Two Differing Approaches to Survival Analysis of Open Source Python Projects

DEREK ROBINSON, KEANELEK ENNS, NEHA KOULECAR, and MANISH SIHAG,
University of Victoria

[Keanu: Abstract Pending]

CCS Concepts: • **Software and its engineering** → **Open source model**; • **Information systems** → *Data mining*;

Additional Key Words and Phrases: data science, survival analysis, open source, python, Kaplan Meier, Cox proportional hazards model, Bayesian analysis

ACM Reference format:

Derek Robinson, Keanelek Enns, Neha Koulecar, and Manish Sihag. 2021. Two Differing Approaches to Survival Analysis of Open Source Python Projects. 1, 1, Article 1 (November 2021), **??** pages.

https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

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 such OSS projects, they likely want to be confident that the projects they work on will not become inactive. If OSS developers are aware of key attributes that are associated with long-lasting projects, they can make informed assessments of a given project before devoting their time to it or they can strive to make their own projects exhibit those attributes. 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 [**?**]. Ali *et al.* specifically studied the effect of the following attributes on the survival of OSS Python projects: publishing major releases, the use of multiple repositories, the type of version control system (VCS), and the size of the volunteer team.

Survival analysis is a set of methods used to determine how long an entity will live (or the time to a given event) and is most often used in the medical field. **[Keanu: do we need to find a citation for this claim/definition?]** For example, survival analysis techniques can determine the probability that a patient will survive when given a certain treatment. Ali *et al.* use a frequentist approach to survival analysis with methods including the Kaplan-Meier (K-M) survival estimator and the Cox Proportional-Hazards model. Though there are advantages to using such approaches **[Keanu: list some advantages and cite]**, another approach to survival analysis, Bayesian analysis, has its own set of advantages, namely **[Keanu: find the advantages, list them, cite]**.

The authors of this paper resonate with Ali *et al.'s* motivation. This paper serves as a replication of [**?**] and seeks to determine if there are gaps or shortcomings in their analysis. This replication also provides artifacts so that others may see how the study was conducted and reproduce it with ease. In addition to the replication, this paper analyzes the same data set using a Bayesian approach to survival analysis as outlined in [**?**] and seeks to compare the results of frequentist and bayesian approaches in the same domain. Thus, the research questions this paper answers are as follows:

RQ1. 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?

RQ2. Are there any shortcomings or gaps in the study done by Ali *et al.*?

RQ3. How do the findings of frequentist survival analysis differ from Bayesian survival analysis?

The following section describes the data used for the studies in this paper. Section 3 covers the methods used for both the replication and bayesian analysis study. Section 4 shows the results for each analysis. Section 5 is a discussion about the limitations of the study, the differences between the types of analysis performed, and suggestions about future work in the topic. The final section concludes by summarizing the purpose and findings of this paper.

2 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) [**?**]. The *popular-3k-python* subset of the Software Heritage graph dataset [**?**] was used in both the replication and Bayesian **[Keanu: should bayesian be capitalized?]** survival analysis study. This dataset **[Keanu: dataset or data set?]** contains information on roughly 3000 popular Python projects hosted on Gitlab, GitHub, Debian, and PyPI between 2005 and 2018. The Software Heritage Graph offers a tutorial on how to import this data [**?**].

3 METHODS

Most of the work done in this study has been documented in a repository for the benefit of the reader. A reference to this repository **[Keanu: and more?]** can be found in appendix A below. **[Keanu: define artifacts?]**

