Beyond Stars and Forks: Predicting Open Source Repository Maintenance Quality Through Statistical Analysis

Seng 404

Richard Gage V00801959
Jack Hamilton V00985614
Santiago Velasco V01050218

Introduction

Open source software (OSS) is a fundamental part of technological development today, allowing people from around the world to collaborate on projects. GitHub is the largest development platform for OSS. It allows users to create their own projects and contribute to others by creating forks. Forks are copies of the original project that allow users to make changes without affecting the main branch. A critical challenge facing open source software is project maintenance quality and sustainability having such a high variance.

Assessing repository health is very complex because traditional popularity metrics - such as stars and forks - do not tell the full story, missing out on the project's actual maintenance quality. A project that is very popular with thousands of stars may not have good issue resolution, which damages the quality and long term viability. The disconnection between popularity and maintenance quality creates many challenges for potential contributors selecting projects to support, researchers studying software sustainability, and developers determining dependencies.

The goal of this study is to provide evidence about which repository characteristics most reliably predict issue resolution effectiveness - the core indicator of good project maintenance. It also aims to analyze how different programming languages affect maintenance patterns and repository health. By identifying reliable indicators of maintenance quality, this research can help developers to determine software dependencies and therefore strategies for ensuring their projects are sustainable. It will also help contributors select projects that will be more likely to remain active.

To conduct this study, we analyzed 200 GitHub repositories spanning multiple programming languages and project types, examining the relationships between popularity metrics, contributor diversity, issue resolution effectiveness, and community engagement patterns. Our analysis employs traditional statistical methods and Bayesian analysis.

Motivation

In today's software landscape, the health and sustainability of open-source repositories is crucial for the developers that depend on them for their projects. Poorly maintained repositories can lead to a number of issues, such as technical debt, abandoned features, and a complete lack of community engagement. Current practices often rely on indicators like GitHub stars and forks as indicators for project quality, but popularity does **not** equal quality. Many popular projects suffer from massive numbers of open issues and poor maintenance health.

Prior research has already explored various indicators of repository health, such as issue resolution rates, contributor activity, and code quality metrics in the search to identify reasons behind project success or failure. Notably, Aranda & Venolia (2009) highlighted how the dynamics behind bugs and issues are more intricate than expected. Our goal is to expand on previously done research in a different direction, with the goal to investigate which specific characteristics and predictors best describe repository health, as well as how these relate to repository maintenance.

Our research addresses this gap by analyzing the relationship between commonly available GitHub metrics and actual maintenance outcomes. By identifying reliable predictors of maintenance quality and demonstrating the limitations of popularity-based assessments, this research can help promote more sustainable development practice across the OSS ecosystem.

Research Questions and Hypotheses

1. What repository characteristics best predict issue resolution effectiveness?

   H1: Community engagement metrics (contributors, comments per issue) will be stronger than popularity metrics (stars, forks).

2. How do programming languages influence repository health and maintenance patterns?

   H2: Languages with larger ecosystems (JavaScript, Python) will show different patterns than system languages(C/C++ / Rust).

3. What is the relationship between contributor diversity and project sustainability?

   H3: Projects with more distributed contributor bases will demonstrate better long-term sustainability metrics.

Related Work

The health and maintenance of repositories has been a point of research in recent years, with various studies approaching the topic from different angles. Some of them look towards programming language choice, commenting, social dynamics, or even open issues for answers, with most using predictive modeling to try and garner their solutions. One piece of related work, Berger et al. (2019), discussed how programming language choice impacts code quality by analyzing bug-fixing commits across multiple projects. Their findings suggested that languages can statistically influence open issues and the rate of defects within their code, and offered the insight that technical choices can correlate with a higher maintainability.

Another, Vishal et al. (2020), focused on code comments and how it related to issues and issue solving rates. The study revealed a small correlation that overly commented code may hint at complexity, and a potential of increased issues. This can tie into the repository health debate quite easily, as comments have a dual role as both a signal of proper quality, but also a marker for technical issues. On a more human perspective, Aranda and Venolia (2009) provided a qualitative perspective highlighting developer communication, bug triaging, and context gaps on how they influence open issue resolutions. The paper stressed that repository metrics miss the human side of bugs, which is equally crucial to the wellness of a repository.

