L'Identification des Projets de Logiciel Libre Accessibles aux Nouveaux Contributeurs

Paul Hervot[1] et Benoît Crespin[2]

[1] Laboratoire de Recherche de L'EPITA (LRE), 14-16 rue Voltaire, 94270 Le Kremlin-Bicêtre, France
[2] Université de Limoges, XLIM/ASALI, UMR CNRS 7252, France

Résumé Le logiciel libre prend de plus en plus de place dans le paysage public et industriel, notamment pour sa transparence et sa gestion démocratique, pour autant publier le code source d'un logiciel ne le rend pas automatiquement accessible, et les nouveaux contributeurs de ces projets rencontrent de nombreuses barrières les entravant dans leurs contributions. Au travers d'une analyse à grande échelle de l'archive de Software Heritage, nous testons la pertinence de trois indicateurs dans l'identification des logiciels libres accessibles aux nouveaux contributeurs. Nos résultats montrent une corrélation positive entre le nombre de premières contributions réussies dans un projet et la présence d'instructions de contribution, ainsi qu'entre ce même nombre et celui des contributeurs uniques récents du projet. Ces indicateurs trouveront une utilité dans l'enseignement des pratiques du logiciel libre pour aider les enseignants à sélectionner des projets accessibles pour leurs étudiants.

Mots-clé : Collecte, traitement et analyse des traces d'apprentissage · Analyse d'usage(s) et de pratiques · Logiciel libre · Analyse automatique de dépôts logiciels · Barrières d'entrée du logiciel libre

Abstract. FOSS makes an increasing amount of the public and industrial software landscape, notably for its transparency and democratic governance. However, simply publishing the source code of a software does not automatically make it accessible, and many barriers impede new contributors approaching these projects. Through a large-scale software mining of the Software Heritage archive, we test the pertinence of three signals in the identification of accessible FOSS projects for new contributors. Our results show a positive correlation between the number of new contributors of a project successfully bringing their contribution to completion and the presence of contributing guidelines, as well as between that same number and the number of recent unique contributors in the project. Such signals could find a use in the teaching of FOSS practices, helping teachers to select accessible projects for their students.

Keywords: Collection, processing and analysis of learning traces · Usage and practices analysis · FOSS · Mining Software Repositories · Open source barriers to entry

1 Introduction

Nous nous intéressons dans cet article aux indicateurs permettant d'identifier les logiciels libres les plus accessibles aux nouveaux contributeurs, de façon à aider les enseignants à sélectionner des projets sur lesquels faire travailler leurs étudiants. Nous présentons dans la section suivante les travaux existants autour de la notion de nouveaux contributeurs. Après avoir décrit rapidement l'archive de Software Heritage choisie pour notre étude, nous testons la pertinence de trois indicateurs dans l'identification des logiciels libres accessibles.

2 Nouveaux contributeurs

Mendez et al. [4] notent la difficulté des nouveaux volontaires à rejoindre une communauté de logiciel libre, citant comme exemple extrême le projet Apache Hadoop qui a vu 82% de ses nouveaux volontaires quitter le projet après leur première contribution [12]. Plusieurs mesures *a posteriori* de l'accessibilité des projets de logiciel libre ont été faites, en majorité de façon qualitative [11, 9]. Une approche quantitative encore rare consiste à étudier la progression du nombre total de contributeurs au cours de la vie du projet [2]. Similairement, Qiu et al. [6] étudient les signaux que les potentiels nouveaux contributeurs observent pour choisir un projet auquel contribuer. Ils s'intéressent pour cela spécifiquement aux projets disponibles publiquement sur GitHub, ce qui leur permet d'étudier empiriquement l'effet des signaux que la plateforme met en évidence. Ils suggèrent qu'une mesure possible de l'accessibilité pour les nouveaux contributeurs (« *newcomers openness* » dans l'article) est le pourcentage de *pull requests* créées par des contributeurs externes. Pour leur analyse quantitative, Qiu et al. [6] collectent leurs données via l'*API* de GitHub, mais les limites de cette approche font l'objet d'une littérature grandissante et remettant en question certains résultats [1, 13]. Steinmacher et al. [10] remarquent de plus que dans la littérature, les études se concentrent trop sur les projets importants et matures. À l'inverse, 71% des projets hébergés sur GitHub sont personnels et non réellement collaboratifs, Kalliamvakou et al. [1] ne retiennent donc quant à eux que ceux ayant au moins deux auteurs uniques différents.

