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 (December 2021), 11 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. Suppose OSS developers are aware of key attributes that are associated with long-lasting projects. In that case, 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 motivated Ali *et al.* to apply survival analysis techniques commonly found in biostatistics to study the probability of survival for popular OSS Python projects [2]. 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 hosting services, the type of version control system (VCS), and the size of the volunteer team.

Manuscript submitted to ACM

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. For example, survival analysis techniques can determine the probability of a patient surviving past a certain time when given a certain treatment. Ali *et al.* use a frequentist approach to survival analysis utilizing such methods as the Kaplan-Meier (K-M) survival estimator and the Cox Proportional-Hazards model [5, 6]. Though frequentist approaches are considered to be unbiased, minimal in variance, efficient, and generally sufficient, some consider them to lack robustness [10]. Another approach to survival analysis, Bayesian analysis, is considered to generate more robust models that perform well under the presence of new data being introduced and are generally easier to interpret results from [10]. **[Keanu: We will flesh this discussion out a bit more and use more citations]**

The authors of this paper resonate with Ali *et al.'s* motivation. This paper serves as a replication of their paper [2] (referred to as the original paper from here on) and seeks to determine 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. For the sake of the authors' curiosity, an additional attribute of the data was analyzed: the commit frequency of the project. In addition to the replication, this paper analyzes the same data set using a Bayesian approach to survival analysis as outlined in [7] and seeks to compare the results of the 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 hosting services, the type of VCS, and the size of the volunteer team affect the probability of survival of an OSS Python project?

RQ2. How does the commit frequency of an OSS Python project affect the probability of its survival?

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 and the process used for manipulating the data into a usable state. Section 4 covers the methods used for the replication, additional frequentist analysis, and Bayesian analysis. Section 5 shows the results for each analysis. Section 6 discusses the comparative differences between the analyses performed, the limitations of the study, related work, and suggestions for future work on the topic. The final section concludes by summarizing the purpose and findings of this paper.

2 RELATED WORK

Although the application of survival analysis is new to predicting the health of software repositories, several studies have attempted to analyze the health of open-source projects using different techniques. Xia *et al.* applied DECART hyperparameter optimization to predict health indicators of open-source projects and observed that DECART reduced prediction errors of certain estimation algorithms by 10% [14]. Norick *et al.* analyzed open-source projects using code quality measures and observed no significant evidence that the number of committing developers affects software quality [8]. Another study by Caivano *et al.* investigated the spread and evolution of dead code by analyzing commit histories of open-source Java applications [4]. They found that dead methods generally survive long, are rarely revived, and that a majority have been dead since their creation.

An application of survival analysis to software was a study by Samoladas *et al.* which predicted future development of FLOSS projects using project duration, application domain. and the number of committers as predictors [11]. The study calculated the probability of termination or continuation for FLOSS projects and quantified the benefit of adding more committers.

3 DATA

Performing survival analysis of OSS projects requires a data set 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) [2]. Revisions are used to assess the activity or health of a project.

The *popular-3k-python* subset of the Software Heritage graph data set [9] contains the necessary information and was used in both the original paper and this paper (a reference to the data set can be found in appendix A of this paper). This data set contains information on roughly 3000 popular Python projects hosted on GitHub/GitLab, Debian, and PyPI between 2005 and 2018. The Software Heritage organization offers a tutorial on how to import this data by creating a postgreSQL database [1] as well as an explanation of the structure of the database (a database schema can be seen in figure 2 of their document [9]) .

Though the data contains all the necessary information to perform survival analysis, it requires additional manipulation into a new format before the analysis can be carried out. For example, most of the analyzed features or attributes of the projects in the original paper are not present in the raw data set and need to be calculated. The work done in this study has been documented in a repository for the benefit of the reader. References to the repository can be found in appendix A, and the jupyter notebook used to perform the data manipulation can be found in Analysis&Data/data-collection.ipynb.