A few more researchers also made efforts in similar veins. Linaker et al. (2022) reviewed how repository health is characterized, much like how we are observing characteristics, and noted fragmented practices and no clear conclusion on metrics. He et al. (2024) showed that a project's centrality in the OSS ecosystem can lead to longer lifespans, pointing to how important relational factors were beyond internal metrics. Lastly, Levin and Yehudai (2016) highlighted developer-level temporal and semantic patterns as very strong predictors of maintenance and activity.

Together, all these studies emphasize that a repository's health stems from a mixture of technical, social, and ecosystem factors; an idea central to our goal of identifying metrics in which we can predict sustainability, health, and flaws within projects.

Dataset

We analyzed 174 GitHub repositories collected on July 28th 2025, after filtering an initial sample of 200 repositories to exclude those with zero recent contributors. This filtering ensures focus on actively maintained projects, addressing concerns about data quality and aligning with best practices from "Promises and Perils of Mining GitHub" (Kalliamvakou et al., 2014). The dataset represents diverse projects across multiple programming languages and popularity levels, with metrics collected over a 180-day observation period.

Repository Characteristics

The 174 repositories span:

- 13 primary programming languages
- Popularity range: 667 to 237,663 stars
- Contributor range: 1 to 694 recent contributors

- Activity levels: 0.005 to 11.1 commits per day

Metrics Collected

Basic Repository Info

repository - The name of the GitHub project (e.g., "facebook/react")
stars - How many people "starred" it (like a "like" button)
forks - How many people copied the project to work on it
language - What programming language it's mainly written in

Health & Quality Scores

health_score - Overall project health rating (0-100, higher = healthier)
open_ratio - % of recent issues that are still unresolved (0 = all fixed, 1 = none fixed)

People Working on the Project

total_contributors - How many different people have contributed code
core_contributors - How many people do most of the work (the "main team")
contribution_gini - How evenly work is distributed (0 = equal, 1 = one person does everything)
top_contributor_percentage - What % of work the top contributor did

Development Activity

total_commits - How many code changes were made (in last 6 months)
commits_per_day - Average code changes per day
activity_ratio - What % of days had any activity (0.5 = active half the days)
recent_contributors - How many different people made code changes in 180 days

Issue Management (Bug Reports & Requests)

open_issues - Current number of unresolved problems/requests
recent_issues_count - New issues created in last 6 months
recent_issues_per_day - How many new issues appear daily
recent_resolution_rate - % of recent issues that got solved
avg_comments_per_issue - How much discussion each issue gets

Helper Variables for Analysis

log_stars - Mathematical transformation of stars (for better statistics)
log_contributors - Mathematical transformation of contributors
log_recent_contributors - Mathematical transformation of recent contributors
contributors_per_star - How many contributors relative to popularity
recent_contributors_per_star - How many recent contributors relative to popularity
commits_per_contributor - How productive each contributor is

recent_contributor_ratio - What % of all-time contributors are recently active
time_window_days - Time period analyzed (180 = 6 months)

Findings

H1: Community Engagement vs Popularity Metrics

The metrics "recent_contributors," "log_recent_contributors," "avg_comments_per_issue," "recent_contribution_ratio," and "activity_ratio" were considered community engagement metrics because they depend exclusively on work done inside the repository. Conversely, "stars," "log_stars," and "forks" were selected as popularity metrics because they reflect general public interest in a project, though they do not necessarily imply further contributions to the repository.

To test this hypothesis, the linear correlation between these two groups of metrics and the recent resolution rate capacity was analyzed.