3 L'archive de logiciels de Software Heritage

Software Heritage est une initiative exploitant des techniques avancées de compression de graphe afin de construire une archive aussi complète que possible du code source actuellement disponible publiquement dans le monde. L'archive est publique et comptait en 2018 plus d'un milliard de *commits* uniques archivés depuis 85 millions d'origines différentes. Elle s'étoffe continuellement à mesure que des origines y sont ajoutées ou revisitées, ce qui en fait l'un des corpus les plus complets et exploitables par la recherche scientifique [5, 8].

4 Méthodologie

Qiu et al. [6] se sont intéressés aux signaux que les potentiels nouveaux contributeurs observent pour choisir un projet auquel *essayer* de contribuer, nous proposons de déterminer si certains de ces signaux sont de surcroit prédictifs d'une réelle accessibilité de ces projets, c'est à dire à quel point de nouveaux contributeurs *réussissent* à produire une contribution apparaissant dans l'historique de développement du projet. Pour dépasser les limitations de GitHub, nous utiliserons l'archive de Software Heritage comme accès à la population étudiée. Ce choix nous empêchera cependant d'étudier les indicateurs propres à GitHub comme le nombre de *pull requests* d'un projet. Pour évaluer l'accessibilité d'un projet, nous proposons d'utiliser comme variable proxy le nombre de contributeurs apparaissant pour la première fois dans l'historique de développement du projet entre le premier juin 2019 et le premier septembre 2019 [6]. Nous considérons comme « récent » tout événement survenu dans les six mois avant la période de référence étudiée.

Hypothèse 01 *: les projets possédant des instructions de contribution sont plus accessibles pour les nouveaux contributeurs que ceux n'en ayant pas.*

Hypothèse 02 *: le nombre de contributeurs uniques récents d'un projet est positivement corrélé à son accessibilité pour les nouveaux contributeurs.*

Hypothèse 03 *: le nombre de* commits *récents au sein d'un projet est positivement corrélé à son accessibilité pour les nouveaux contributeurs.*

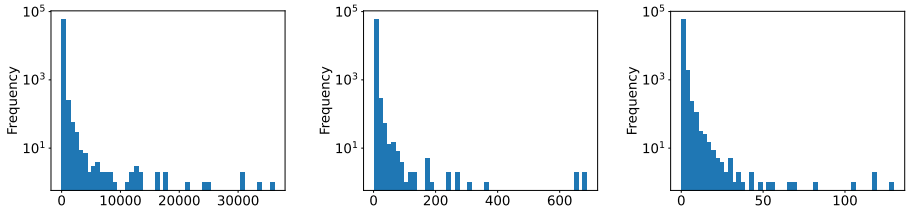
Sont exclus de l'échantillon les projets : 1. qui sont des *forks* d'un autre (en cas de *commits* en commun, seul le projet qui la plus longue chaîne de *commits* a été retenu comme représentant du groupe) ; 2. n'ayant reçu aucun *commit* pendant la période de référence (inactivité) [1] et 3. ayant moins de deux contributeurs uniques récents (projet personnel non-collaboratif) [1].