Queries were created in order to join the necessary tables and extract the relevant values. The queries accomplish the following objectives: they group records by the *url* field of the origin table (used to identify projects), filter records to be within the desired time frame, count distinct authors that have participated in a given project, record the host service (depicted by the *type* field of the origin table), record the earliest and latest revisions associated with each project url, count the number of revisions made during each project's duration, and identify whether major releases were published. **[Keanu: should we show pseudocode of one of the queries? Also, I wonder if we can find a way to condense the queries into as few queries as possible for the final product]****[Derek: I think that condensing the queries is not required. If Neil decides to look at them he should be able to understand what is going on. If anything we should add explanatory markdown to the notebook.]**

The notebook uses popular python libraries such as psycopg2, pandas, numpy, and matplotlib to connect to the database, extract the results of the previously mentioned queries, manipulate and filter the data, calculate new fields, and plot results. **[Keanu: This section is getting long, I was going to go into details about the new fields that were calculated (multi_repo, commit_freq, etc.), but maybe we should wait and see]**

4 METHODS

4.1 Replication

Two critical concepts in survival analysis are events of interest and censoring. In the case of this paper, as well as the original paper, the event of interest is the abandonment of a project. That is, if a project stops receiving revisions, it is considered abandoned. However, it is impossible to determine whether a project will receive revisions in the future, so a threshold of some kind is required to determine whether the project has indeed been abandoned for the purposes of the study. This is where censoring is used. Censoring involves using data from subjects of a study for which the time of interest is not observed. There are multiple types of censoring in survival analysis, but this study uses random censoring or type III censoring. **[Keanu: or right censoring, we should flesh out the terminology as I have seen some conflicting definitions]** **[Derek: I would say just go with that Ali et al. say (type III/random censoring)]** Random censoring involves removing subjects (in this case, projects) from a study at varying times relative to when they began to be observed [10]. **[Keanu: find another citation to support this, http://www.stat.columbia.edu/ madigan/W2025/notes/survival.pdf mentions it, but refer to it as random type 1 right censoring]**

Ali *et al.* set a time frame of 165 months (where a month is defined as 28 days), starting in 2005 and ending in January 2018. This paper uses the same time frame and determines exact start and end dates. Using January 1, 2018, as a strict end date and maintaining the study duration as 4620 days (165 months as defined), the start date is found to be May 9, 2005.

Projects are censored if there is reason to believe that they would receive revisions after the end of the time frame (i.e. the last *observed* revision is unlikely to be the last revision of the project). In this paper, any project with revisions on or after November 1, 2017, was censored. **[Keanu: NOTE: Our censoring method may change. This is because the paper [2] conflicted with the video presentation [3] with respect to the censoring method. After further inspection, it appears the method in the paper is superior, as we only censor data that truly had revisions after the time frame (though this probably will not affect many of the data points)]** The reason this is considered random censoring is that projects that are censored have varying start dates within the time frame of the study. The distribution of the project timeline can be seen in Figure 1, which is a replication of Figure 1 in the original paper.

Using both the calculated duration associated with each project and the censoring status of each project, the survival analysis can be performed. The analysis was carried out using an R notebook (this can be found in the project repository under Analysis&Data/frequentist-analysis.rmd).

The K-M estimator is a non-parametric estimation technique that estimates the survival function, $S(t)$. The survival function gives the probability that a given project will survive past a particular time t. At $t = 0$, the K-M estimator is 1 and as t approaches infinity, so does the K-M estimator. More precisely, $S(t)$ is given by $S(t) = p_0 \times p_1 \times p_2 \times \cdots \times p_t$, where p_i is the proportion of all projects that survived at the i^{th} time step.**[Keanu: citation?]** The K-M estimator produces curves that approach the true survival function of the data.

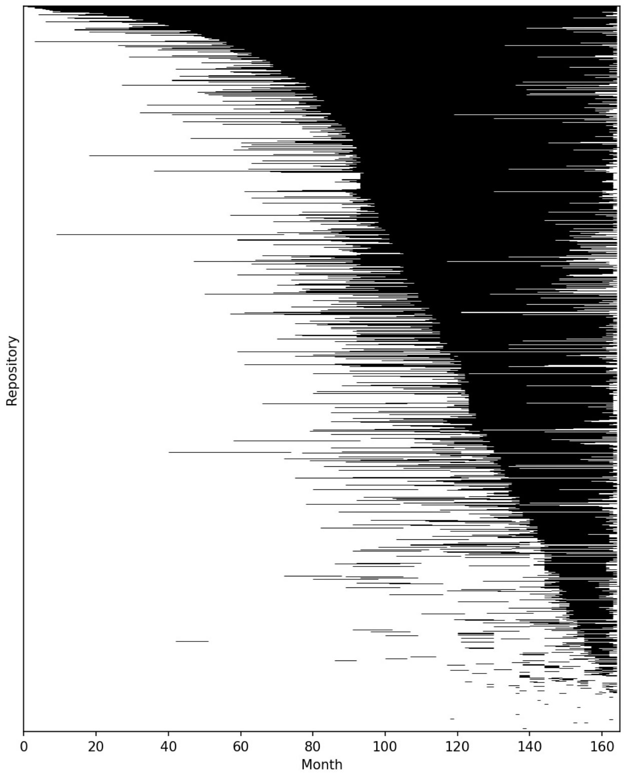

Fig. 1. Graph of project timelines within the given time frame. The projects are ordered by duration and plotted from and to their respective start and end dates. Month 0 begins on May 9, 2005, and Month 165 concludes on January 1, 2018.

