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), 12 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.* (referred to as the original authors from here on) 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 hosting service, 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 of interest, such as death) and is often used in the medical field. For example, survival analysis can determine the probability of a patient surviving past a certain time when given a treatment. However, death is not as well defined for a software project as it is for a living organism. A project may not receive revisions for an extended period only to be returned to at a later date, or perhaps a project no longer receives revisions at all, but the community that uses it continues to be active. Samoladas *et al.* considered a project inactive if it received less than two revisions a month; two months of inactivity led to it being considered abandoned or dead [17]. Evangelopoulos *et al.* and the original authors

considered a project dead once there were no revisions at all [2, 7]. The latter definition of project abandonment or death is used in this study to measure the duration of a project or its survival.

The original authors use a frequentist approach to survival analysis utilizing such methods as the Kaplan-Meier (K-M) survival estimator and the Cox Proportional-Hazards model [6, 8]. Though frequentist approaches are considered to be unbiased, minimal in variance, efficient, and generally sufficient, some consider them to lack robustness [16]. 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 [16].

The authors of this paper resonate with the motivation of the original authors. This paper serves as a replication of their paper [2] (referred to as the original paper from here on) and seeks to assess the validity of 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 revision 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 [9] 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 hosting service, and the size of the volunteer team affect the probability of survival of an OSS Python project?

RQ2. How does the revision 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?

Section 2 outlines other research which has utilized survival analysis to study OSS and other work which has studied attributes similar to those studied in this paper. Section 3 describes the source of the data set, the data set itself, and the required preparation in order to perform survival analysis. 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, implications, and suggestions for future work on the topic. The final section concludes by summarizing the purpose and findings of this paper.

2 RELATED WORK

Several other researchers have employed survival analysis to study the health of OSS projects. For example, Samoladas *et al.* studied the affect of application domain and developer count on OSS project health [17]. They found that applications within the domain of *games and entertainment* and *security* had the lowest probability of survival. Additionally, they found that for each new developer introduced to a project, the projects survivability increased by 15.8%. On the topic of developers, several studies have used survival analysis to study developer disengagement from OSS projects [10, 11, 13]. Miller *et al.* made use of a survey and survival analyses, to determine the causes behind why a developer might stop contributing to an OSS project [11]. Their analysis revealed that developers have a higher probability of project disengagement when going through job transitions and when working longer hours. Lin *et al.* determined that developers who balance maintaining files they have created with maintaining files others have created have a higher survival probability than developers who only maintain their own files or only maintain others files [10]. Additionally, Lin *et al.* found that developers who maintained files and developers who mainly wrote code had a higher survival probability than those who solely created files and those whose main focus was writing documentation. Ortega and Izquierdo-Cortazar analyzed the survival of FLOSS committers and Wikipedia editors and found that FLOSS committers have higher mean survival times than Wikipedia editors [13]. Survival analysis can also be applied to the software it

self, this has been demonstrated by Aman *et al.* and Caivano *et al.* [4, 5]. In [4], survival analysis was used to analyze time to bug-fix for files modified by developers of different experience levels. This analysis determined that files which most recently modified by less experienced developers had an increased probability of needing a bug fix within a shorter time frame [4]. Caivano *et al.* explored the affect of dead code within OSS projects using survival analysis [5]. They found that dead methods are present in Java code, persist for a long time (in terms of commits) before being buried or revived, are rarely revived, and that most dead methods have been dead since their inception.

Other studies have examined the health of OSS projects using methods other than survival analysis. Xia *et al.* predicted a number of health indicators of OSS projects, such as, the number of developers and the number of revisions. These predictions were made using regression trees that were optimized using differential evolution, leading to a 10% increase in prediction accuracy over the base line [20]. Norick *et al.* analyzed OSS projects using code quality measures and observed no significant evidence that the number of committing developers affects software quality [12].

3 DATA

Performing survival analysis of OSS projects requires a data set that records the repositories for projects on common hosting sites, 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]. The *popular-3k-python* subset of the Software Heritage graph [15] contains the necessary information and was used in the original paper and also in this paper. This data set contains information on 3052 popular Python projects hosted on GitHub/GitLab, Debian, and PyPI, and records revisions between 1980 and 2019 at the time of writing (the Software Heritage graph is subject to updates, which makes it a non-reproducible data set). Following the tutorial provided by the Software Heritage organization [1], a PostgreSQL database was hosted on the authors' local machines to facilitate data collection.

