Cheating Death: A Statistical Survival Analysis of Publicly Available Python Projects

Rao Hamza Ali, Chelsea Parlett-Pelleriti, Erik Linstead
{raali,cparlett,linstead}@chapman.edu
Machine Learning and Assistive Technology Lab
Chapman University
Orange, CA, USA

ABSTRACT

We apply survival analysis methods to a dataset of publicly-available software projects in order to examine the attributes that might lead to their inactivity over time. We ran a Kaplan-Meier analysis and fit a Cox Proportional-Hazards model to a subset of Software Heritage Graph Dataset, consisting of 3052 popular Python projects hosted on GitLab/GitHub, Debian, and PyPI, over a period of 165 months. We show that projects with repositories on multiple hosting services, a timeline of publishing major releases, and a good network of developers, remain healthy over time and should be worthy of the effort put in by developers and contributors.

CCS CONCEPTS

• **Software and its engineering** → *Collaboration in software development*.

KEYWORDS

open source software projects, survival analysis, software repository health, hazard ratios

ACM Reference Format:
Rao Hamza Ali, Chelsea Parlett-Pelleriti, Erik Linstead . 2020. Cheating Death: A Statistical Survival Analysis of Publicly Available Python Projects. In *17th International Conference on Mining Software Repositories (MSR '20), October 5–6, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3379597.3387511

1 INTRODUCTION

Open Source Software (OSS) projects are ubiquitous in today's software landscape and provide a rich set of data on which to analyze facets of the software development process using everything from traditional statistics to deep learning [5, 11, 15–17]. They are unique in that they allow developers to volunteer their time and effort into creating software that is open for all to use. While open source development efforts typically have a single person or body that selects a subset of developed code for build releases and makes it available for distribution [4], these projects are maintained by a decentralized team of developers, who with low organizational cost, are able to produce applications that are at times used by millions. The decentralized nature of the teams piques the interest of many programmers who end up contributing to these projects. There are no weekly meetings, developers rarely meet face-to-face, people undertake the work of their choice, and the geographical diversity of all contributors is immense, yet, there is a clear weekly pattern of code update and addition[3]. This results in a software development process that is substantially different from industry-level processes and potentially allows more creative and innovative practices to emerge. In a survey, 72% of participants said that they always seek out open source options when evaluating new tools [1].

Lucassen et al. define the health of a software ecosystem as *"longevity and a propensity for growth."*[12] Every healthy open source project needs a team of dedicated developers and a set timeline of goals and achievements. These projects also need to be popular enough to gain interest from potential volunteers. It is hard to predict the health of an open source project at the time of its inception, when developers are excited about the project and the end goals. But it is possible to see how a project has performed over time. The health of a project could be computed by the number and frequency of contributions, how frequently big targets are met by the developers, or how focused the team is on making the software ready for distribution. Since developers work on these projects as volunteers, they want to ensure that their contributions do not go into a project that might end up inactive. If this knowledge were available to volunteers beforehand, they could consider other avenues before dedicating their efforts to a single project.

Here we are mainly interested in the health of a project from an additions-made point of view, since every new code addition made to a project repository means that the team is meeting its goals. We also want to look at the number of volunteers that work on a project, the timeline they work in, and the number of version control systems (VCS) they use to host their projects. Having a project on multiple VCSs or repositories like PyPI or Debian highlights the accessibility of the project and points to the seriousness of the developers and the team working on it. We use survival analysis, commonly used in medical studies to predict treatment efficacy, to find the probability of survival of popular open source projects over time using Kaplan-Meier survival analysis, and quantify the effects of these variables using a Cox Proportional-Hazards model.

