activity level, or number of contributors, associate with asking for or receiving donations.

5.1 Research Methods

The key idea is to create a sample of two types of repositories — some that ask for donations and some that do not — then collect characteristics of each repository and use statistical modeling to identify which characteristics tend to associate, on average, with repositories being of one or the other type. Similarly, for repositories that ask for donations we model which characteristics associate with more successful fundraising, in terms of amounts of donations received.

Corpora. To study characteristics of repositories asking for donations (RQ 3), we cannot take the entire corpus from Step 1, because only a small fraction of those repositories ask for donations, making statistical modeling unsound. Instead, we take all the repositories asking for donations from Step 1 and compare them to a control group[2] — a random subset — of repositories from the prior corpus not asking for funding. We do this both for our *npm* corpus (1,145 asking for funding, 8,124 not asking, post outliers) and our GitHub corpus (25,885 asking, and 109,289 not asking, post outliers).

To study who receives donations among those asking (RQ 4), we only analyze the 817 *npm* and 6,516 GitHub repositories asking for donations with either *Patreon* or *OpenCollective*, as only these two services publicly display the received amount of donations.

Funding levels. We collected the funding level of a repository from *Patreon* and *OpenCollective* for the associated donation profile (personal profile on *Patreon* and project page on *OpenCollective*). As the funding level, we measure the amount of donations received within the last 9 months before our cutoff date of May 23rd, 2019. Although *Patreon* only reports current monthly donations, the *Graphtreon* website tracks donations over time; *OpenCollective* natively provides a log of all transactions (earnings and expenses).

Project characteristics. For each repository, we collect a number of characteristics that capture different dimensions of activity and popularity, using a combination of locally cloned git repositories, GHTorrent, the GitHub API, and the *npm* API, as appropriate: the *total number of commits* up until May 23rd, 2019; a binary flag *is active* indicating whether the repository had at least one commit in the 9 months before that day; the *size of the repository*, measured in kilobytes; the *project age* measured as months between the creation of the repository and May 2019; a binary flag *is org* indicating whether the repository is owned by an organizational account on GitHub; the *total number of issues* on GitHub; and the *total number of stars* on GitHub. In addition, for the *npm* repositories, we can collect additional data: the *total number of downloads* and the *reverse dependency count*, indicating how many other *npm* packages depend on the given repository.

Modeling. We estimate two sets of multiple regression models. For RQ 3, we regress the binary dependent variable *asks for donations* on all the explanatory variables above using logistic regression. For RQ 4, we estimate hurdle regression models with the same explanatory variables to understand, among those repositories asking for donations, (a) what distinguishes those that receive any from those that do not, and (b) only for those receiving donations, how the amount received varies with the different project characteristics. We use this split modeling strategy, known as hurdle regression, because of the zero-inflation of our response variable — that is, many projects asking for donations do not receive any.

We then follow a standard practice for model fit and diagnostics. We log-transform variables with skewed distributions to reduce heteroscedasticity [24]; the model summary tables mention "(log)" next to all transformed variables. We also conservatively remove outliers for predictors with exponential distributions, *i.e.*, those values exceeding $k(1+2/n)\text{median}(x)+\theta$ [56], where θ is the exponential parameter [60], and k is computed such that no more than 1 % of values are labeled as outliers; typically among these there are high-leverage points that disproportionately affect regression slopes, reducing the robustness of our models. We test for multicollinearity using the variance inflation factor (VIF), comparing to the recommended maximum of 5 [10]; the *total number of issues* variable exceeded the threshold and was subsequently removed from the models; all other variables were within acceptable VIF bounds. We assess goodness of fit using the standard pseudo-R^2 for the linear models and using McFadden's pseudo R^2 [72] for the logistic models. We check the diagnostic plots for violations of modeling assumptions, finding none that would invalidate the models. Finally, we report the regression coefficients together with their p-values. The estimated coefficients do not depend on the order of predictors in the regression equations (*i.e.*, we ran one-shot regressions). For each estimated coefficient, we also report the units of variance explained (the "Deviance" and "Sum sq" columns in the table), as derived from ANOVA type-II analyses (*i.e.*, each variable is added after all the others); these values, when relative to the total amount of variance explained by a model (*i.e.*, the column total), serve as a proxy for effect size.

Threats to validity. As in all empirical studies of this kind, our measures for the studied characteristics can only capture some aspect of the underlying quality, and there are project characteristics that we could not measure at scale and thus not did include in the models (*e.g.*, presence at conferences, marketing). We also note that projects using other fundraising mechanisms than *Patreon* and *OpenCollective* may differ from those using these two platforms in unknown and unpredictable ways; our results should only be interpreted with respect to this sample. Finally, our analysis does not distinguish between donations requested for individuals and donations for projects.

5.2 Results

Overall, the logistic regression model of *npm* projects asking for donations compared to a randomly sampled control group of *npm* projects not asking for donations (RQ 3; Table 1) fits acceptably well ($R^2 = 31$ %), revealing several characteristics that distinguish the two groups of projects on average, all with sizable effects. Recently active projects are more likely to ask for donations (the strongest effect in the model, 53 % of the variance explained), as are more mature projects (*num commits*, 8 %; *project age*, 9 %), holding other variables fixed. Project popularity correlates positively with the likelihood of asking for donations (*num stars*, 14 %; no additional variance

[2] Note that, due to the lack of a donation platform link, we cannot remove duplicate fork-like repositories from the control group as we did in Section 4.

Table 1: Characteristics of *npm* projects asking for donations.

	Resp: *Asks for donations*	
	Coeffs (Err.)	Deviance
(Intercept)	−4.01 (0.19)***	
commits (log)	0.40 (0.05)***	72.95***
size (log)	−0.30 (0.03)***	125.74***
project age	0.02 (0.00)***	85.94***
is active	1.95 (0.09)***	502.20***
is org	−0.57 (0.10)***	33.63***
stars (log)	0.27 (0.02)***	129.89***
downloads (log)	−0.02 (0.02)	0.88
dependents (log)	0.01 (0.05)	0.02
Num. obs.	9137	
R^2	0.31	

$^{***}p < 0.001, ^{**}p < 0.01, ^{*}p < 0.05$

Table 2: Characteristics of *npm* projects receiving donations via *Patreon* and *OpenCollective*.