The hazard function is another useful function in survival analysis. It describes the probability of the event of interest or hazard (project abandonment in this case) occurring up to a given time point. The original paper uses the Cox Proportional-Hazards model which allows for fitting a regression model in order to better understand how the health of projects relate to their key attributes. The previously mentioned R notebook makes use of the survival package for K-M estimation and Cox Proportional-Hazards model. It also utilizes the survminer package, which provides functions for facilitating survival analysis and visualization. The analysis begins by using the Surv() method to build the standard survival object and the survfit () method to produce the K-M estimates of the probability of survival over time. K-M curves, with their associated confidence interval for all the four attributes described in the original paper, are generated via the ggsurvplot() function, which also plots the p-value of a log-rank test. The next step was to build the Cox proportional hazards using the coxph() function and visualize them using ggforest (). This analysis results in the hazards ratio (HR), which is derived from the model for all covariates that are included in the formula. Briefly, a $HR > 1$ indicates an increased risk of abandonment; on the other hand, $HR < 1$, indicates a decreased risk of abandonment. As such, the HR represents a relative risk of abandonment that compares one instance of a binary feature (e.g. yes or no) to the other instance. **[Keanu: need citations for this section]**

4.2 Revision Frequency Analysis

The original paper mentions that "The health of a project could be computed by the number and frequency of contributions..." but never addresses this measurement. This paper seeks to explore this method of assessment. The commit frequency (defined as the number of commits divided by the number of days in the observed project's duration) was dichotomized into two groups depending on the frequency above and below the median. This study applies both the K-M estimator and the Cox Proportional-Hazards model to the data to stratify the effects of high commit frequency on the overall health of an open-source project. **[Keanu: more to come...]**

4.3 Bayesian Survival Analysis

This section focuses on the Bayesian approach to survival analysis as outlined in [7]. Although applying the Bayesian approach to survival analysis is less common due to computational difficulties, it offers multiple advantages over the frequentist approach. The Bayesian analysis uses posterior distributions of model parameters to conclude them. These posterior distributions are obtained via Markov-Chain-Monte-Carlo (MCMC) algorithms. The statistical modelling language used for Bayesian analysis is Stan, specifically the rstan interface. In addition to rstan, the R packages tidybayes and tidyverse were used to aid in visualization and data manipulation.

This study used a parametric exponential model that assumes the survival times of a project $y = (y1, y2, \dots, yn)$ are exponentially distributed with parameter λ. We have so far derived a posterior survival function for one of the project's attributes: the use of multiple VCSs. We will be further deriving posterior survival functions for the remaining attributes, namely major releases, VCS of the project, commit frequency and author count.

5 RESULTS

5.1 Replication

The replication study performed in this paper yielded extremely similar results to those shown in the original paper.

[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]

5.2 Revision Frequency Analysis

[Keanu: It is worth noting the difference between this graph and the other graphs. The other ones indicate a clear "winner" whereas this has an interesting point where the functions cross. More investigation is required, but my initial interpretation is that, while many projects with hight commit frequencies are abondoned shortly after, the ones that do end up maturing seem to have a higher probability of surviving