Nous effectuons un premier parcours du graphe à partir de chaque origine afin d'identifier le représentant de chaque groupe de fork. Pour le *snapshot* le plus récent de chaque origine ainsi retenue, un nouveau parcours est effectué à partir de sa branche principale afin de récolter les données de recherche. La présence d'instructions de contribution en particulier est difficile à vérifier car le graphe ne contient que les noms et la hiérarchie des fichiers d'un projet, pas leur contenu. Si un fichier `CONTRIBUTING.md` ou assimilé existe, nous validons la présence d'instructions de contribution, sinon, nous cherchons un fichier `README.md` ou assimilé. En l'absence d'un tel fichier, nous concluons à l'*absence* d'instructions de contribution, en sa présence, nous sauvegardons l'identifiant unique du fichier afin d'analyser son contenu ultérieurement. Celui-ci peut être téléchargé publiquement en HTTP sur le site archive.softwareheritage.org ou depuis le *registry* Amazon S3 software-heritage. Le site limite le débit des requêtes autorisées, contrairement au *registry*, mais ce dernier est incomplet. Nous téléchargeons donc en priorité via le *registry* Amazon S3 et nous nous rabattons sur le site pour les fichiers manquants. Nous cherchons ensuite dans le `README` une section contenant le mot *contributing* ou une expression proche. Les noms de sections sont identifiés

en supposant que les `README` sont formatés en Markdown ou reStructuredText [3, 7]. Enfin, le nombre de nouveaux contributeurs est calculé en comptant les contributeurs uniques de la période de référence qui n'apparaissent dans aucun *commit* antérieur à cette période, et le nombre de contributeurs uniques récents en les comptant simplement dans la période de six mois précédent la période de référence.

Un *replication package* contenant le code source et les bibliothèques utilisés pour l'étude présentée dans notre article, ainsi que les données brutes collectées aux différentes étapes, est disponible sur Zenodo : zenodo.org/record/7888415.

5 Résultats et discussion

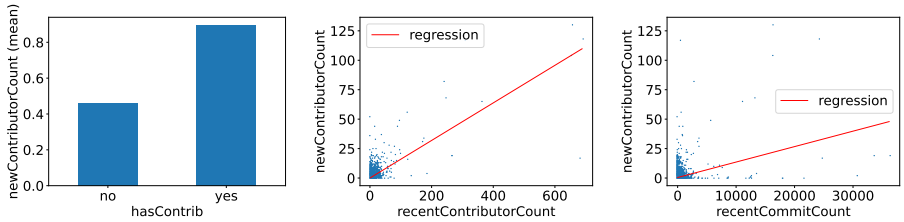


(a) *Commits* récents (b) Contributeurs récents (c) Nouveaux contributeurs

Fig. 1 – Distributions (avec ordonnée logarithmique) des données étudiées

Sur les 60966 projets distincts (sans historique commun) analysés, 14% d'entre eux possèdent des instructions de contribution. Les nombres de *commits* récents, de contributeurs récents et de nouveaux contributeurs varient fortement au sein de cette population, la distribution n'est normale dans aucun des cas (voir Fig. 1). Les projets possédant des instructions de contribution ont en moyenne plus de nouveaux contributeurs (voir Fig. 2a) avec une taille d'effet $\rho \approx 0.57$ (test MWW), ce qui confirme ($p \approx 0$) que les deux distributions sont bien différentes et valide l'hypothèse H01. De futures recherches pourraient essayer de déterminer si ce plus grand nombre de nouveaux contributeurs *ayant réussi* à contribuer est dû uniquement à ce plus grand nombre de nouveaux contributeurs *essayant* de contribuer, ou si la présence d'instructions de contribution a un réel rôle dans le succès d'une tentative de contribution. Notons tout de même que la robustesse du test MWW diminue significativement dans une situation comme la nôtre (distribution non-normale et variances significativement différentes entre les deux catégories) [14].

Une régression GLS suggère que plus le nombre de contributeurs récents d'un projet est élevé, plus son nombre de nouveaux contributeurs l'est aussi (voir Fig. 2b). Le coefficient de détermination $R^2 \approx 0.45$ du modèle indique que le nombre de contributeurs récents explique environ 45% de la variation du nombre



(a) Présence d'instructions de contribution

(b) Nombre de contributeurs récents

(c) Nombre de *commits* récents

Fig. 2 – Nombre de nouveaux contributeurs en fonction des variables mesurées.

de nouveaux contributeurs, ce qui valide l'hypothèse H02, avec une taille d'effet tout de même modérée. Suivant la même approche pour l'analyse du nombre de *commits* récents, une régression GLS suggère ici aussi une corrélation positive entre le nombre de *commits* récents d'un projet et son nombre de nouveaux contributeurs (voir Fig. 2c), mais son coefficient de détermination est trop faible ($R^2 \approx 0.10$) pour considérer que le modèle représente fidèlement des données du problème. Nous ne pouvons donc valider l'hypothèse H03.