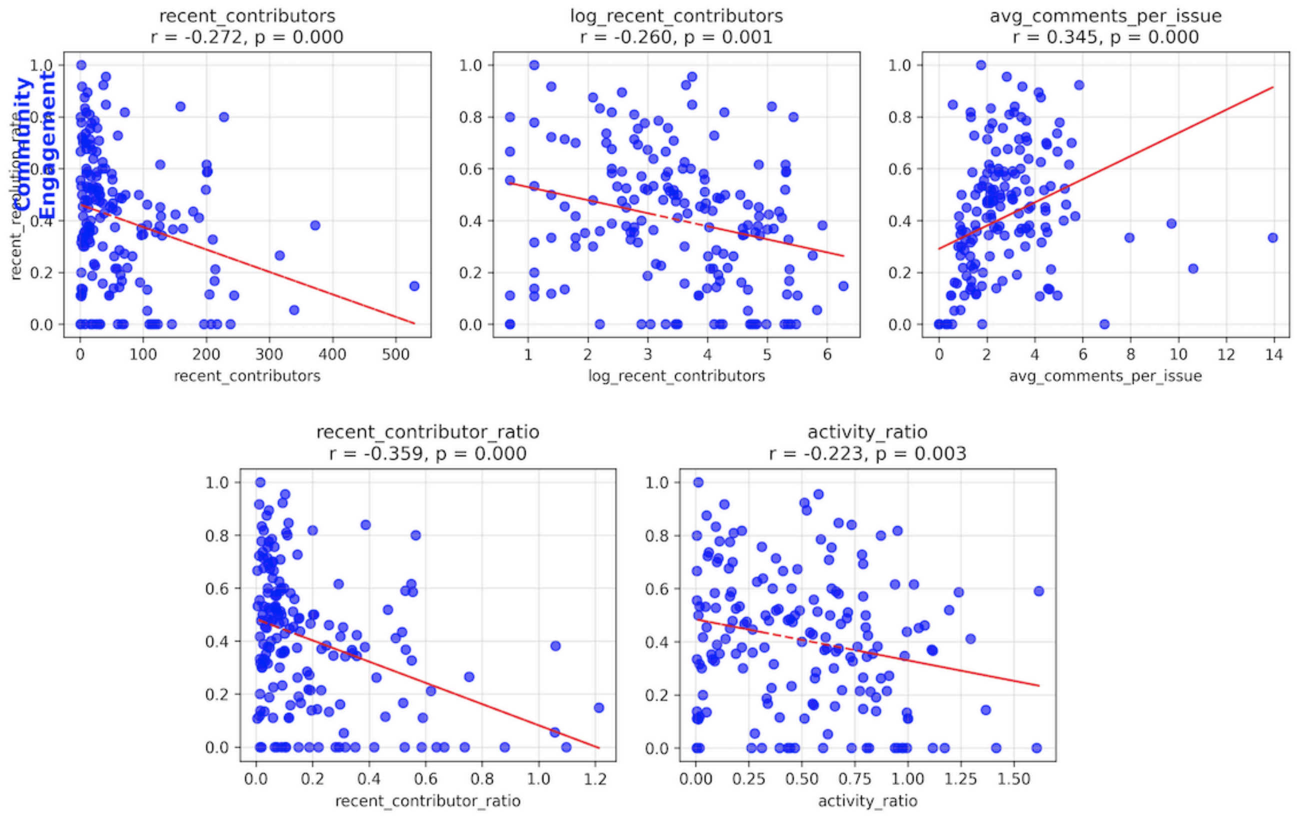

Figure 01. Community Engagement Metrics against Recent Resolution Rates.