	Hurdle model Resp: *Received any*		**Count model** Resp: *Amount received*	
	Coeffs (Err.)	Deviance	Coeffs (Err.)	Sum sq.
(Intercept)	0.12 (0.38)		4.17 (0.39)***	
commits (log)	−0.20 (0.12)	3.05	−0.26 (0.11)*	20.41*
size (log)	−0.10 (0.06)	2.80	0.06 (0.07)	2.67
project age	0.05 (0.01)***	58.63***	−0.01 (0.00)	10.93
is active	1.33 (0.22)***	38.73***	0.00 (0.21)	0.00
is org	0.84 (0.26)**	10.78**	0.12 (0.20)	1.37
stars (log)	0.14 (0.06)*	6.06*	0.39 (0.06)***	182.17***
downloads (log)	−0.11 (0.06)	3.51	0.13 (0.05)**	28.60**
dependents (log)	0.31 (0.11)**	8.98**	−0.04 (0.08)	0.85
Num. obs.	735		527	
R^2	0.29		0.30	

$^{***}p < 0.001, ^{**}p < 0.01, ^{*}p < 0.05$

explained by *num downloads*). Repository size (13 %) and organizational affiliation (4 %) both correlate negatively with the likelihood of asking for donations — bigger projects that are part of GitHub organizations may be sustained through other channels, perhaps needing donations less. The equivalent model of the whole GitHub corpus (see supplementary material) confirms the main effects. Projects asking for donations tend to be more popular (*num stars*, 81 % of the variance explained), more mature (*num commits*, 13 %), and active recently (5 %); other variables have negligible effects.

Looking at the 817 *npm* projects using the *Patreon* and *OpenCollective* services, we observe a highly skewed distribution of received funds: min 0 USD, median 55 USD, mean 6108 USD, max 589,382 USD in the observed 9 month window. Regarding who receives funding (RQ 4), not all repositories that ask for funding actually receive any: only 575 (70 %) received any funding at all, and only 45 repositories (5 %) received more than 1000 USD/month — the level at which donations could support a single developer at the 2018 US income poverty guidelines [68].[3] The situation is similar across the larger GitHub corpus: 58 % of repositories received donations; only 10 % received 1000 USD/month or more.

In the hurdle regression (RQ 4; Table 2), which distinguishes *npm* projects that receive funding from those that do not, we observe a positive correlation of the activity and popularity metrics with the likelihood of receiving any funding. The strongest effects are: *project age* (44 % of the variance explained), perhaps indicative of the importance of reputation in the community, which develops with time; and being *recently active* (29 %). Popularity as indicated by the *number of stars* (4 %) and usage as indicated by the *number of reverse dependencies* (6 %) are also positively correlated with the likelihood of attracting funding. The count model (RQ 4; Table 2) reveals that among projects receiving donations, only popularity (*number of stars*, 74 %; and *number of downloads*, 12 %) correlates with the amounts received. The equivalent models for the larger GitHub corpus (see supplementary material) do not fit the data well, suggesting that these characteristics might be *npm* specific. More targeted analyses of different communities could help explain this phenomenon, but go beyond the scope of this paper.

6 STEP 3: EXPECTATIONS FOR DONATIONS

To understand whether donations are effective, we first need to understand the *expectations* that developers have toward them. Hence, we ask: **RQ 5: What do developers who ask for donations want to spend the money on?** This question will help us develop hypotheses to test whether expectations match outcomes.

6.1 Research Methods

As discussed in Sec. 3, we deliberately avoid surveys and instead analyze already publicly available text. Specifically, we analyze a sample of `README.md` files and profiles on online donation platforms to identify reasons, hopes, goals, or statements about how developers would spend funds they receive.

For this analysis, we assembled a sample of repositories with corresponding `README.md` files and donation profile pages. Specifically, we downloaded the HTML donation profile pages of all *npm* repositories using *Patreon*, *OpenCollective*, or *Kickstarter*, because on those platforms, it is common to describe the project, funding levels requested, and so forth; this resulted in 1259 candidate projects. We then created a sample for subsequent qualitative analysis, as follows. Observing that *Patreon* and *OpenCollective* profile pages are often empty, we sorted all the HTML pages by file size and sampled 100 randomly from among the larger half — the larger HTML files are likely to contain more detailed descriptions. We additionally selected the 14 largest *Kickstarter* campaigns in the *npm* corpus, because they are typically very detailed. After removing 5 duplicates, our final sample contains 109 repositories.

We qualitatively analyzed the text of all `README.md` files and donation profile pages in our sample using *card sorting* [57]. Specifically, two researchers extracted all text fragments that refer to *expectations* for funding, such as reasons given, goals set, or intentions on how to spend funds received. We printed all fragments and then

[3] A donation income of 1000 USD/month could support a fulltime position in some countries with lower costs of living and may provide part time support for developers everywhere, however it is still far from a competitive salary for IT professionals.

sorted them into groups, discussing and refining groups in the process. Initial inter-rater agreement between the two coders was high (Cohen's kappa > 0.63); discrepancies were then discussed and the cards were re-sorted together to finalize the categorizations.

Threats to credibility. Self-described goals for funding may not honestly reflect the developers' intentions and expectations, as developers might self-censor when they perceive expectations to be less socially acceptable [8]. While we cannot avoid such potential bias, we note that it would similarly surface in interviews or surveys. Though a discrepancy between what is displayed online and reality is possible, project web pages contain all the information that open source developers want other people in the community to know about their projects. Since this content is what potential donors will be seeing, it making sense to analyze it. We also point out that the identified reasons contain those that are frequently mentioned in grey literature (*i.e.*, practitioners' blog posts and talks about funding in open source).

6.2 Results

A large number of analyzed repositories (41 %) did not list any expectations or justifications or merely mentioned that donations "show appreciation." Among the identifiable expectations, we identified 4 themes:

Engineering: The requested funding is intended for creating new features, resolving issues, improving the project, or paying salary for a developer by 48 % of the sampled repositories. While some repositories request funds for specific features (especially in *Kickstarter* campaigns), requests are often more generic — which is also why we could not separate requests to support maintenance from new development clearly. For example, one developer mentions in a *Patreon* page *"I will be able to dedicate more time to the development and improvement of existing components and plugins."* Several projects explicitly suggest that the project is not sustainable in its current form and funds are needed to continue development and maintenance, often in a form that reads as a "cry for help", *e.g.*, *"Before I give up [the project] completely I wanted to give this one last try and see if I can find enough supporters who would like to use [it] and a newer better version of it in the future!"*

Community: Funding is requested for creating documentation and tutorials, keeping the project ad-free for users (which improves the user experience for the community), supporting other projects being used by the current one, and other community activities in 18 % of the sampled repositories, *e.g.*, *"I can spend less time thinking about private monetization channels (e.g., taking on support/consulting contracts) and instead work more on content that benefits the entire community, e.g., more educational blog posts, videos and even books!"*

Project expenses: Funding is requested for cost associated with running the project, commonly server and hosting fees, in 13 % of the sampled projects, *e.g.*, *"The money from this Patreon keeps the servers for my projects running."*

Personal: Funding is requested for taking time off from one's job to work on open source, paying off loans, buying coffee, and generically increasing motivation in 9 % of the sampled projects, *e.g.*, *"The $3,000 per month will be put toward my living expenses and the student loan bills that I will need to start paying off during the project."*

Overall, donations are requested predominantly, but not exclusively, to support engineering activities (RQ 5).