2 DATA

An analysis of this nature is only possible with a dataset that records repositories for projects on common VCSs in their entirety along with a history of all commits (referred to as revisions) and major

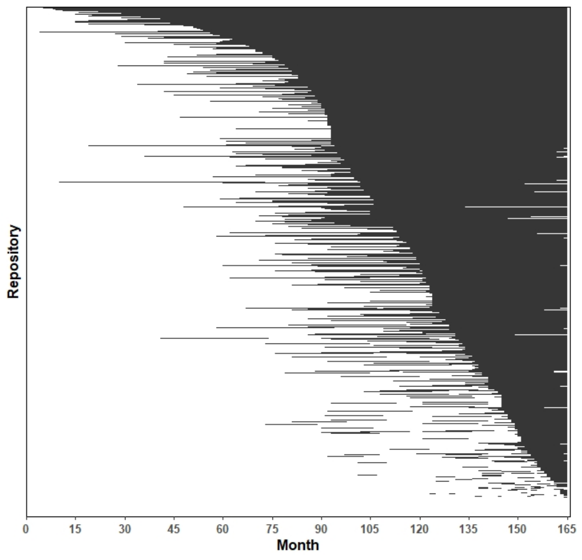

Figure 1: Duration of all projects

releases (noteworthy revisions with a specific name like a version number or release date). The *popular-3k-python* subset of the Software Heritage graph dataset [18] is used for this analysis. The dataset includes snapshots of nearly 3000 popular projects, between 2005 and 2018, hosted on GitLab, GitHub, Debian, and PyPI. These projects were tagged with using Python as the main programming language. Analyzing individual repositories, we are able to discern exactly how many times a revision is made to a project as well as cross-reference a project's repositories hosted on multiple VCSs. Overall, we extract the timestamp and author identifier for each revision and release made on a project, and the VCS used to host the project repository.

For survival analysis, we first need to establish a timeline within which we analyze the health of the projects. Our proposed timeline spans 14 years or 165 months (we define a month to be 4 weeks), beginning in 2005 and ending in January 2018. Because Software Heritage collected multiple snapshots for each project, across several months, there is a lot of variability in the recency of the latest snapshot for each project. To ensure that we look at the same duration for all projects, we use a single cutoff date in January 2018. Figure 1 shows the duration of all projects ordered by longest to shortest duration, within the time duration. The cut off date also ensures that projects that started during 2018 will get discarded from the study, since they did not have enough time to establish their timeline of revisions. After removing such projects and those that only had a single instance in their history, we end up with 2059 projects and extract all their historical data for the duration of 165 months. No outliers were excluded.

3 METHOD

Survival analysis is a statistical methodology used in biostatistics to study the duration of the life of an entity [21]. The approach is based on measurements of events that can occur at any time during a study. The data used for survival analysis includes the time until an event of interest occurs. For example, survival analysis can be used to model time until tumor recurrence, death after a treatment intervention, or presence of symptoms in patients. Towards the application of survival analysis to OSS development, Lin et al. [10] and Ortega et al. [14] defined the event of interest as developers who stopped contributing after some time, and used it to study the effects of developers dropping out on the health of a project. Aman et al. [2] used commits by new developers as their event, to analyze the effects of introduction of buggy code to a software repository. For this study, we define the event of interest as the event of repository abandonment or complete lack of activity.

An important aspect of survival analysis is censoring. During the time all projects are observed, if inactivity, as the event of interest, does not occur, then we only know the total number of months in which the event did not occur. In other words, the exact time-to-event is censored. To determine which projects should be censored, Samoladas et al. [19] used a month-by-month analysis to check activity of each project. If a project had 2 months of consecutive inactivity, it was deemed abandoned. But this approach resulted in a very small subset of projects for which the event of inactivity had occurred, and a large share of inactive projects for the study came from a different approach. Instead, we use the approach used by Evangelopoulos et al. [9] where a project is deemed abandoned if there is no activity at all. For our study, a project that has revisions beyond the January 2018 cutoff date surely is active and is deemed censored, since the time-to-event of inactivity is not observed during the 165 months period. And the remaining projects, that suddenly showed no activity (no new revisions or releases published) by the end of the time duration, become inactive. This form of censoring is called Type III censoring (commonly referred to as random censoring) and allows for staggered start times for various projects. Avelino et al. [4] describe random censoring as the most common case in software project research. The period of study is predefined and projects start at different times during that period, as can be seen in Figure 1. We note that the dataset contains more active projects than inactive projects for the time duration.