Though the *popular-3k-python* data set contains all the necessary information to perform survival analysis, it first must be manipulated into a more suitable format before the analysis can be carried out. In this case, the collected data was manipulated such that the final data set contained the duration of the project, the censorship value, and the attributes of interest. Descriptions of each column present in the data set can be found in Table 1. Data collection and manipulation were performed in Jupyter Notebooks, which are available in this paper's repository, a link to which can be found in appendix A.

Table 1. Data Set Column Descriptions

Column Name	Description
Host Type	Which hosting service the project's repository resides on
Major Releases	Whether or not the project publishes major releases
Censored	True if the project's death is not observed (for more information see section 4.1.1)
Duration_months	The duration of the project in months
High_rev_frequency	Whether or not the project has high revision frequency (greater than one revision per day)
Multi_repo	Whether or not the project is hosted on multiple hosting services
High_author_count	Whether or not the project has a high author count (greater than twenty unique authors)

For their study, the original authors 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. After following the same procedures described in the original paper, a list of 2066 projects and their associated information was obtained.

4 METHODS

4.1 Replication

4.1.1 Death and Censoring. Two critical concepts in survival analysis are events of interest and censoring. As previously discussed, the event of interest for this study is project abandonment or death. As defined in the introduction, a project is considered dead when it no longer receives any revisions. With this definition, it is impossible to know whether any project is truly dead, because, unlike a living organism, any project may receive revisions at any point in the future, making it "not dead" by the working definition. However, there are multiple practical ways of determining whether a project is dead, all of which rely on the scope of the studied data set. For this study, the death of a project is determined in the following way:

- Two special revisions are defined for each project
  - Last revision: The last recorded revision of a project within the scope of the data set (i.e. 1980 - 2019).
  - Last observed revision: The last recorded revision of a project within the studied time frame (i.e. 2005 - 2018).
- If, and only if, the last revision is also the last observed revision, the project death is said to be observed.

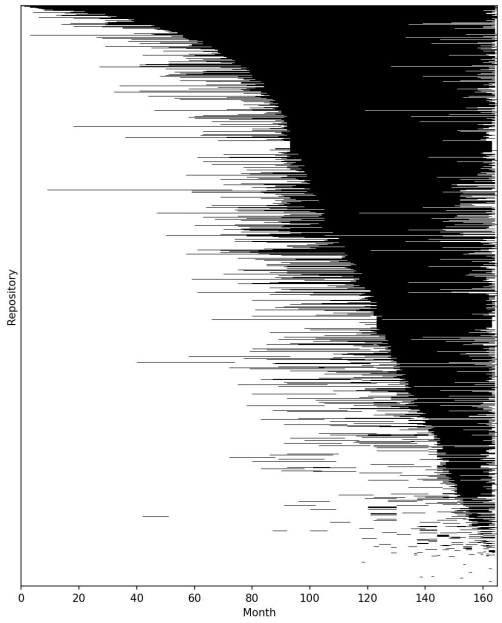

Fig. 1. Graph of project durations within the studied time frame. The projects are ordered by duration and plotted from and to their respective start and end dates. Month 0 begins on May 9, 2005, and Month 165 concludes on January 1, 2018. The black portion of the horizontal lines indicates the active period of a given project

What happens if a project's death is not observed? This is where censoring is used. Suppose a project has its first revision one month before the end of the studied time frame, and continues to be revised daily past the studied time

frame. It would be incorrect to indicate that this project survived for only one month, but it is also impossible to include the project's future activity in the study. Rather than discarding such projects, censoring causes them to be considered by the study for the observed duration, but without considering them as dead projects (i.e. they are removed from the calculations after they stop being observed). There are multiple types of censoring in survival analysis, but this study uses random censoring or type III censoring, which involves removing subjects from a study at varying times relative to when they began to be observed (as is the case here) [16]. If a project's death is not observed, it is considered censored.

The distribution of the project durations over the studied time frame can be seen in Fig. 1, which is a more detailed rendition of Figure 1 in the original paper. Note that when the black lines extend to the end of the time frame, this likely indicates the project was censored. About 62% of the studied projects were censored.