7 STEP 4: OBSERVABLE OUTCOMES OF DONATIONS

After analyzing the intentions of developers for donations in the previous step, we now ask **RQ 6: Do donations have measurable effects on development and maintenance outcomes?** Specifically, we observe and quantitatively model, across a large number of projects, whether observed outcomes tend to change as projects request and receive donations, using interrupted time series models.

While there are many interesting potential outcomes that can be derived from the developers' stated intentions, we focus on engineering activities, which was the most commonly given reason and can be operationalized at scale with public archival data. Specifically, we observe the combined maintenance, development, and support activities through two proxy measures:[4] *number of commits*, which captures both coding and noncode activities; and *issue resolution speed*, which captures the efficiency of maintenance and community support activities.

7.1 Research Methods

While receiving donations may have noticeable effects on the activity metrics in specific individual projects, we are interested in effects that generalize across many projects — only then can we be confident that donations could be an effective mechanism to promote open-source sustainability more generally. The key idea behind our analysis is treating donations as an *intervention* and observing (and modeling) how potential trends in the outcome measures changed, if at all, after the intervention, across a large sample of projects aligned on their respective intervention dates. Over a large-enough sample, aligning the different projects' time series on the intervention date, which likely occurred at very different dates for each project, enables us to assume that the effects of potential environmental confounding factors on the observed trends are uniformly distributed. Therefore, any observable trends in the outcome measures can be attributed to the intervention. This approach is known as an *interrupted time series design* [5]; it originated in medical research and has since been applied successfully in software engineering [e.g., 69, 78].

Variables. To perform this kind of modeling, we need to collect historic data on donations, outcomes, and covariates. We collect:

(1) Intervention date: We record the date of the first donation received by each project via *Patreon* or *OpenCollective*. We also considered the date when a project posted the call for donations, but decided against it, as there may be a lag before donations start to arrive, which could impede the activity in the project.

(2) Monthly donations (control): We collect cumulative donations from *Patreon* and *OpenCollective* in the observation window, as described in Sec. 5. If the funding is used to support developers' time, the amounts of funding could impact the levels of activity; we model total rather than monthly donations because funds are

[4] Unfortunately, other hypothesized outcomes like increased developer motivation and reduced developer stress are very difficult to observe and study longitudinally. While it would be possible to recruit a cohort of developers and poll them repeatedly, for example with the *Maslach Burnout Inventory* survey [61], such a study is very difficult to conduct and far exceeds the scope of our exploratory work.

not necessarily spent in the same month they are raised.

(3) Monthly issues closed (control): Using GHTorrent, we collect the number of issues closed per month. During periods with higher workload (more issues closed), the issue resolution times may naturally increase, thus we add this control to the model. We excluded issues that were closed within a minute of opening, typically caused by bot activities.

(4) Monthly commits (outcome): We collect the number of commits per month using GHTorrent. Commits have been shown to be a robust measure of a project's level of development activities [71].

(5) Issue resolution speed (outcome): Using GHTorrent, we record the average time that issues closed in a given project and month have been previously open.

Sample. The modeling technique sets requirements on which projects can be analyzed. Specifically, the projects need to have received donations (intervention) and need to have sufficient activity before and after their first donation, such that trends and changes therein, if any, can be estimated statistically; at least 9 months before and after the intervention has been used in the past [69, 78], hence we choose this threshold here. Since many projects in our previous analysis have adopted donations early in their history or only recently received their first donations or have not been active during the entire period of observation (9×2 monthly windows), our corpus for this analysis is necessarily smaller. Again, to collect received donations, we need to limit our analysis to projects using *Patreon* or *OpenCollective*.

Another challenge relates to the aggregation level. During our earlier manual investigation we found that it is common for multiple repositories to ask for donations using the same profile (URL) on a donation platform. The typical scenarios are: (a) a larger project organized into multiple GitHub repositories that all link to the same donation profile for the larger project; and (b) developers asking for donations with a personal profile on multiple of their own repositories. Since we can determine funding level only per donation profile, we further group repositories by donation profile and treat the activities in the entire group as a single repository, *i.e.*, we aggregate all the variables above per donation profile.

Since these limitations restrict the pool of projects, we only ran this analysis on the larger GitHub corpus, with 337 projects meeting our conditions, of which only 16 projects received more than 1000 USD in the analyzed 9 month window.

Statistical modeling. We model the two interrupted time series, one per outcome variables, as multiple mixed-effects linear regression models, similar to prior work [69, 78]. Besides the variables above, we include two integer counters for time window, whose estimated coefficients capture the trend before the intervention (*month_index*) and change in trend after the intervention (*month_after*); moreover, we include a dummy variable *intervention*, whose estimated coefficient captures the change in level associated with the intervention. We follow standard model fit and diagnostic procedures, as described above in Sec. 5.

Threats to validity. The analysis is constrained by the outcomes we were able to quantify, *e.g.*, not covering outcomes regarding non-code activities or developer stress. Further, as donations often ramp up slowly, it is difficult to define an exact point that should be considered as an intervention; our results are robust though to different operationalizations (first time asking for donations and first time receiving donations at different thresholds). Finally, results may be dominated by the majority of projects that receive low levels of donations; considering only high-earning projects leaves too few projects for confidence in the statistical results.

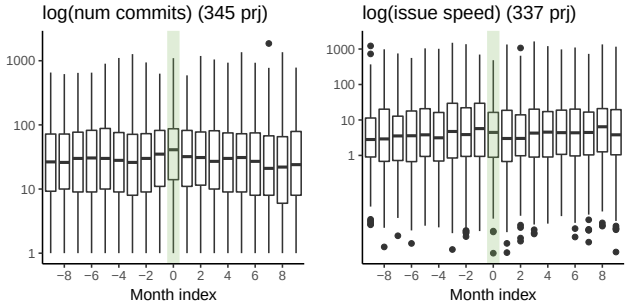


Figure 4: The distribution of outcomes before and after receiving the first donation.