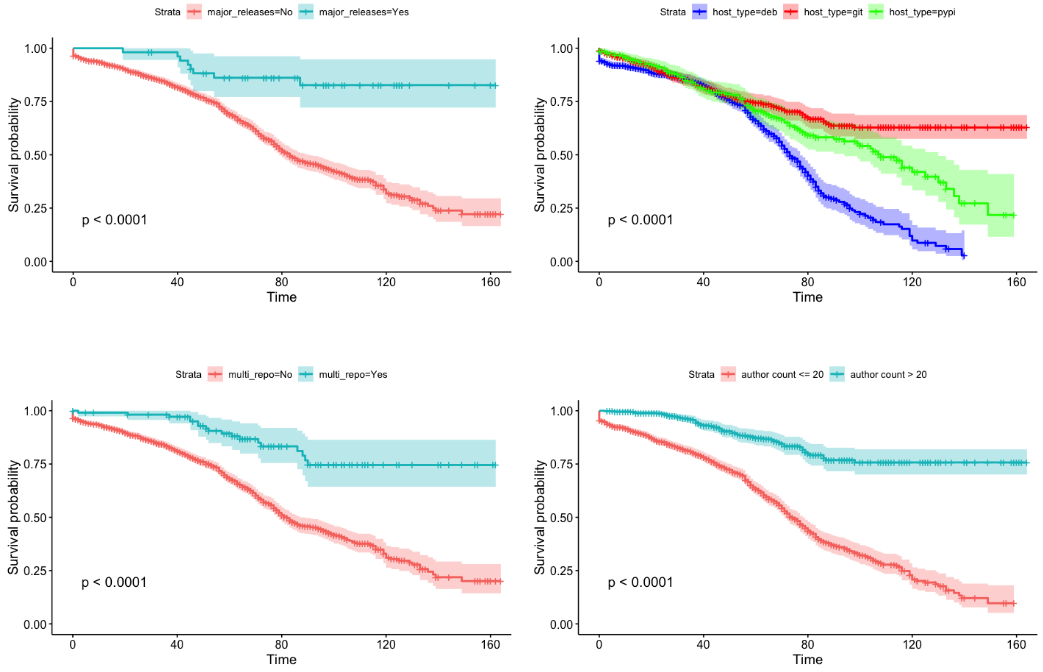

Fig. 2. KM curves of the python project data analyzed by major releases, host type, multiple repository hosting, and number of authors

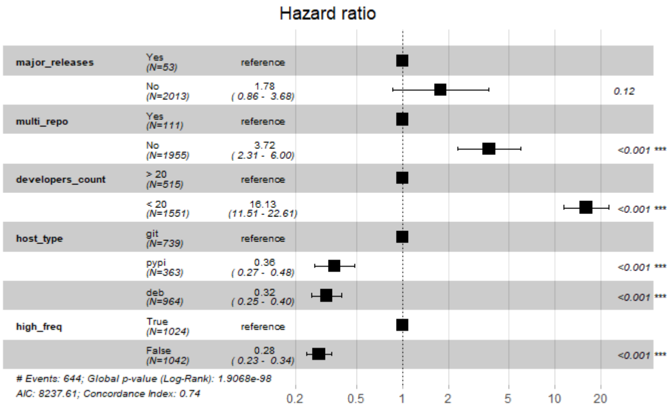

Fig. 3. The results of running the Cox Proportional hazards tool on the data. **[Keanu: we definitely want better descriptions for the images]**

even longer than the other projects. Maybe consistency is the key here, not so much the frequency. There may not be any statistical significance to this result though, so we will look into it.]