4.1.2 Survival Analysis. 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 [8]. The K-M estimator produces curves that approach the true survival function of the data.

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. 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 [6]. 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.

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 the frequency of contributions as a method of assessment (simply analyzing the number of contributions would not yield useful results given the varying nature of the project durations in the studied time frame). The revision frequency, defined as the number of commits divided by the number of days in the project's observed lifetime, was dichotomized into two groups depending on whether the frequency was above one revision per day. Although the median revision frequency was approximately 0.68 revisions per day, the threshold value of one was chosen because it is easier to remember when keeping these attributes in mind and, similar to the other dichotomizing attributes, it provides a threshold that fewer projects attain to, which sets them apart. This study applies both the K-M estimator and the Cox Proportional-Hazards model to the data to stratify the effects of high revision frequency on the overall health of an open-source project.

4.3 Bayesian Survival Analysis

The Bayesian approach to survival analysis is less common due to computational difficulties. However, it offers multiple advantages over the frequentist approach [9]. We replicate the methods outlined in [9] to apply Bayesian survival

analysis to our study. The Bayesian analysis uses posterior distributions of model parameters to draw inferences about them. These posterior distributions are obtained via Markov-Chain-Monte-Carlo (MCMC) algorithms. The statistical modelling language used was Stan.

Our study applies the same methods as found in the section titled *A detailed example* of [9]. 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 was created. The censoring indicators as $v = (v_1, v_2, \ldots, v_n)$ where $v_i = 0$ if y_i is right censored (lost to follow-up) and $v_i = 1$ if y_i is a failure time (project abandonment), the survival function which is the probability of surviving past the time point y_i is given by

$$S(y_i|\lambda) = P(T \geq yi|T \geq 0) = 1 - [1 - \exp(-\lambda y_i)] = \exp(-\lambda y_i) \tag{1}$$

and the survival model is denoted as

$$y_i|v_i \sim f(y_i|\lambda)^{v_i} + S(y_i|\lambda)^{1-v_i} = [\lambda exp(-\lambda y_i)]^{v_i} + [exp(-\lambda y_i)]^{1-v_i} \tag{2}$$

$$\lambda \sim p(\lambda) \tag{3}$$

$$\lambda = exp(x_i^T \beta) \tag{4}$$

This model was then used to visualize the posterior survival functions for the following five project attributes: major releases, hosting service of the project, use of multiple hosting services, team size, and revision frequency.

5 RESULTS

5.1 Replication

The replication study performed in this paper yielded extremely similar results to those shown in the original paper. Fig. 2 represents K-M curves along with their confidence interval and p-value which implies the survival probability of projects when grouped on the categorical value of each attribute. As seen in Fig. 2a, we found out that projects with at least one releases has higher chances of survival. The curve for projects having major releases plateaus around 65% survival probability during the 120 month mark whereas the survival probability of projects with no major releases end up to be less than 20% by the end of study period. Fig. 2b represents the significance of hosting the project on different hosting services, we found out that projects that are hosted on GitHub has a higher survival chances in the long run, as the curves suggest all three hosting services has similar trend for first 55 months which is within the average duration of projects hosted on these services. We observed that having multiple repositories hosted on multiple services has significantly increased the chances of a project's survival. As seen in Fig. 2c, the survival rate for such projects is 75%, whereas around 20% for projects with only one package repository system. The Curve in fig. 2d implies the significance of network of developers for survival of open source project, clearly the project having more than 20 different authors end up having around 60% survival rate which are significantly different form projects with small team of developers with less than 20% survival rate in the end of the study period.

The Table in figure 3 quantify estimates of these attribute's effect on the survival probability using the Cox Proportional-hazards model. Third column shows the hazards ratio which indicates the probability of abandonment with respect to the reference feature. In the first raw we can the hazard ratio for projects not having a major release with respect to projects having major releases which is nearly 3, it implies that the projects without major releases are three times more likely to become inactive compared to projects with at least one major release. Similarly, projects

having only one repository system are 3.3 times more likely to be abandoned. The third row highlights that the projects with less developers count are 19.3 times more likely be inactive. For the type of hosting used, we see that ratio implies that projects hosted on PyPi or Debian are less likely to be abandon compared to projects that are hosted on GitHub. This could be because Debian and PyPi-based projects have a higher survival rate during the first 55 months, the cox model considers this portion where most of the projects were participating heavily.