Table 3: Interrupted time series regression models.

	Commits model Resp: *num commits (log)*		**Issue speed model** Resp: *issue speed (log)*	
	Coeffs (Err.)	Chisq	Coeffs (Err.)	Chisq
(Intercept)	0.79 (0.42)		1.79 (0.29)***	
earnings (log)	0.30 (0.06)***	21.25***	−0.14 (0.04)**	9.75**
closed issues (log)			0.38 (0.03)***	134.65***
month_index	0.15 (0.02)***	59.59***	0.05 (0.02)**	10.69**
intervention	0.03 (0.16)	0.05	−0.37 (0.12)**	9.72**
month_after	−0.25 (0.03)***	79.70***	−0.01 (0.02)	0.27
Num. obs.	6210		4912	
R^2	0.36		0.33	

$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$

7.2 Results

Figure 4 shows how the distributions, computed over all the projects in our sample, of the two monthly outcome measures, *number of commits* and *issue resolution speed*, evolve before and after projects start receiving donations. Visually, the trends appear stationary.

To formally test these trends, we turn to the regression models in Table 3. First, we observe a slight increasing trend of **commits** over time (β(month_index) = 0.15), which gets reversed after the intervention (β(month_index) + β(month_after) < 0), *i.e.*, the pre-donations commit activity increases slowly, but that increase is not sustained in the long term, after the first donation. The intervention itself does not appear to be associated with any short-term shift in commit activity on average (β(intervention) is indistinguishable from zero). Notably, the funding level associates positively with the amount of commit activity: projects with higher overall funding tend to be more active; for every factor e increase in earnings (note the log-transformed predictor), the number of commits is expected to go up by $e^{0.3} \simeq 35\,\%$, other variables held fixed (due to the different corpus this effect is not comparable to Sec. 5). The fraction

of variance explained by this predictor, however, is small.

The **issue resolution speed** model reveals an increasing trend pre-intervention (β(month_index) = 0.05), *i.e.*, over time issues take longer to resolve on average, after controlling for the total monthly workload. The trend does not change after the first donation received (β(month_after) $\simeq$ 0). However, we note a negative effect associated with the intervention itself (β(intervention) = −0.37), suggesting a short-term decrease in issue resolution speed after the first donation. Taken together, the two coefficients suggest that the impact of donations is short-lived. As in the previous model, increases in funding level associate with improvements in issue resolution speed (note the reverse-coded dependent variable); the issue resolution speed decreases by $e^{0.14} \simeq 15\,\%$ for every factor e increase in earnings, other variables held fixed. The fraction of variance explained by this predictor is again small.

To explore the effect of projects asking for donations as a 'cry for help' (see Sec. 6), we explored another model among projects that received no donations at all, using the time of *asking* for donations as the intervention. Plots and model (in supplementary material) show that after stagnant activity leading up to asking for donations (and not getting any), activity levels actually *decline* slowly.

In a nutshell, we find some evidence, but not strong support, for the hypothesis that donations lead to higher levels of development or maintenance activity (RQ 6), though we find that projects asking for but not receiving any donations actually decline in activity. We suspect that the positive outcomes of donations may be more subtle or more varied than captured by our analysis, therefore, in a last step, we explore a small sample of projects in more depth.

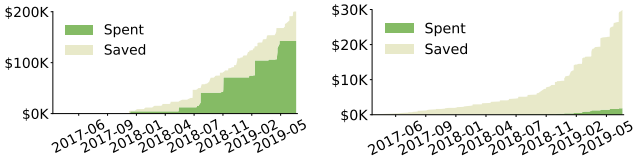
8 STEP 5: USE OF DONATIONS

To explore reasons for the limited observable effects of donations, we analyzed individual projects in more depth and found that the use of donations (when publicly visible) differed significantly among projects. We thus ask **RQ 7: How are donations used?**

8.1 Research Methods

After some exploratory analysis of 15 high-donation and medium-donation projects, we identified that projects using *OpenCollective* would provide the most insights because the donation profiles are usually most detailed and they include a public list of expenditures. We assembled a new corpus of 60 projects that asked for donations using (only) *OpenCollective*. We started with the 15 *npm* projects receiving the largest donations. To achieve diversity in our corpus, we added 45 additional projects using stratified sampling: We divided the *OpenCollective* projects in our GitHub corpus into three strata – projects making more than an average of 1000 USD/month in the last 9 months, projects making less than 100 USD/month, and projects in between – randomly picking 15 projects each.

We started our exploration by automatically assembling summary descriptions for each project. Each description contains data and plots on transaction histories and funding amounts collected from *OpenCollective*. We then enriched the automatically assembled summaries with notes by manually investigating public artifacts of each project. We carefully read the project's `README.md` file, web pages, documentation, and donation profile pages, and investigated their donation history (earnings and expenses) as well as their



(a) Example spender project **(b) Example saver project**

	Spending < 25 %	Spending > 75 %
Donations > 1000 USD/month	37 %	17 %
Donations 100–1000 USD/month	59 %	10 %
Donations < 100 USD/month	68 %	11 %

(c) Spenders and savers among *OpenCollective* projects on GitHub

Figure 5: Spenders and savers.

GitHub activity charts at the repository and contributor levels. To guide our investigation into how the donations are used, we attempted to gather information on why the project asks for donations, who receives the funds, and what the funds are used for. After collecting raw observations for all 60 projects, three authors read and discussed the results, looking for patterns in the data and discussing potential implications of the observations.

Where possible, we then operationalized specific observations (*e.g.*, projects receiving but not spending donations, see below) to quantify how frequent they are across all 540 projects in our GitHub corpus that received donations through *OpenCollective* in the last 9 months before our cutoff date May 23, 2019.

Threats to validity and credibility. Our observations can only partially explain the previously observed mismatch between expectations and outcomes regarding donations. Relying on public data, rather than conducting in-depth interviews may lead to overgeneralizations. Results may differ for projects that use other donation services, for example, as *OpenCollective*'s public accountability may influence behavior, even though it is explicitly designed to accurately depict all project transactions.

8.2 Results

In the following, we discuss our observations, grouped by themes that emerged during our qualitative analysis.

Savers vs spenders: We found that there are big differences regarding spending: Some projects actively spend all raised funds while others barely spend any, sometimes accumulating significant amounts of money — in Figure 5 we show two examples.