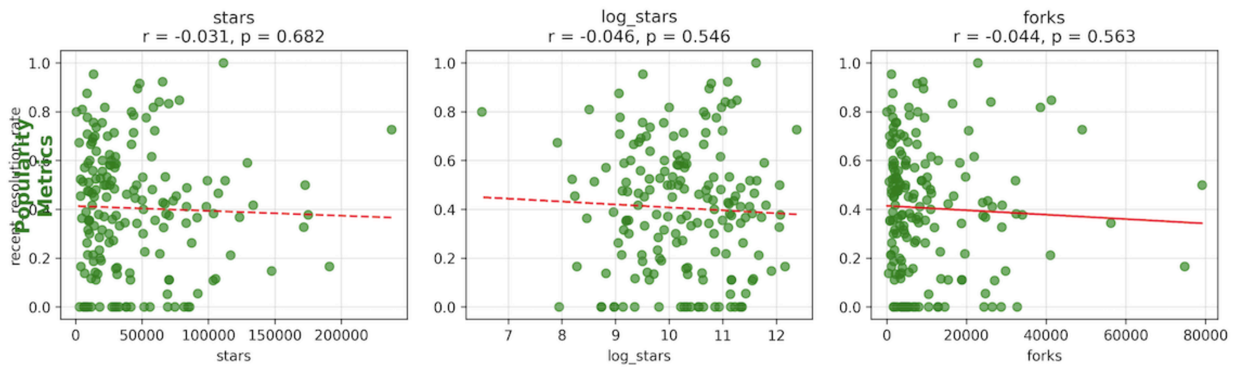

Figure 02. Popularity Metrics against Recent Resolution Rates.

The average correlation of the community engagement metrics was 0.292. In contrast, the popularity metrics obtained a value of 0.041. This difference is +0.251. A t-test was performed to determine the statistical significance of this difference, yielding a p-value of 0.0004. Therefore, at a 99% confidence level, it is shown that community engagement metrics are significantly stronger predictors, thus supporting H1.

H2: Languages with larger ecosystems (JavaScript, Python) will show different patterns than system languages(C/C++ / Rust).

The following figure shows the difference of the Recent Resolution Rate and Total Contributors metrics distributions grouped by Ecosystem and System Languages.

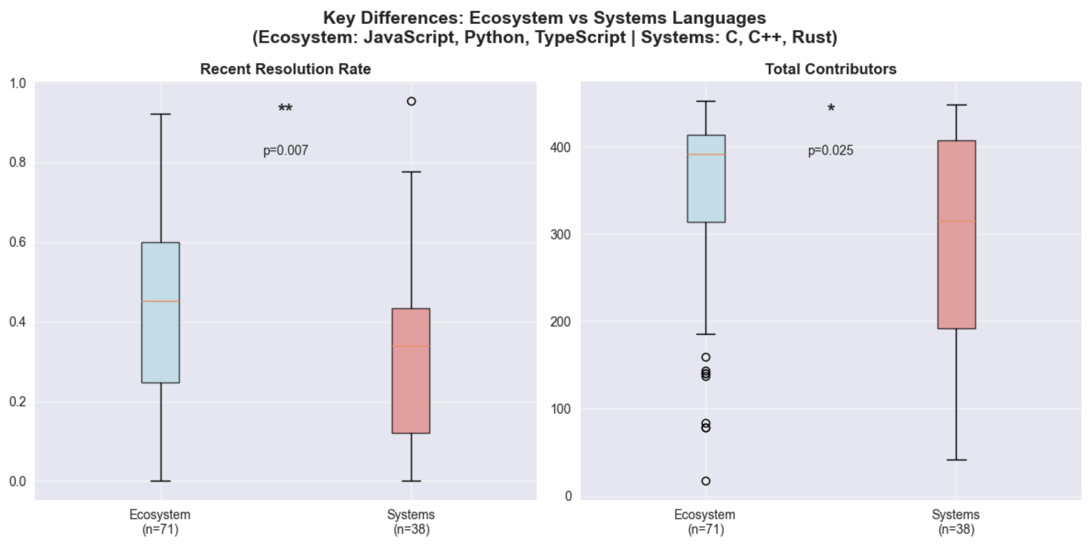

Figure 03. Recent Resolution Rates (Left) and Total Contributors (Right) in Ecosystem and System Languages

An independent samples t-test was used to compare ecosystem language with systems language. Effect sizes were calculated using Cohen's d; values of 0.2, 0.5, and 0.8 represent small, medium, and large effects, respectively. Cohen’s d was calculated using the formula: difference between the two means divided by the standard deviation. The following values were obtained:

recent_resolution_rate: Eco=0.438, Sys=0.303, d=+0.555 (Medium), p=0.0068 **
total_contributors: Eco=345.338, Sys=293.158, d=+0.456 (Small), p=0.0252 *

It was found that ecosystem languages resolve 14% more issues within the first six months. Additionally, the average number of total contributors for repositories using ecosystem programming languages is almost 400, while the average for repositories using system programming languages is slightly more than 300. The difference in recent resolution rate is significant, with a d-value of +0.555 and a p-value of 0.0068. For total contributors, the difference in means has a significant d-value of +0.456 and a p-value of 0.0252.

