Two Differing Approaches to Survival Analysis of Open Source Python Projects

Derek Robinson, Keanelek Enns, Neha Koulecar, Manish Sihag
Department of Computer Science
University of Victoria
Victoria, Canada
{drobinson, keanelekenns, nehakoulecar, manishsihag}@uvic.ca

I. MOTIVATION

The developers of Open Source Software (OSS) projects are often part of decentralized and geographically distributed teams of volunteers. As these developers volunteer their free time to build free software for the masses, they likely want to ensure the projects they work on do not become inactive. If OSS developers knew which projects would remain active and which would become inactive, they could better decide if their free time was worth volunteering to a given project. Understanding which attributes of an OSS project lead to its longevity is what motivated Ali *et al.* to apply survival analysis techniques commonly found in biostatistics to study the probability of survival for popular OSS Python projects [1]. Ali *et al.* specifically studied the effect of publishing major releases, the use of multiple repositories, the type of version control system (VCS), and the size of the volunteer team had on the survival of OSS Python projects. We resonate with ALi *et al.'s* motivation and would like to replicate [1] in order to determine if there are any shortcomings in their analysis. In addition to the replication study, we also plan on applying a Bayesian approach to survival analysis as outlined in [2]. The Bayesian portion of our paper is motivated by comparing the findings of the two different approaches to survival analysis.

Thus, the research questions we plan on answering are as follows:

1) **How do major releases, the use of multiple repositories, the type of VCS, and the size of the volunteer team affect the survival of an OSS Python project?**

2) **How do the findings of frequentist survival analysis differ from Bayesian survival analysis?**

II. RESEARCH STRATEGIES

A. Data

Performing survival analysis of OSS projects requires a dataset that records the repositories for projects on common VCSs, including a history of all commits (revisions from here on out) and major releases (revisions of note, often with a specific name and release date) is required [1]. The *popular-3k-python* subset of the Software Heritage graph dataset [3] is what will be used in both our replication study and Bayesian survival analysis study. This dataset contains information on roughly 3000 popular Python projects hosted on Gitlab, GitHub, Debian, and PyPI between 2005 and 2018.

B. Methods

Survival analysis is a set of methods used to determine how long an entity will live and is most often used in the medical field. For example, survival analysis techniques can determine the probability that a patient will survive when given a certain treatment. The methods of survival analysis we will be using in the replication study are the Kaplan-Meier (K-M) survival estimator, and we will also be fitting a Cox Proportional-Hazards model. For the comparison between traditional survival analysis and Bayesian survival analysis, we will be applying the methods found in [2].

1) Kaplan-Meier Estimator and Cox Proportional-Hazards Model: When applying survival analysis techniques, one very important aspect is censoring. If our event of interest is the inactivity of a project and this event does not occur during the period our data covers, then the time to event is said to be censored. Ali *et al.* decided that the period that they would analyze was 165 months long, starting in 2005 and ending in January 2018. They also deemed that any project "that has revisions beyond the January 2018 cutoff date is surely active and is deemed censored." [1] The K-M estimator is used to estimate the survival function. In our use case, the K-M estimator will estimate the probability that the duration of a project is longer than time t. The other analysis we will be performing is fitting a Cox Proportional-Hazards model. The Cox Proportional-Hazards model is a regression model that will allow us to investigate the association between the survival of a project and its key attributes.

2) Bayesian Survival Analysis: For the second portion of our study, we will be applying the same methods as found in the section titled *A detailed example* of [2]. More specifically, we

will be using a parametric exponential model that assumes the survival times of a project $y = (y_1, y_2, \ldots, y_n)$ are exponentially distributed with parameter λ. We will then use our model to visualize the posterior survival functions for the following four project attributes: major releases, VCS of the project, use of multiple VCSs, and team size.

III. EXPECTED RESULTS

We expect that our replication of [1] will yield similar results to the original analysis. Additionally, as the survival analysis techniques outlined in [1] and the Bayesian approach to survival analysis outlined in [2] are two different approaches to the same goal, we suspect that the results of both will be comparable.

IV. LIMITATIONS

Both the data we analyze and the methods of analysis have their own respective limitations, as such, this section will cover each individually.

A. Limitations of the Methods

B. Limitations of the Data

The data we are analyzing has been aggregated from multiple version control systems across the web over a long period. As such, the data set is not fully reproducible, as pointed out by the original authors of the Software Heritage Graph [3]. Additionally, we cannot ensure that the data we are analyzing is a full history of the respective repositories. The lack of certainty about the full history is because the repository admin can modify the history of revisions to suit their liking.

REFERENCES

[1] R. H. Ali, C. Parlett-Pelleriti, and E. Linstead, "Cheating death: A statistical survival analysis of publicly available python projects," in *Proceedings of the 17th International Conference on Mining Software Repositories*, 2020, pp. 6–10.

[2] R. Kelter, "Bayesian survival analysis in stan for improved measuring of uncertainty in parameter estimates," *Measurement: Interdisciplinary Research and Perspectives*, vol. 18, no. 2, pp. 101–109, 2020.

[3] A. Pietri, D. Spinellis, and 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)*. IEEE, 2019, pp. 138–142.