6 Conclusion

L'analyse de l'archive de Software Heritage nous a permis de trouver une corrélation positive entre le nombre de nouveaux contributeurs d'un projet de logiciel libre et deux indicateurs facilement observables pour les aspirants contributeurs ou les enseignants souhaitant faire travailler leurs étudiants sur ces problématiques : la présence d'instructions de contribution au sein d'un projet et le nombre de contributeurs uniques ayant récemment contribué au projet. Nous n'avons en revanche pas trouvé de lien avec le nombre de *commits* récents des projets. Restent par ailleurs à identifier les liens de causalité expliquant nos observations, ainsi que la mise en situation de ces indicateurs dans un contexte pédagogique.

Références

1. E. Kalliamvakou et al. The Promises and Perils of Mining GitHub. In : *Proceedings of the 11th Working Conference on Mining Software Repositories*. MSR 2014. Hyderabad, India : Association for Computing Machinery, 2014, p. 92-101. DOI : 10.1145/2597073.2597074.
2. J. Lerner, P. A. Pathak et J. Tirole. The Dynamics of Open-Source Contributors. In : *American Economic Review* 96.2 (mai 2006), p. 114-118. DOI : 10.1257/000282806777211874.
3. Markdown Basic Syntax - Headings. URL : www.markdownguide.org/basic-syntax/#headings.

4. C. Mendez et al. Open Source Barriers to Entry, Revisited : A Sociotechnical Perspective. In : *Proceedings of the 40th International Conference on Software Engineering*. ICSE '18. Gothenburg, Sweden : Association for Computing Machinery, 2018, p. 1004-1015. DOI : 10.1145/3180155.3180241.
5. A. Pietri, D. Spinellis et S. Zacchiroli. The Software Heritage Graph Dataset : Public Software Development Under One Roof. In : *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. 2019, p. 138-142. DOI : 10.1109/MSR.2019.00030.
6. H. S. Qiu et al. The Signals That Potential Contributors Look for When Choosing Open-Source Projects. In : *Proc. ACM Hum.-Comput. Interact.* 3.CSCW (nov. 2019). DOI : 10.1145/3359224.
7. reStructuredText Syntax - Sections. URL : docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#sections.
8. G. Rousseau, R. Di Cosmo et S. Zacchiroli. Growth and Duplication of Public Source Code over Time : Provenance Tracking at Scale. working paper or preprint. Juin 2019. URL : hal.archives-ouvertes.fr/hal-02158292.
9. C. Stanik et al. A Simple NLP-Based Approach to Support Onboarding and Retention in Open Source Communities. In : *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 2018, p. 172-182. DOI : 10.1109/ICSME.2018.00027.
10. I. Steinmacher et al. A systematic literature review on the barriers faced by newcomers to open source software projects. In : *Information and Software Technology* 59 (2015), p. 67-85. DOI : 10.1016/j.infsof.2014.11.001.
11. I. Steinmacher et al. Overcoming Open Source Project Entry Barriers with a Portal for Newcomers. In : *Proceedings of the 38th International Conference on Software Engineering*. ICSE '16. Austin, Texas : Association for Computing Machinery, 2016, p. 273-284. DOI : 10.1145/2884781.2884806.
12. I. Steinmacher et al. Why do newcomers abandon open source software projects ? In : *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. 2013, p. 25-32. DOI : 10.1109/CHASE.2013.6614728.
13. M. Z. Trujillo, L. Hébert-Dufresne et J. Bagrow. The penumbra of open source : projects outside of centralized platforms are longer maintained, more academic and more collaborative. In : *EPJ Data Science* 11.1 (2022), p. 31.
14. D. W. Zimmerman. Invalidation of Parametric and Nonparametric Statistical Tests by Concurrent Violation of Two Assumptions. In : *The Journal of Experimental Education* 67.1 (1998), p. 55-68. DOI : 10.1080/00220979809598344.