Therefore, at a 95% confidence level, the recent resolution rate and total contributors are both significantly different, so H2 is supported. Ecosystem and system programming languages exhibit distinct repository health and maintenance patterns.

H3: Projects with more distributed contributor bases will demonstrate better long-term sustainability metrics.

To test this hypothesis two approaches were taken, the first was to test the linear correlation between the Contributor Diversity and Recent Resolution Rate metrics. The second was to group the Recent Resolution Rate metrics by quartiles of the Contributor Diversity.

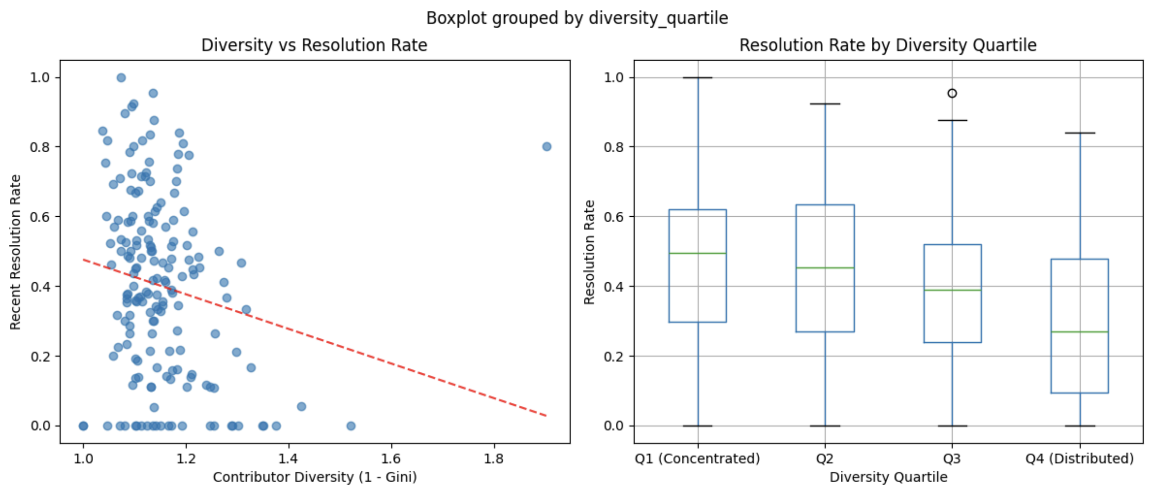

Figure 04. Linear Correlation of Contributors Diversity and Recent Resolution Rate (Left) Distribution of Recent Resolution Rate by Contributor Diversity Quartiles (Right)

It was found that Contributor Diversity negatively correlates with Recent Resolution Rate with a Pearson coefficient of -0.235 and significance p-value of 0.002. Another interesting finding was that projects with concentrated contributions (Q1) have a 46.6% resolution rate, while projects with distributed contributions (Q4) have 31.1% resolution rate, difference statistically significant with a t-value of -2.741 and p-value of 0.008. The effect size in the Recent Resolution Rate was 15.6 percentage points higher for concentrated projects.

There is significant evidence against H3, and therefore is rejected. This result was interesting, as it challenged our assumption that diversity improves outcomes. Having a smaller core team is more effective for issue resolution.

Discussion

The experimental findings indicated that the majority of the healthy properties in repositories adhere to the principle of "less is more." It has been demonstrated that repositories with the highest popularity metrics are not necessarily the most efficient ones. Indeed, emergent issues are resolved most efficiently by repositories that engage more with contributors, have constant activity, and are well documented. This prompts further experimentation, which in turn provides insight into the matter. The core contributors are essential for maintaining the quality of an open-source project. As demonstrated in previous studies, there is no clear correlation between the diversity of contributors and the ability to resolve open issues. The underlying causes of this phenomenon may be multifaceted, including the uneven distribution of work among contributors, which can impede the capacity of the most active and knowledgeable contributors. There are several other factors to consider. Firstly, the projects are characterized by a high degree of specification and complexity. Secondly, repositories contain a variety of sections that require contributors with specific skills. While attracting contributors is convenient, if they are unable to resolve issues in these specific sections, it becomes difficult to overcome the issues that cause bottlenecks.