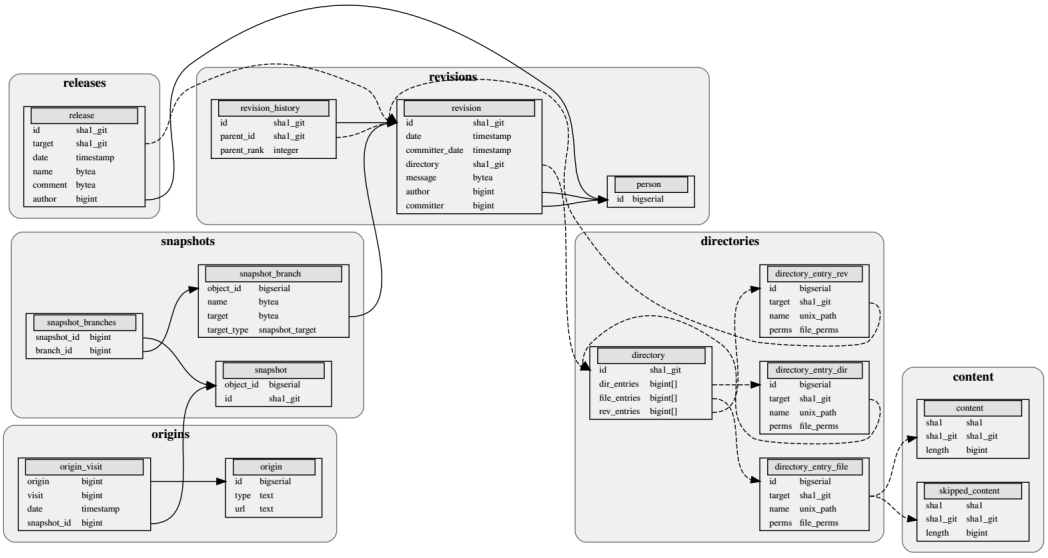

Fig. 1. The results of running the Cox Proportional hazards tool on the data. **[Keanu: we definitely want better descriptions for the images]** **[Derek: I don't think that we really need the schema in the paper]**

3.1 Replication

[Keanu: I'm having a hard time figuring out what Neil would want here. I don't want to just repeat what the original study said, but I'm not sure what else to do] **[Derek: We need to to talk about the same topcis that Ali et al. did. However, we need to use our own language. Maybe finding another SE replication paper and basing it off how they talk about the same topics as the original?]**

Steps in the replication:

Importing data from Software Heritage Graph [**?**]

Inspecting the provided schema

Crafting queries to give us the necessary information (join necessary tables and group by appropriate fields, filter out dates that were out of range, create indicator field for censoring etc.)

Constructing additional fields (multi_repo and duration) using pandas in python. Sort by duration and plot project from start to end month to recreate figure 1.

Perform frequentist analysis on data using survival package in R. First apply kaplan meier based on each attribute, then run Cox proportional hazards tool on the data.

3.2 Bayesian Survival Analysis

A common distribution for survival analysis is the exponential distribution [**? ?**]. **[Keanu: I don't have anything to write for this]**

4 RESULTS

4.1 Replication

We found the results of our replication study to be extremely similar to those shown in the original paper.

[Keanu: Put images here]

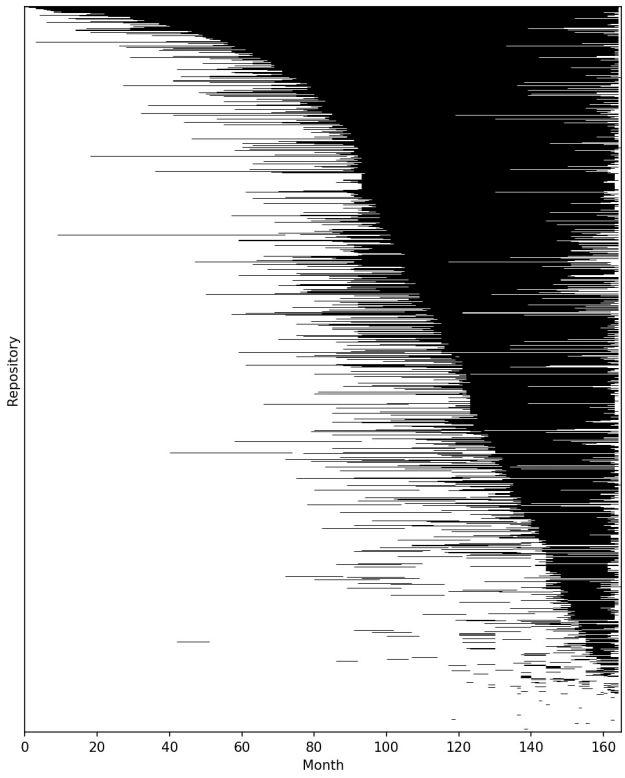

Fig. 2. Graph of project timelines within the given time frame. The projects are ordered by duration.

[Keanu: talk about significance and meaning of the KM and Cox results]

[Keanu: As you can see in the Cox table summary, the numbers in each category are slightly different than the original study, we plan to look further into what has caused this difference in values]