The Kaplan-Meier (K-M) survival estimator is an important tool to analyze and compare survival probabilities. It is a nonparametric estimation technique and a widely used method for estimating the survival function, in the presence of censored values [8], where the survival function is the probability that the duration of a project is longer than time t [19]. The K-M estimator produces a curve which approaches the true survival function for the data. This allows us to compare survival probabilities of OSS projects, with different attributes, even though some data is censored. While the K-M curves give us a visual representation of the survival of projects with varying attributes over time, the Cox Proportional-Hazards model allows us to fit a regression model to investigate the association between the health of projects and key project attributes. There are various parametric models available for modeling the relationship of duration with other attributes but the Cox Proportional-Hazards model allows estimation of effect parameters without consideration of the hazard function, which describes how the risk of event occurring changes over time [6]. We apply both the K-M estimator and the Cox Proportional-Hazards model on the data to estimate the effects of attributes on the overall health of an open source project.

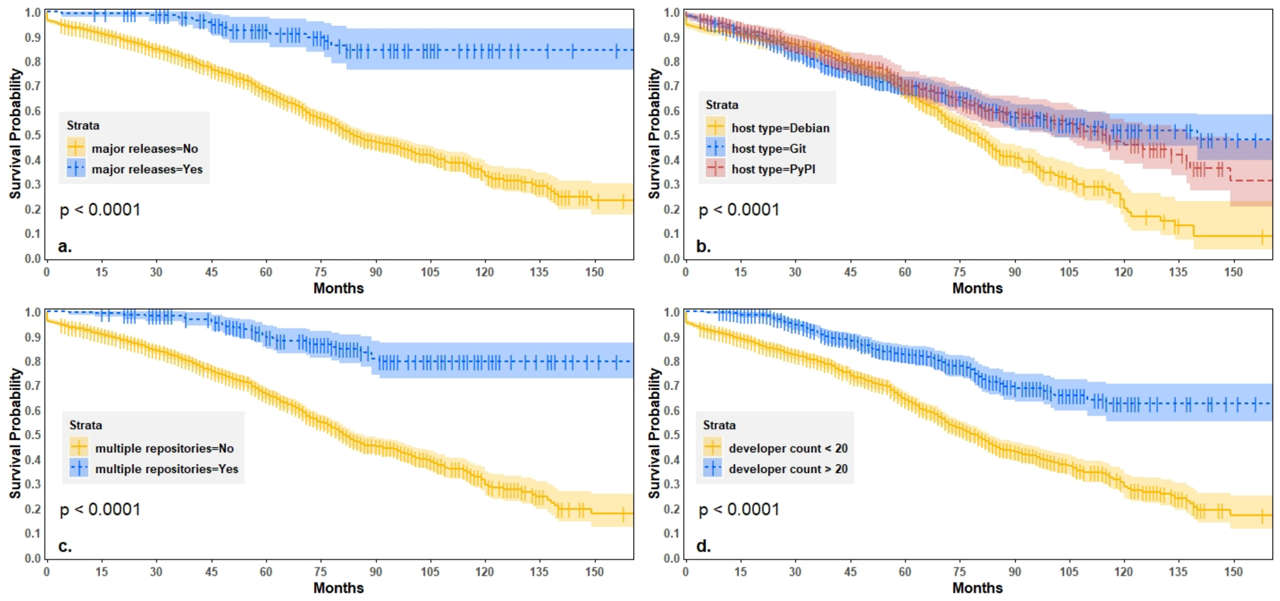

Figure 2: Kaplan-Meier estimations of the survival functions when comparing projects based on selected attributes

4 RESULTS

We use the following attributes of a project as estimators of survival rate over time:

- `majorReleases` whether releases were published by the project developers
- `hostType` type of hosting service used for the project repository
- `authorCount` total unique developers that have committed a revision to the repository
- `multipleRepositories` whether the project is hosted on multiple version control systems

We now discuss the results of running K-M curves and fitting Cox Proportional-Hazard models on the data.

4.1 Kaplan-Meier Survival Curves

We generate separate K-M curves, with confidence intervals, for all four factored attributes described above. Figure 2 shows the survival probabilities of projects when grouped on the categorical values of each attribute. We also show the p-values from the log-rank test, with results indicating that each set of projects is significantly different in terms of survival, for each attribute tested. Just 5.73% of projects published at least one release; a revision that consolidates several commits and brings major updates to the project, and we observe that having at least one release significantly increased the chances of a project's survival at the end of the 165 month period, as seen in Figure 2a. Official releases are a way to show that significant changes and additions have been made to a project, so much so that they can be consolidated as a single version of the software. It also signifies that developers are meeting targets and the project will continue to remain active. The right tail of the curve for projects that had releases plateaus at 85% survival probability around the 80 month mark implying the presence of long-term survivors [7], while projects with zero releases end up with below 30% survival rate at the end of the study duration.