It was also revealed that the metrics used to describe the healthiness of a repository can vary depending on the type of language employed. A recent study revealed that projects developed using languages with extensive ecosystems, such as Python or JavaScript, demonstrate higher efficiency in addressing recent issues compared to those employing Rust or C++. Two possible explanations can be posited. First, large ecosystem languages contain numerous tools and libraries that could facilitate the work of resolving issues. Second, large ecosystem languages tend to have lower skill entry, meaning that they are programming languages that do not require as much expertise as others to make useful contributions.

These results contrast from the ones obtained by Berger et al. (2019), they considered popularity as a fundamental piece to maintain the quality of a repository, as we previously explained our results indicate that popularity and quality are not correlated. Furthermore, they found that programming languages have little influence in the popularity of a repository and hence its quality, which differs from our test that showed a correlation between the type of programming language used and the number of recent closed issues.

Finally, Berger et al. (2019) obtained that metrics such as number of projects/developers followed by team members obtained a higher correlation coefficient with the ability of repositories to close issues within a window of 3 days and 365 days. Our research concluded that a major concentration of contributors and better community engagement metrics are associated with a stronger capacity to resolve recent issues.

Limitations

Temporal Bias

Our 180-day observation window provides recent activity data but may miss longer term sustainability patterns. Projects with seasonal development cycles or those in a refactoring stage now could show artificially low resolution rates during this time period. This window may also favour projects with faster release schedules versus projects with longer development timelines. Future research could include longer windows to compare both styles.

Statistical Limitations

Our analysis focused on correlations and t-tests, which assumes linear relationships that may not reflect the more complex dynamics of software development.

Sampling Bias

Our 174 repository sample is a very small fraction of GitHub's 400 million repositories, limiting the generalizability. We also favoured popular projects, and we recognize that very niche projects could be healthy but very difficult to stumble upon and use.

Missing Metrics

Due to time restraints on the project, we admit that there could be additional metrics that would likely change the results such as:

- Funding
- Project Age
- Documentation Quality

GitHub API Limitations

We had no access to any private activity, only their public activity. Comments per issue have no further context - there is no guarantee that it is healthy discourse. Those comments might actually be debate or disagreement.

Ethical Considerations

Mental Health

Having a more concentrated contributor base shows better metrics for project health, but it does not take into consideration that these contributors are human (for now at least). There is an unmeasured mental cost to pushing more of the weight of projects onto less developers, and there is a real risk of burnout for those people. Without considering the higher workloads impacts on mental health seems dangerous.

Funding Implications

We admit that we have recommended for funding to focus on core contributors rather than expansion, which could make it harder for newer developers to gain experience and get their feet in the door.

We must emphasize that our metrics capture only certain aspects of project health, and that different contexts require different approaches.

Conclusion

Our analysis of 174 GitHub repositories challenges fundamental assumptions about open source project health and sustainability. The data reveals three critical insights that should be taken into consideration when we evaluate and support open source projects.

First, the metrics that matter most for project health are not the ones most visible. Community engagement metrics will be stronger than popularity metrics, it is best to stop optimizing for GitHub stars. GitHub stars do not imply any efficient issue resolution, as our analysis shows they have a correlation of $r = -0.031$ with issue resolution rate. Community engagement metrics such as average comments per issue have $r = 0.345$. These numbers suggest that active discussion on issues between project contributors will have a greater effect on issue resolution than focusing on increasing the repository popularity.

Turning towards our second accepted hypothesis: languages with larger ecosystems will show different patterns than system languages. We found that projects primarily using ecosystem languages have a 14% higher resolution rate for issues compared to systems languages. This result reflects that metrics describing healthiness will differ between different very different languages. The communities and tools all impact project sustainability, which makes a single overarching evaluation metric that accounts for these differences difficult to create.