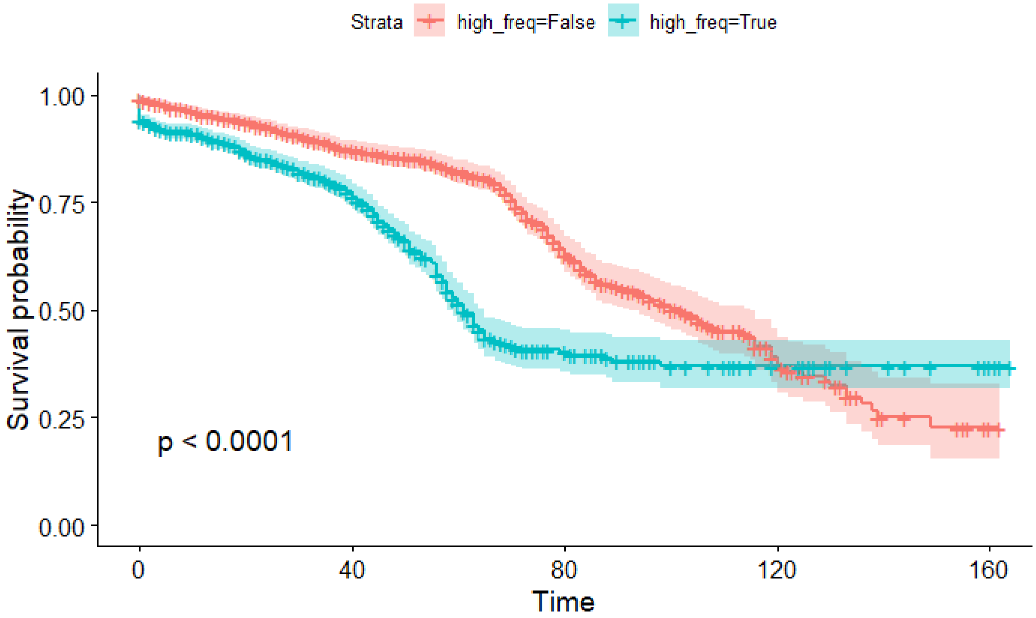

Fig. 4. KM curve for the data by revision frequency **[Keanu: need a better caption]**

5.3 Bayesian Survival Analysis

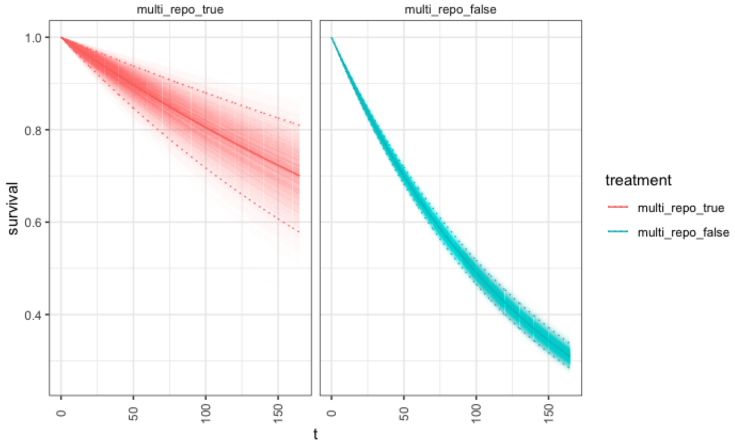

Fig. 5. Posterior survival functions of projects hosted on multiple repositories and projects hosted on a single repository

Shown in figure 5 are the posterior survival functions for projects hosted on multiple repositories versus those hosted on single repositories. The dotted lines represent the 2.5% and 97.5% quantiles while the middle solid line represents the posterior mean of β (the prior on the project attribute of interest). The remaining lines all represent valid posterior survival functions.

6 DISCUSSION

6.1 Comparison of Analyses

[Keanu: We are not quite ready for a comparison yet]

6.2 Limitations

6.2.1 Limitations of the Methods. The original paper [2] and the MSR presentation given [3] have contradicting methods of censoring. The method discussed in the original paper was deemed superior and is used in this paper. **[Keanu: not used as of now, but will be]**

Many details of the original paper's methods are left out for the sake of brevity. This lead to assumptions being made when manipulating the data. Hence, there are variations in the values retrieved compared to the original paper. **[Keanu: NOTE: We are continuing to look into the possible causes of these inconsistencies. We will analyze our data-collection jupyter notebook and compare it with the end result of the R notebook to verify consistency between them before concluding it is an issue with the methods used]**

Survival analysis methods such as the K-M estimator and the Cox Proportional-Hazards model have limitations of their own. When applying the K-M estimator, it is common to use a log-rank test to test the significance between the two groups which are being compared. The log-rank test only indicates whether or not the probability of survival is statistically significant between the two groups and is not able to provide any information about the size of the difference between the two groups [12]. Additionally, the K-M estimator does not account for confounding factors [12]. In more traditional uses of the K-M estimator, an example of a confounding factor could be the age of the study participants. In the case of this study, there may be confounding factors such as the experience level of the developers or whether the developers received funding to work on the project. Neither of these factors are represented in the data set. The Cox Proportional-Hazards model is used with the assumption that, over the period of observation, the hazards within each group are proportional [13]. If the assumption that the hazards within each group are proportional is not true, then the Cox Proportional-Hazards model will lead to incorrect estimates of the survival probability [13]. **[Keanu: we should assure the reader that it holds true in our case]**

The Bayesian approach to survival analysis comes with its own limitations as well. As pointed out by Renganathan, Bayesian survival analysis can be subjective as the analyst places their own bias into the model when selecting the prior distributions [10]. In order to mitigate this bias, prior selection requires both epistemological and ontological reasoning. **[Keanu: We need to flesh out our reasoning more, probably written in the methods, but then perhaps referred to here]**