In our sample, 24 projects (40 %) spent less than 25 % of their raised donations, and 9 projects (15 %) spent more than 75 %. When automating this analysis across all *OpenCollective* projects in our GitHub corpus, we found that saving is common in general, with 64 % spending less than 25 % of their received donations and only 11 % spending more than 75 %. The results can be observed across different funding levels, though saving is less frequent among well-funded projects as shown in Figure 5c — some projects with lower funds may wait to accumulate more money for bigger expenses.

Types of expenses: We observed that most donations received through *OpenCollective* were spent on engineering-related expenses,

Table 4: Distribution of 2957 *OpenCollective* transactions across 540 GitHub projects.

Category	Transactions	Amount in USD
Engineering	1,133	1,316,225
Services	493	238,761
Travel/Food	333	104,875
Donations	591	79,751
Social	296	61,128
Unclassified	111	20,421

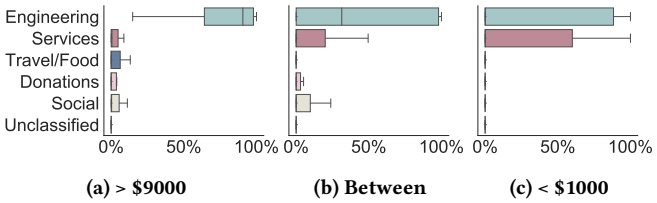


Figure 6: Distribution of *OpenCollective* transaction categories across GitHub projects at three funding levels.

including paychecks to developers. Projects with lower volumes of donations tend to spend a higher fraction on project expenses, such as hosting and domains fees.

An interesting outlier in our sample is a well-funded project owned by a large corporation, in which the main developers are employees and donations (also largely but not exclusively received from the corporation) are spent on supporting travel for other community members and other community activities.

Usually, the way that projects spend funds aligns with what they request donations for (if they give specific expectations in their requests and if they receive enough donations). Of the 21 projects that have clear reasons to ask for donations and clear expenses, only one showed a discrepancy: it asked for donations to support full-time maintainers and to grow tooling support but only spends money on branding and travel; however, its donation income and expenses are fairly low, so it is possible that funding engineering is not feasible at the current funding level.

To automate this analysis across all GitHub projects collecting donations using *OpenCollective*, we manually developed a classifier to categorize a transaction description using keyword matching into the categories *engineering*, *services* (*e.g.*, servers), *travel/food* (*e.g.*, conference fees), *donations* to other projects, and *social* (*e.g.*, merchandise). As shown in Table 4, engineering costs are still dominating, with service costs in second place, aligning with the expectations we identified in Sec. 6. A breakdown by funding level (Fig. 6) supports our observation that projects with high funding levels spend funds primarily on engineering, whereas projects with lower funding levels spend a larger proportion on services.

Recipients of funds: A majority of projects in our sample (43, or 72 %) gave money to their top contributors. Among those, 21 projects gave money also to non-top contributors. Only two mid-sized projects gave money only to non-top contributors. For one project it was intended for hosting expenses, and for the other project it was for maintenance work done by an external contributor. In general, projects that received fewer donations tended to give money to fewer people and mainly to a single top contributor. Projects that gave money to non-top contributors did so for various reasons including investing in newcomers, paying for specific contributions (*e.g.*, bounty claims), and paying travel costs for speakers.

We expect that donations can significantly contribute to the functioning of a community, encouraging onboarding, supporting community events, or documentation, that may be hard to observe directly in the outcome measures used in our analysis (Sec. 7).

9 DISCUSSION

Though only a small percentage of all open-source projects ask for donations, we find that donations are a common mechanism to support open-source work. As expected, more established and more active projects are more successful at raising funds. However, donations alone rarely ever raise enough funds to support paying full-time developers for their work. The current level of funding seems to provide at most marginal observable benefits to project productivity in terms of the two variables we measured; the decline of activity in projects unsuccessful in raising donations (Sec. 7) is worrying for sustainability in general.

Our study is exploratory in nature and can only be a first step in understanding the current state of donations in open source using public data. Of course, there are many facets that we cannot explore with our methods, as large parts of the funding landscape are not publicized (*e.g.*, *PayPal* donations, sponsorship). Still, our work reveals insights that can be starting points for many interesting future research directions.

First, we suggest researchers should explore *What level of funding is actually needed to sustain open-source projects?* From our observations (Sec. 8), we suspect that the answer varies widely from funds to support multiple full-time engineers, to supplementing one maintainer's other income, to server costs, or to mere 'thank you' gestures. We see that operating expenses, *e.g.*, hosting costs, are typically paid first. It is unclear whether many projects are not spending their raised funds because they are waiting for larger expenses, because they do not have more expenses, or because they do not feel comfortable accepting money personally [cf. 37].

Second, another interesting direction is to compare different forms of spending, or *What kind of spending is effective at sustaining open-source projects?* We see a wide variety of different forms of spending (Sec. 8), including paying engineering salaries, spending on advertisement and community activities, supporting travel, and paying non-core contributors — can the return on investment of such different strategies be quantified and compared, as the movement on effective altruism [66] does for charities? Unfortunately, too few projects are transparent about their spending to use our interrupted-time series method (Sec. 7), suggesting more qualitative longitudinal research approaches. Ideally, subsequent research should look at more nuanced outcomes (Sec. 6), such as success at onboarding new contributors, reducing developer stress, and user satisfaction, beyond the commit and issue activity in our study.

Third, while there are significant differences between open source and traditional charitable organizations, we believe the open-source community can learn from empirical research on philanthropy

to determine *How to effectively raise donations for an open-source project?* For instance, a recent metastudy identified 8 mechanisms that drive charitable giving [2]. Our analysis of which projects receive donations (Sec. 5) seems to align with the importance of *reputation* observed in philanthropy, but many more factors can be explored. For example, our results show that many projects are not explicit about goals when raising *awareness of need* (Sec. 6) and most projects do not demonstrate *efficiency* of using funds in that they are rarely transparent about how they use received donations and what outcomes they achieved (Sec. 8). As another example broadly studied in philanthropy, it is worth exploring the positive and negative effects of providing direct or indirect incentives to the donor, from sending stickers, to promoting them as sponsors, to preferential handling of their issue reports [53].

Finally, there is a dark side of money in open source that is worth exploring: *What are negative effects of collecting donations and how can they be mitigated?* Zhou et al. [79] observed that increased involvement of payed developers in an open-source project can crowd out volunteers; some developers observe more entitled users even when small amounts of money are involved (*e.g.*, expecting rapid attention to their issues because they attached a $25 bounty);[5] and preferential treatment of sponsors (*e.g.*, in issue resolution [53] or code review [25]) may be perceived as unfair. Furthermore, literature on volunteers suggests that paying volunteers a little can be worse than not paying them at all because it may shift their mindset from that of an intrinsically motivated volunteer to that of an underpaid employee [20, 21]; in that sense, observing so many open-source projects with minimal donations may be dangerous.