Figure 2b implies the significance of using different hosting services for open source projects. GitHub is the largest social coding platform where all commits, issues, code changes, and requests are archived publicly [25], and has become the standard for OSS development. This attract a lot of developers to use GitHub as the host for their repositories, while Debian and PyPI are package repositories where developers host their projects for distribution. Projects hosted on GitHub come out on top in the survival race compared to the other two hosting services. Though it should be noted that both Debian and PyPI based projects have a higher survival rate during the first 55 months, which is within the average duration of projects hosted on all three services (50-57 months). When a project is ready for distribution, specifically for the Debian operating system or as a library for Python, it is traditionally added to the Debian and PyPI package repositories. This implies that an open source project has made significant developments and is very active. But we see that once the average duration threshold is passed, both sets of projects, hosted on Debian and PyPI significantly drop in survival probability. Figure 2c signifies this fact. We see that projects that do host their repositories on multiple services, for distribution purposes, have a very healthy survival rate of 80% compared to below 20% for projects that use only one package repository system, by the end of the analysis duration.

Publishing major releases or distributing the project on different repositories is a nod to the healthy development cycle for an open source project. If a project has version releases, then it can be viable for distribution across different systems. And we see that this characteristic of a project significantly increases its probability of survival over time. But a very important component of a healthy open source project is the network of developers, that actually produces these releases and distributions. Even when a project is

Table 1: Hazards Ratio of categorical attributes of projects with count and confidence intervals for ratios

Attribute	Value	N	Ratio
releases	Yes	118	reference (1)
	No	1941	3.00 (1.70-5.31)***
multiple repos.	Yes	188	reference
	No	1871	3.30 (2.24-4.87)***
developers > 20	Yes	535	reference
	No	1524	5.95 (4.53-7.82)***
hosting service	Git	734	reference
	PyPI	368	0.21 (0.16-0.28)***
	Debian	957	0.28 (0.22-0.34)***

*** *p < 0.001*

popular, if lead developers leave the project, it becomes in danger of getting abandoned, as analyzed by Sentas et al. [21]. So it is important that major development changes are not maintained by a select few developers. In our data set, 28.8% projects had revisions made by just one developer. If that developer were to leave the project, it could pose serious problems for the project to continue to remain active. The K-M curve in figure 2d highlights this issue where we show survival rate of projects that have had at least 20 different committers (to revisions) versus those that do not. Twenty was chosen as a threshold because that number will include the core group of developers as well as a small secondary group. It becomes apparent that a good network of developers, where the task to make commits and revisions is shared between many developers, end up having a survival rate of above 65% which is significantly different from projects that maintain a small team of lead developers, with around 20% survival rate after 165 months.

4.2 Cox Proportional-Hazards Model

We fit the categorical attributes that we used in the Kaplan-Meier analysis to a Cox Proportional-Hazards model to estimate the effect of these attributes on the health of open source projects. We use projects that published major releases, had repositories on multiple hosting services, had more than 20 unique developers who published revisions, and used GitHub/GitLab as their main hosting service as the baseline. This baseline acts as the control or reference group for the hazard ratio [22].

Table 1 gives the hazards ratios for each attribute. We show that all ratios are statistically significant and that projects with no releases are 3 times more likely to become inactive compared to those that do have periodic releases. Similarly, projects with a single hosted repository are 3.3 times more likely to become inactive than those with multiple repositories. As seen with the K-M curve in Figure 2 where we show a significant difference between projects when comparing the number of unique developers who made revisions, projects with that count below 20 are 5.95 times more likely to become abandoned. For the type of hosting service used, we see that projects hosting their repositories on PyPI or Debian are less likely to be abandoned compared to those that are hosted on GitHub/GitLab. This observation goes along with the insight that developers will only host repositories on PyPI and Debian when they are ready to distribute their software and have thus made significant progress with the software development.