6.2.2 Limitations of the Data. The data set in this study has been aggregated from multiple version control systems across the web over a large period of time. As such, the data set is not fully reproducible, as pointed out by the original authors of the Software Heritage organization [9]. Additionally, it cannot be ensured that the data contains 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. **[Keanu: add citation to perils of mining git]**

There are inherent differences in the ways developers use the different hosting services **[Keanu: I have a strong intuition that this is the case, it looks like PyPi and debian don't have as much granularity compared to git and only show major releases, but I don't have anything to back this up, so maybe we can find something to cite for that]**. This could skew the results because it may be the case that services such as PyPi and Debian are primarily used to host major releases of a product. This may hide information about the number of developers and the commit frequency. Additionally, the potential confounding factors mentioned in section 6.2.1 are not represented in this data set.

It may also be worth noting that this data is only for Python projects and that it is possible that different behaviours are associated with development in different languages. Python is a relatively easy language to use, and can often be used for small tasks that are not maintained. However, because this data set comprises popular projects, it is unlikely to contain projects that were used for a small task and discarded.

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 services 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 projects created by other users. This issue was accounted for in this paper's analysis, but it is unclear whether Ali *et al.* accounted for this.

The data contains a large portion of censored data. This means the abandonment of most of the projects was not observed. As data points are censored (denoted by the vertical tick marks in the KM curves), there is 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.

6.3 Future Work

[Keanu: Just a couple of ideas so far]

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

Perform separate studies on each hosting service to remove variability in the way the services are utilized.

7 CONCLUSION

REFERENCES

[1] Setup on a PostgreSQL instance âĂŤ Software Heritage - Development Documentation documentation. https://docs.softwareheritage.org/devel/swh-dataset/graph/postgresql.html. (????). (Accessed on 10/15/2021).

[2] Rao Hamza Ali, Chelsea Parlett-Pelleriti, and Erik Linstead. 2020. Cheating Death: A Statistical Survival Analysis of Publicly Available Python Projects. In *Proceedings of the 17th International Conference on Mining Software Repositories*. 6–10.

[3] Rao Hamza Ali, Chelsea Parlett-Pelleriti, and Erik Linstead. 2020. Cheating Death: A Statistical Survival Analysis of Publicly Available Python Projects (MSR 2020 - Mining Challenge) - MSR 2020. https://2020.msrconf.org/details/msr-2020-mining-challenge/1/Cheating-Death-A-Statistical-Survival-Analysis-of-Publicly-Available-Python-Projects. (June 2020). (Accessed on 11/17/2021).

[4] Danilo Caivano, Pietro Cassieri, Simone Romano, and Giuseppe Scanniello. 2021. An Exploratory Study on Dead Methods in Open-source Java Desktop Applications. In *Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 1–11.

[5] David R Cox. 1972. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)* 34, 2 (1972), 187–202.

[6] Edward L Kaplan and Paul Meier. 1958. Nonparametric estimation from incomplete observations. *Journal of the American statistical association* 53, 282 (1958), 457–481.

[7] Riko Kelter. 2020. Bayesian survival analysis in STAN for improved measuring of uncertainty in parameter estimates. *Measurement: Interdisciplinary Research and Perspectives* 18, 2 (2020), 101–109.

[8] Brandon Norick, Justin Krohn, Eben Howard, Ben Welna, and Clemente Izurieta. 2010. Effects of the number of developers on code quality in open source software: a case study. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. 1–1.

[9] 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.

[10] Vinaitheerthan Renganathan. 2016. Overview of frequentist and bayesian approach to survival analysis. *Applied Medical Informatics*. 38, 1 (2016), 25–38.

[11] 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.

[12] Vianda S Stel, Friedo W Dekker, Giovanni Tripepi, Carmine Zoccali, and Kitty J Jager. 2011. Survival analysis I: the Kaplan-Meier method. *Nephron Clinical Practice* 119, 1 (2011), c83–c88.

[13] Vianda S Stel, Friedo W Dekker, Giovanni Tripepi, Carmine Zoccali, and Kitty J Jager. 2011. Survival analysis II: Cox regression. *Nephron Clinical Practice* 119, 3 (2011), c255–c260.

[14] Tianpei Xia, Wei Fu, Rui Shu, and Tim Menzies. 2020. Predicting project health for open source projects (using the DECART hyperparameter optimizer). *arXiv preprint arXiv:2006.07240* (2020).

A ARTIFACTS

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

Data Set: https://annex.softwareheritage.org/public/dataset/graph/latest/popular-3k-python/sql/