Overall, we argue that we need a better understanding of both positive and negative effects of donations in open source and how to more effectively raise and use donations. While we expect that donations will always be only one of many mechanisms for sustaining open source overall, we believe it may be an important one, given that it is easy to adopt (compared say to founding a company), it may support a large range of activities (including community support and travel), it is broadly supported technically (now also directly by GitHub), and there seems to be some acceptance by both individuals and corporations to support open source with donations.

10 CONCLUSION

We used mixed methods to explore the state of donations in open-source projects. We found that only a small fraction of *npm* packages and GitHub repositories ask for donations (RQ 1) and that the most commonly used donation platforms are *PayPal* and *Patreon* (RQ 2). The projects asking for and receiving donations tend to be more active, more mature, and more popular (RQ 3 and 4). When asking for donations, developers typically suggest that they want to support code and non-code engineering activities, but covering operating expenses, fostering community, and reducing stress are also mentioned (RQ 5). However, when modeling whether projects that start to receive donations actually tend to be more active in engineering activities, we do not find strong evidence (RQ 6). Exploring this seeming mismatch between expectations and outcomes, we studied how raised funds are spent and found that they are often spent in line with purposes for which they were requested (if sufficient funds are raised) but also that many projects actually only spend a fraction of their funds and that funding contributors beyond the top contributors is not uncommon (RQ 7). Overall, we provided data and raised new research questions regarding needed funding, effective spending, efficient fundraising, and the downsides of donations (Sec. 9).

[5] https://news.ycombinator.com/item?id=15747743

ACKNOWLEDGEMENTS

We thank Juan David Hoyos Renteria for his help with collecting open-source projects for the GitHub corpus and Chris Bogart for his help with writing GHTorrent queries. Overney was supported through Carnegie Mellon's Research Experiences for Undergraduates in Software Engineering. Kästner and Meinicke have been supported in part by the NSF (awards 1552944, 1717022, and 1813598) and AFRL and DARPA (FA8750-16-2-0042). Vasilescu has been supported in part by the NSF (awards 1717415 and 1901311) and the Alfred P. Sloan Foundation.

SUPPLEMENTARY MATERIAL AND REPLICATION PACKAGE

A replication package for our work, also including the statistical analyses discussed but not presented here, is available online at https://github.com/CMUSTRUDEL/oss-donations.

REFERENCES

[1] Heather Arthur. 2013. Being Ridiculed for My Open Source Project. https://harthur.wordpress.com/2013/01/24/771/ Blog post.
[2] René Bekkers and Pamala Wiepking. 2011. A literature review of empirical studies of philanthropy: Eight mechanisms that drive charitable giving. *Nonprofit and Voluntary Sector Quarterly* 40, 5 (2011), 924–973.
[3] Christopher Bogart, Christian Kästner, James Herbsleb, and Ferdian Thung. 2016. How to Break an API: Cost Negotiation and Community Values in Three Software Ecosystems. In *Proc. Int'l Symposium Foundations of Software Engineering (FSE)* (Seattle, WA). ACM Press, New York, 109–120. http://breakingapis.org
[4] Simon Butler, Jonas Gamalielsson, Björn Lundell, Per Jonsson, Johan Sjöberg, Anders Mattsson, Niklas Rickö, Tomas Gustavsson, Jonas Feist, Stefan Landemoo, et al. 2018. An investigation of work practices used by companies making contributions to established OSS projects. In *Proc. Int'l Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*. IEEE, 201–210.
[5] Donald Thomas Campbell and Thomas D Cook. 1979. *Quasi-experimentation: Design & analysis issues for field settings*. Rand McNally College Publishing Company Chicago.
[6] Brett Cannon. 2018. Setting expectations for open source participation. https://snarky.ca/setting-expectations-for-open-source-participation/ Blog post.
[7] InduShobha Chengalur-Smith, Anna Sidorova, and Sherae Daniel. 2010. Sustainability of free/libre open source projects: A longitudinal study. *Journal of the Association for Information Systems* 11, 11 (2010), 657.
[8] Looi Theam Choy. 2014. The strengths and weaknesses of research methodology: Comparison and complimentary between qualitative and quantitative approaches. *IOSR Journal of Humanities and Social Science* 19, 4 (2014), 99–104.
[9] Jailton Coelho, Marco Tulio Valente, Luciana Lourdes Silva, and André Hora. 2018. Why we Engage in FLOSS: Answers from Core Developers. In *Proc. Int'l Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 114–121.
[10] Jacob Cohen, Patricia Cohen, Stephen G West, and Leona S Aiken. 2013. *Applied multiple regression/correlation analysis for the behavioral sciences*. Routledge.
[11] Eleni Constantinou and Tom Mens. 2017. An empirical comparison of developer retention in the RubyGems and npm software ecosystems. *Innovations in Systems and Software Engineering* 13, 2-3 (2017), 101–115.
[12] Kevin Crowston, Kangning Wei, James Howison, and Andrea Wiggins. 2012. Free/Libre open-source software development: What we know and what we do not know. *ACM Computing Surveys (CSUR)* 44, 2 (2012), 7.
[13] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository. In *Proc. Conf. Computer Supported Cooperative Work (CSCW)* (Seattle, Washington, USA). ACM Press, New York, 1277–1286.
[14] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. 2008. Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering*. Springer, 285–311.

[15] Nadia Eghbal. 2016. *Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure.* Technical Report. Ford Foundation. Retrieved from https://www.fordfoundation.org/library/reports-and-studies/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure/.

[16] Nadia Eghbal. 2019. A handy guide to financial support for open source. https://github.com/nayafia/lemonade-stand. Accessed Aug 16, 2019.

[17] Matthieu Foucault, Marc Palyart, Xavier Blanc, Gail C Murphy, and Jean-Rémy Falleri. 2015. Impact of developer turnover on quality in open-source software. In *Proc. Int'l Symposium Foundations of Software Engineering (FSE).* ACM, 829–841.

[18] Linux Foundation. 2019. Community Health Analytics Open Source Software (CHAOSS). https://chaoss.community/

[19] Egon Franck and Carola Jungwirth. 2003. Reconciling Rent-Seekers and Donators–The Governance Structure of Open Source. *Journal of Management and Governance* 7, 4 (2003), 401–421.