Thirdly, and most counterintuitively, we believed that more contributors would be more effective, but in reality, less is more. Projects with concentrated contributor bases achieve 15.6% higher resolution rate than more distributed projects. Simply increasing the number of contributors does not necessarily improve the health of a repository, since the work is not evenly distributed and it is necessary to identify where work concentrates the most. It is important to note that although we have statistically significant results here, there are likely measurable benefits to having more contributors involved that our dataset did not cover.

These findings have immediate practical implications. Core maintainer teams should be prioritized rather than expanding contributor bases for project teams, which implies where funding should be focused. Developers looking for dependencies should not focus on popularity metrics, and instead examine issue discussion activity levels as well as issue resolution patterns.

In closing, there are many pitfalls that developers will fall into when working on their projects, and seemingly intuitive metrics can be very misleading. It is not always obvious as to why some repositories fail and struggle with sustainability, but as our data has shown, there are always solutions to optimize both workflow, and collaborative work. The most popular projects are not necessarily the healthiest, and the most diverse teams are not always the most effective. Looking towards the future, there are always more aspects of repository wellness to be considered, and there are sure to be more characteristics within even our dataset that we had yet to consider that might lead to surprising conclusions. Developing a better understanding of open source project sustainability is critical for the long term stability of our digital infrastructure.

AI Usage

GenAI was useful for:

- searching for similar research papers
- assisting with coding for plotting data results.
- assistance with navigating GitHub API Data Collection

References

Berger, E. D., Hollenbeck, C., Maj, P., Vitek, O., & Vitek, J. (2019, April 24). *On the impact of programming languages on code quality*. arXiv.org. https://arxiv.org/abs/1901.10220

Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D., & Damian, D. (n.d.). The promises and perils of mining github | proceedings of the 11th Working Conference on Mining Software Repositories. https://dl.acm.org/doi/10.1145/2597073.2597074

Vishal, M et al. (2020, March 30). *Is there a correlation between code comments and issues?: an exploratory study*. ACM Digital Library. https://dl.acm.org/doi/10.1145/3341105.3374009

Aranda, J., & Venolia, G. (2009). *The secret life of bugs: Going past the errors and omissions in software repositories*. Research Gate. https://www.researchgate.net/publication/221555113_The_secret_life_of_bugs_Going_past_the_errors_and_omissions_in_software_repositories

Linåker, J., Papatheocharous, E., & Olsson, T. (2022, August 1). *How to characterize the health of an open source software project? A Snowball Literature Review of an emerging practice*. arXiv.org. https://arxiv.org/abs/2208.01105

Destefanis, G., & London, Brunel University. (n.d.). *Introducing repository stability*. https://arxiv.org/html/2504.00542v1

He, R., Ye, H., & Zhou, M. (2024, May 13). *Revealing the value of repository centrality in lifespan prediction of Open source software projects*. arXiv.org. https://arxiv.org/abs/2405.07508

Levin, S., & Yehudai, A. (2016, November 30). *Using temporal and semantic developer-level information to predict maintenance activity profiles*. arXiv.org. https://arxiv.org/abs/1611.10053

Soto, Martín & Ciolkowski, Marcus. (2009). The QualOSS open source assessment model measuring the performance of open source communities. 498-501. 10.1145/1671248.1671316. https://www.researchgate.net/publication/221494821_The_QualOSS_open_source_assessment_model_measuring_the_performance_of_open_source_communities

Jarczyk, Oskar & Gruszka, Błażej & Jaroszewicz, Szymon & Bukowski, Leszek & Wierzbicki, Adam. (2014). GitHub Projects. Quality Analysis of Open-Source Software. 80-94. 10.1007/978-3-319-13734-6_6.https://www.researchgate.net/publication/278703145_GitHub_Projects_Quality_Analysis_of_Open-Source_Software

Appendix A

All data, and analysis scripts are available at: https://github.com/UVic-Data-Science-for-SE/404-504-project-team-1

Appendix B

Jack: Conclusion, Motivation, Related Work, Article Research, References, Reviewing

Santiago: Discussion, Findings Interpretation and Editing, Introduction, Related Work, References, Initial Findings

Richard: Dataset acquisition, Findings, results, limitations, ethical considerations, conclusion editing.