The two survival analysis techniques applied to the data highlights the importance of routine updates and a good network of developers for an OSS project. It is possible that these characteristics are associated with unseen variables such as private funding for a project or high demand for a software tool. It is also possible that developers are paid to work on the project, exceeding the 20 unique author count that we used for our analysis, and that a project becomes abandoned because of a sudden cut in funding. However, this analysis shows a clear association between the chances of a project's survival and diversity of developers and routine releases. While survival analysis is commonly used in medical studies, to estimate survival of living beings, open source projects too act like organisms, which remain healthy when all components work well and continue to nourish and grow as time passes. But neglect, in the form of low revision rate and too much reliance on a small group of developers, is associated with inactivity over time.

5 RELATED WORK

Health analysis of software repositories using survival analysis techniques is relatively new. Schweik et al. [20] applied logistic regression on open source projects to find that adding more developers raised the chances of a project to be successful. Subramaniam et al. [23], who investigated the longitudinal effects of project specific characteristics on the success of open source projects, found that developer interest and project activity affected the project success measures. Our work explores these findings further and uses survival analysis to not only emphasize the role of these attributes to an open source project over time, but also estimates the measure of the effect of what VCS are used by the projects.

For related work that uses survival analysis for estimating health of open source projects, Miller et al. [13] fit a hazard model to estimate the effects of covariates on disengagement of developers and found that long working hours and job transitions increased the chances of disengagement from an open source project. Valiev et al. [24] did a study on comparing different attributes of OSS projects on long term activity, along with interview insights, and were able to conclude the positive effect of organizational support, and coding competitions, on the health of open source projects. While they were not able to measure the effect of the size of the core team, we find that having a good diversity of developers who author revisions significantly increases the survival rate of projects.

6 CONCLUSION

Our research aims to investigate the effects of attributes of open source project to its overall health using survival analysis. We find that while the percentage of projects that do have major releases is relatively low, the survival rate for such projects was significantly higher than the projects that did not publish noteworthy revisions. We also show the effect of using multiple hosting repositories as well as the type of service used on the projects in the long run. Finally, using Cox Proportional-Hazards model, we estimate that projects with a small core team is around six times more likely to become inactive compared to those that boast a diverse set of core team developers.

REFERENCES

[1] 2017. *Open Source Survey.* Retrieved February 2, 2020 from https://opensourcesurvey.org/2017/

[2] Hirohisa Aman, Sousuke Amasaki, Tomoyuki Yokogawa, and Minoru Kawahara. 2017. A survival analysis of source files modified by new developers. In *International Conference on Product-Focused Software Process Improvement.* Springer, 80–88.

[3] Abigail Atchison, Christina Berardi, Natalie Best, Elizabeth Stevens, and Erik Linstead. 2017. A time series analysis of TravisTorrent builds: to everything there is a season. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR).* IEEE, 463–466. https://doi.org/10.1109/MSR.2017.29

[4] Guilherme Avelino, Eleni Constantinou, Marco Valente, and Alexander Serebrenik. 2019. On the abandonment and survival of open source projects: An empirical investigation. *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 1–12. https://doi.org/10.1109/ESEM.2019.8870181

[5] Natalie Best, Jordan Ott, and Erik Linstead. 2020. Exploring the Efficacy of Transfer Learning in Mining Image-Based Software Artifacts. *arXiv preprint arXiv:2003.01627* (2020).

[6] David Roxbee Cox. 2018. *Analysis of survival data.* Chapman and Hall/CRC. https://doi.org/10.1002/bimj.4710290119

[7] Vera Damuzzo, Laura Agnoletto, Luca Leonardi, Marco Chiumente, Daniele Mengato, and Andrea Messori Messori. 2019. Analysis of survival curves: statistical methods accounting for the presence of long-term survivors. *Frontiers in oncology* 9 (2019), 453. https://doi.org/10.3389/fonc.2019.00453

[8] Bradley Efron. 1988. Logistic regression, survival analysis, and the Kaplan-Meier curve. *Journal of the American statistical Association* 83, 402 (1988), 414–425.