[20] Bruno S Frey and Lorenz Götte. 1999. Does pay motivate volunteers? *Working paper / Institute for Empirical Research in Economics* 7 (1999).

[21] Bruno S Frey and Reto Jegen. 2001. Motivation Crowding Theory. *Journal of Economic Surveys* 15, 5 (2001), 589–611.

[22] Vahid Garousi, Michael Felderer, and Mika V. Mäntylä. 2016. The need for multivocal literature reviews in software engineering: complementing systematic literature reviews with grey literature. In *Proc. Int'l Conf. Evaluation and Assessment in Software Engineering (EASE).* ACM Press, New York.

[23] Jeff Geerling. 2017. Don't drown in your open source project! https://www.jeffgeerling.com/blog/2017/dont-drown-your-open-source-project Blog post.

[24] Andrew Gelman and Jennifer Hill. 2006. *Data analysis using regression and multilevel/hierarchical models.* Cambridge University Press.

[25] Daniel German, Gregorio Robles, Germán Poo-Caamaño, Xin Yang, Hajimu Iida, and Katsuro Inoue. 2018. Was My Contribution Fairly Reviewed? A Framework to Study the Perception of Fairness in Modern Code Reviews. In *Proc. Int'l Conf. Software Engineering (ICSE).* IEEE, 523–534.

[26] Rishab A Ghosh, Ruediger Glott, Bernhard Krieger, and Gregorio Robles. 2002. *Free/libre and open source software: Survey and study – Part 4: Survey of Developers.* Technical Report. Int'l Institute of Informatics, University of Maastricht.

[27] GitHub. 2017. Open Source Survey 2017. http://opensourcesurvey.org/2017/.

[28] Georgios Gousios. 2013. The GHTorrent dataset and tool suite. In *Proc. Int'l Conf. Mining Software Repositories (MSR).* IEEE, 233–236.

[29] Shane Greenstein and Frank Nagle. 2014. Digital Dark Matter and the Economic Contribution of Apache. *Research Policy* 43, 4 (2014), 623–631.

[30] Stefan Haefliger, Georg Von Krogh, and Sebastian Spaeth. 2008. Code Reuse in Open Source Software. *Management Science* 54, 1 (2008), 180–193.

[31] Eran Hammer. 2018. A New Social Contract for Open Source. https://hueniverse.com/86d1fcf3e353 Blog post.

[32] David Heinemeier Hansson. 2013. The perils of mixing open source and money. http://david.heinemeierhansson.com/2013/the-perils-of-mixing-open-source-and-money.html Blog post.

[33] Dirk Homscheid and Mario Schaarschmidt. 2016. Between Organization and Community: Investigating Turnover Intention Factors of Firm-sponsored Open Source Software Developers. In *Proc. Conf. Web Science (WebSci)* (Hannover, Germany). ACM, New York, NY, USA, 336–337.

[34] Giuseppe Iaffaldano, Igor Steinmacher, Fabio Calefato, Marco Gerosa, and Filippo Lanubile. 2019. *Why do developers take breaks from contributing to OSS projects? A preliminary analysis.* Technical Report 1903.09528. arXiv.

[35] Marco Iansiti and Roy Levien. 2004. *The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability.* Harvard Business Press, Boston, MA.

[36] Javier Luis Cánovas Izquierdo and Jordi Cabot. 2018. The role of foundations in open source projects. In *Proc. Int'l Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS).* ACM, 3–12.

[37] Sandeep Krishnamurthy, Shaosong Ou, and Arvind K Tripathi. 2014. Acceptance of monetary rewards in open source software development. *Research Policy* 43, 4 (2014), 632–644.

[38] Sandeep Krishnamurthy and Arvind K Tripathi. 2009. Monetary Donations to an Open Source Software Platform. *Research Policy* 38, 2 (2009), 404–414.

[39] Karim Lakhani and Robert G Wolf. 2003. *Why hackers do what they do: Understanding motivation and effort in free/open source software projects.* Technical Report. MIT Sloan working paper.

[40] Nolan Lawson. 2017. What it feels like to be an open-source maintainer. https://nolanlawson.com/2017/03/05/what-it-feels-like-to-be-an-open-source-maintainer/ Blog post.

[41] M.M. Lehman. 1980. Programs, Life Cycles, and Laws of Software Evolution. *Proc. IEEE* 68, 9 (Sept 1980), 1060–1076.

[42] Jan Lehnardt. 2015. Sustainable Open Source. https://writing.jan.io/2015/11/20/sustainable-open-source.html Blog post.

[43] Jan Lehnardt. 2017. Sustainable Open Source: The Maintainers Perspective or: How I Learned to Stop Caring and Love Open Source. https://writing.jan.io/2017/03/06/sustainable-open-source-the-maintainers-perspective-or-how-i-learned-to-stop-caring-and-love-open-source.html Blog post.

[44] Josh Lerner and Jean Tirole. 2002. Some simple economics of open source. *The Journal of Industrial Economics* 50, 2 (2002), 197–234.

[45] Pia Mancini et al. 2017. *Sustain: A One Day Conversation for Open Source Software Sustainers – The Report.* Technical Report. Sustain Conference Organization. https://sustainoss.org/2017-report/

[46] Jennifer Marlow and Laura Dabbish. 2013. Activity Traces and Signals in Software Developer Recruitment and Hiring. In *Proc. Conf. Computer Supported Cooperative Work (CSCW)* (San Antonio, Texas, USA). ACM Press, New York, 145–156.

[47] Jennifer Marlow, Laura Dabbish, and Jim Herbsleb. 2013. Impression Formation in Online Peer Production: Activity Traces and Personal Profiles in GitHub. In *Proc. Conf. Computer Supported Cooperative Work (CSCW)* (San Antonio, Texas, USA). ACM Press, New York, 117–128.

[48] Courtney Miller, David Widder, Christian Kästner, and Bogdan Vasilescu. 2019. Why Do People Give Up FLOSSing? A Study of Contributor Disengagement in Open Source. In *Proc. IFIP Int'l Conf. Open Source Systems (OSS).* Springer, 116–129.

[49] Audris Mockus. 2007. Large-scale code reuse in open source software. In *Int'l Workshop on Emerging Trends in FLOSS Research and Development (FLOSS).* IEEE, 7–7.

[50] Audris Mockus, Roy T. Fielding, and James D. Herbsleb. 2002. Two Case Studies of Open Source Software Development: Apache and Mozilla. *ACM Trans. Softw. Eng. Methodol. (TOSEM)* 11, 3 (July 2002), 309–346.