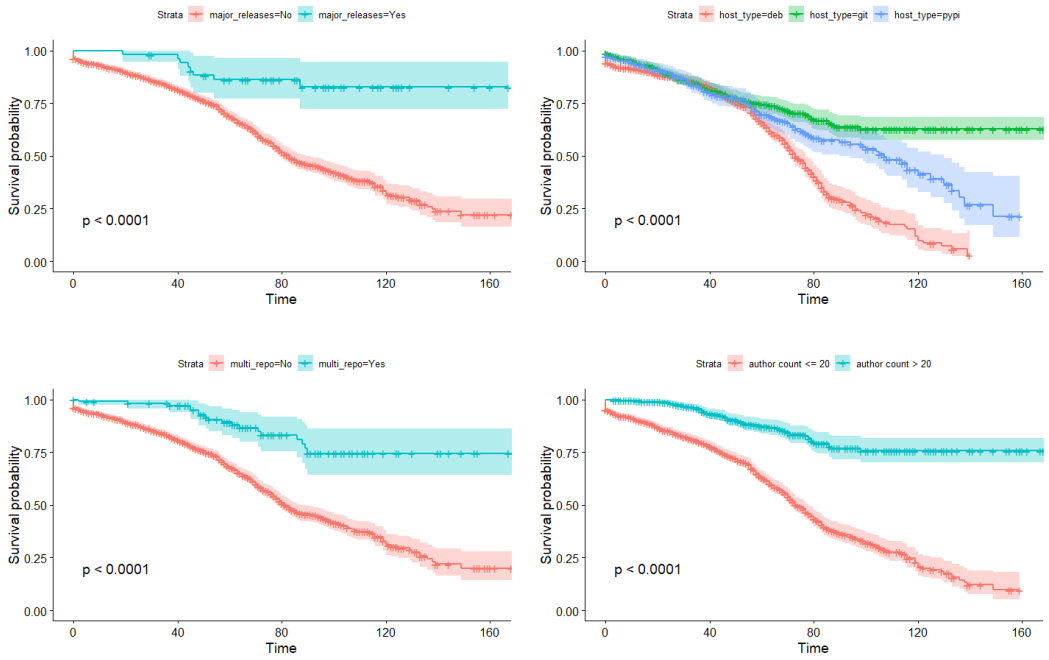

Fig. 3. KM curves of the python project data analyzed by major releases, host type, multiple repository hosting, and number of authors

4.2 Bayesian Survival Analysis

[Keanu: Nothing for this yet]

5 DISCUSSION

5.1 Comparison of Analyses

5.2 Limitations

5.2.1 Limitations of the Methods. The method Ali *et al.* used for censoring was slightly ambiguous.

5.2.2 Limitations of the Data. There was no clear method for determining whether a project was hosted in multiple repositories. The only unique identifier for each project was a url, from which the project name was extracted using regular expressions. This may have resulted in the extraction of inconsistent project names across host types. The assumption is that each project that was hosted on multiple repositories would be given the exact same name, and that projects of the same name hosted on different platforms were indeed the same project (which may not always be the case). Additionally, it is possible for users on Github to give their projects the exact same name as pre-existing

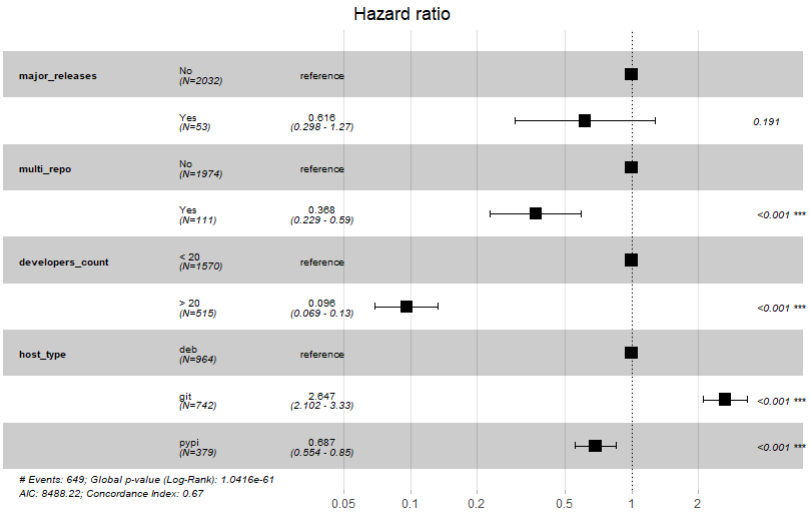

Fig. 4. The results of running the Cox Proportional hazards tool on the data. **[Keanu: we definitely want better descriptions for the images]**

projects created by other users. However, this issue was accounted for in our analysis, but it is unclear whether Ali *et al.* accounted for this.

The data contains a large portion of censored data. This means that we did not observe the abandonment of most of the projects. As data points are censored (denoted by the vertical tick marks in the KM curves), we have a smaller and smaller group of data points to study. This means that the results towards the 165 month mark may be less representative. This is to be expected with any SE study, as recent years have lead to an exponential increase in the number of OSS projects **[Keanu: find citation?]**.

The data set contained many revisions (over 4 million) that were not associated with project URLs. It is unclear how this happened.

5.2.3 *Replication Challenges.*

5.3 Future Work

Increase the time frame of the study (the original paper could not do this because the paper was written in 2018).

6 CONCLUSION

A ARTIFACTS

Project Repository: https://github.com/DerekRobin/CSC578B-Project