[9] Nicholas Evangelopoulos, Anna Sidorova, Stergios Fotopoulos, and Indushobha Chengalur-Smith. 2008. Determining process death based on censored activity data. *Communications in Statistics—Simulation and Computation®* 37, 8 (2008), 1647–1662. https://doi.org/10.1080/03610910802140224

[10] Bin Lin, Gregorio Robles, and Alexander Serebrenik. 2017. Developer turnover in global, industrial open source projects: Insights from applying survival analysis. In *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE).* IEEE, 66–75. https://doi.org/10.1109/ICGSE.2017.11

[11] Erik Linstead, Lindsey Hughes, Cristina Lopes, and Pierre Baldi. 2009. Exploring Java software vocabulary: A search and mining perspective. In *2009 ICSE Workshop on Search-Driven Development-Users, Infrastructure, Tools and Evaluation.* IEEE, 29–32.

[12] Garm Lucassen, Kevin Van Rooij, and Slinger Jansen. 2013. Ecosystem health of cloud PaaS providers. In *International Conference of Software Business.* Springer, 183–194. https://doi.org/10.1007/978-3-642-39336-5_18

[13] Courtney Miller, David Gray Widder, Christian Kästner, and Bogdan Vasilescu. 2019. Why do people give up flossing? A study of contributor disengagement in open source. In *IFIP International Conference on Open Source Systems.* Springer, 116–129. https://doi.org/10.1007/978-3-030-20883-7_11

[14] Felipe Ortega and Daniel Izquierdo-Cortazar. 2009. Survival analysis in open development projects. In *2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development.* IEEE, 7–12. https://doi.org/10.1109/FLOSS.2009.5071353

[15] Jordan Ott, Abigail Atchison, Paul Harnack, Adrienne Bergh, and Erik Linstead. 2018. A deep learning approach to identifying source code in images and video. In *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR).* IEEE, 376–386.

[16] Jordan Ott, Abigail Atchison, Paul Harnack, Natalie Best, Haley Anderson, Cristiano Firmani, and Erik Linstead. 2018. Learning lexical features of programming languages from imagery using convolutional neural networks. In *Proceedings of the 26th Conference on Program Comprehension.* 336–339.

[17] Jordan Ott, Abigail Atchison, and Erik J Linstead. 2019. Exploring the applicability of low-shot learning in mining software repositories. *Journal of Big Data* 6, 1 (2019), 35.

[18] Antoine Pietri, Diomidis Spinellis, and Stefano Zacchiroli. 2019. The software heritage graph dataset: public software development under one roof. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR).* IEEE, 138–142. https://doi.org/10.1109/MSR.2019.00030

[19] Ioannis Samoladas, Lefteris Angelis, and Ioannis Stamelos. 2010. Survival analysis on the duration of open source projects. *Information and Software Technology* 52, 9 (2010), 902–922. https://doi.org/10.1016/j.infsof.2010.05.001

[20] Charles M Schweik, Robert C English, Meelis Kitsing, and Sandra Haire. 2008. Brooks' versus Linus' law: an empirical test of open source projects. In *Proceedings of the 2008 international conference on Digital government research.* Digital Government Society of North America, 423–424. https://doi.org/10.5555/1367832.1367924

[21] Panagiotis Sentas, Lefteris Angelis, and Ioannis Stamelos. 2008. A statistical framework for analyzing the duration of software projects. *Empirical Software Engineering* 13, 2 (2008), 147–184. https://doi.org/10.1007/s10664-007-9051-7

[22] Spotswood L Spruance, Julia E Reid, Michael Grace, and Matthew Samore. 2004. Hazard ratio in clinical trials. *Antimicrobial agents and chemotherapy* 48, 8 (2004), 2787–2792. https://doi.org/10.1128/AAC.48.8.2787\T1\textendash2792.2004

[23] Chandrasekar Subramaniam, Ravi Sen, and Matthew L Nelson. 2009. Determinants of open source software project success: A longitudinal study. *Decision Support Systems* 46, 2 (2009), 576–585. https://doi.org/10.1016/j.dss.2008.10.005

[24] 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 *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering.* 644–655. https://doi.org/10.1145/3236024.3236062

[25] Bogdan Vasilescu, Daryl Posnett, Baishakhi Ray, Mark GJ van den Brand, Alexander Serebrenik, Premkumar Devanbu, and Vladimir Filkov. 2015. Gender and tenure diversity in GitHub teams. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems.* 3789–3798. https://doi.org/10.1145/2702123.2702549