[51] Evgeny Morozov. 2013. The Meme Hustler. *The Baffler* 22 (4 2013).

[52] Frank Nagle. 2018. Open source software and firm productivity. *Management Science* 65, 3 (2018), 1191–1215.

[53] Keitaro Nakasai, Hideaki Hata, and Kenichi Matsumoto. 2018. Are Donation Badges Appealing? A Case Study of Developer Responses to Eclipse Bug Reports. *IEEE Software* 36, 3 (2018), 22–27.

[54] Keitaro Nakasai, Hideaki Hata, Saya Onoue, and Kenichi Matsumoto. 2017. Analysis of Donations in the Eclipse Project. In *Proc. Int'l Workshop on Empirical Software Engineering in Practice (IWESEP).* IEEE, 18–22.

[55] Elinor Ostrom. 1990. *Governing the commons: The evolution of institutions for collective action.* Cambridge University Press.

[56] Jagdish K Patel, CH Kapadia, and Donald Bruce Owen. 1976. *Handbook of statistical distributions.* M. Dekker.

[57] Jane Ritchie, Jane Lewis, Carol McNaughton Nicholls, Rachel Ormston, et al. 2013. *Qualitative research practice: A guide for social science students and researchers.* Sage.

[58] Jeffrey A Roberts, Il-Horn Hann, and Sandra A Slaughter. 2006. Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the Apache projects. *Management Science* 52, 7 (2006), 984–999.

[59] Gregorio Robles and Jesus M Gonzalez-Barahona. 2006. Contributor turnover in libre software projects. In *Proc. IFIP Int'l Conf. Open Source Systems.* Springer, 273–286.

[60] Peter J Rousseeuw and Christophe Croux. 1993. Alternatives to the median absolute deviation. *J. Amer. Statist. Assoc.* 88, 424 (1993), 1273–1283.

[61] Wilmar B Schaufeli, Arnold B Bakker, Kees Hoogduin, Cas Schaap, and Atilla Kladler. 2001. On the clinical validity of the Maslach Burnout Inventory and the Burnout Measure. *Psychology & Health* 16, 5 (2001), 565–582.

[62] Andreas Schilling, Sven Laumer, and Tim Weitzel. 2012. Who will remain? An evaluation of actual person-job and person-team fit to predict developer retention in FLOSS projects. In *Proc. Hawaii Int'l Conf. System Sciences (HICSS).* IEEE, 3446–3455.

[63] Isaac Schlueter. 2015. Money and Open Source. https://medium.com/open-source-life/d44a1953749c Blog post.

[64] Sonali K Shah. 2006. Motivation, governance, and the viability of hybrid forms in open source software development. *Management Science* 52, 7 (2006), 1000–1014.

[65] Zed Shaw, Adam Stacoviak, and Jerod Santo. 2018. The Changelog, Episode 300: Corporate interests in open source and dev culture. https://www.changelog.com/podcast/300 Podcast.

[66] Peter Singer. 2015. *The most good you can do: How effective altruism is changing ideas about living ethically.* Yale University Press.

[67] Manuel Sojer and Joachim Henkel. 2010. Code reuse in open source software development: Quantitative evidence, drivers, and impediments. *Journal of the Association for Information Systems* 11, 12 (2010), 868–901.

[68] André Staltz. 2019. Software below the poverty line. https://staltz.com/software-below-the-poverty-line.html. Blog post.

[69] Asher Trockman, Shurui Zhou, Christian Kästner, and Bogdan Vasilescu. 2018. Adding Sparkle to Social Coding: An Empirical Study of Repository Badges in the npm Ecosystem. In *Proc. Int'l Conf. Software Engineering (ICSE).* ACM, 511–522.

[70] Rebecca Turner. 2017. npm, the npm github issue tracker, and you! https://blog.npmjs.org/post/161832149430/npm-the-npm-github-issue-tracker-and-you Blog post.

[71] Marat Valiev, Bogdan Vasilescu, and James Herbsleb. 2018. Ecosystem-Level Determinants of Sustained Activity in Open-Source Projects: A Case Study of the PyPI Ecosystem. In *Proc. Int'l Symposium Foundations of Software Engineering*

(FSE). ACM Press, 644–655.

[72] Michael R Veall and Klaus F Zimmermann. 1996. Pseudo-R2 measures for some common limited dependent variable models. *Journal of Economic Surveys* 10, 3 (1996), 241–259.

[73] Georg Von Krogh, Stefan Haefliger, Sebastian Spaeth, and Martin W Wallin. 2012. Carrots and rainbows: Motivation and social practice in open source software development. *Mis Quarterly* 36, 2 (2012), 649–676.

[74] Georg Von Krogh and Eric Von Hippel. 2003. Special Issue on Open Source Software Development (Editorial). *Research Policy* 32 (2003), 1149–1157.

[75] Joel West and Scott Gallagher. 2006. Challenges of Open Innovation: The Paradox of Firm Investment in Open-Source Software. *R&D Management* 36, 3 (2006), 319–331.

[76] Tim Wood. 2016. moment().endOf('term'). https://medium.com/timrwood/moment-endof-term-522d8965689 Blog post.

[77] Yiqing Yu, Alexander Benlian, and Thomas Hess. 2012. An empirical study of volunteer members' perceived turnover in open source software projects. In *Proc. Hawaii Int'l Conf. System Sciences (HICSS)*. IEEE, 3396–3405.

[78] Yangyang Zhao, Yuming Zhou, Alexander Serebrenik, Vladimir Filkov, and Bogdan Vasilescu. 2017. The Impact of Continuous Integration on Other Software Development Practices: A Large-Scale Empirical Study. In *Proc. Int'l Conf. Automated Software Engineering (ASE)*. IEEE, 60–71.

[79] Minghui Zhou, Audris Mockus, Xiujuan Ma, Lu Zhang, and Hong Mei. 2016. Inflow and Retention in OSS Communities with Commercial Involvement: A Case Study of Three Hybrid Projects. *ACM Trans. Softw. Eng. Methodol. (TOSEM)* 25, 2 (2016), 13.

[80] Henry Zhu and Jeff Meyerson. 2018. Software Engineering Daily: Babel with Henry Zhu. https://softwareengineeringdaily.com/2018/06/21/babel-with-henry-zhu/ Podcast.

[81] Devon Zuegel. 2019. Announcing GitHub Sponsors: a new way to contribute to open source. https://github.blog/2019-05-23-announcing-github-sponsors-a-new-way-to-contribute-to-open-source/. Accessed May 23, 2019.