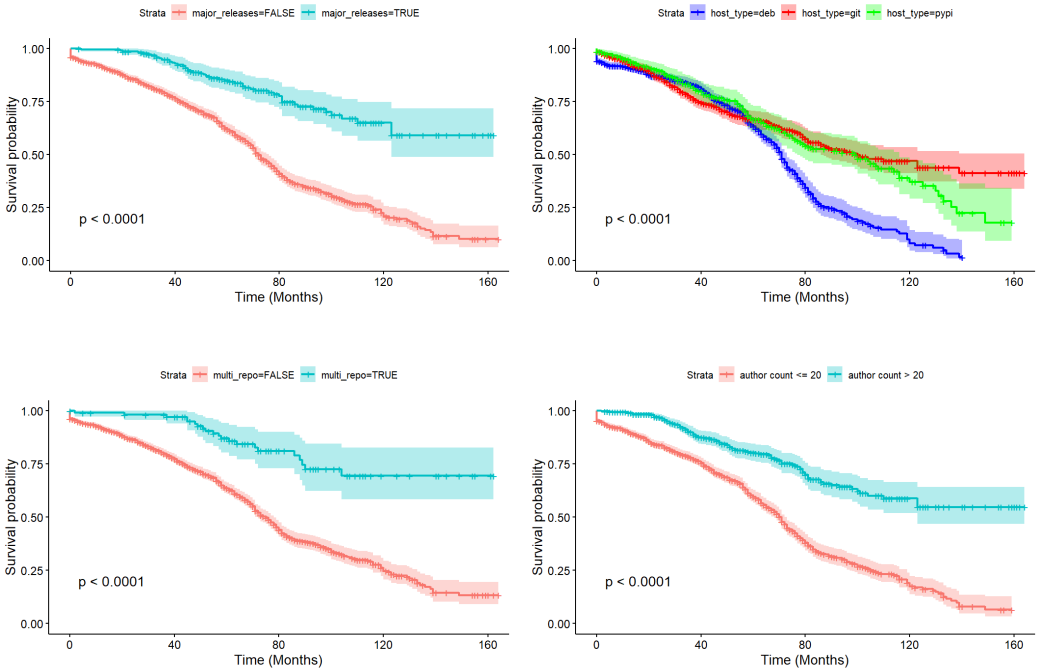

Fig. 2. KM curves of the python project data analyzed by major releases, host type, multiple repository hosting, and number of authors

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

5.2 Revision Frequency Analysis

The K-M curves for revision frequency are little different from the graphs we generated for other attributes, as seen in Fig 4. We observed curves drop for the projects with more than one revision per day in the starting of study period. In Addition, the projects with less revision frequency out performed projects with higher revision frequencies during the mean duration period of the studied projects. As shown in the Fig. 3, results of our regression model also indicate that the projects with less revision frequency are less likely to be abandoned. This may be due to cox proportional model gives more consideration to the portion where most of the projects were active.

[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

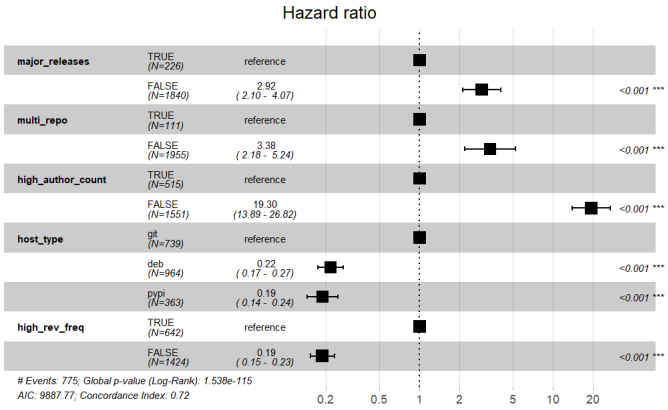

Fig. 3. The results of running the Cox Proportional hazards tool on the data. **[Keanu: we definitely want better descriptions for the images]**

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.]

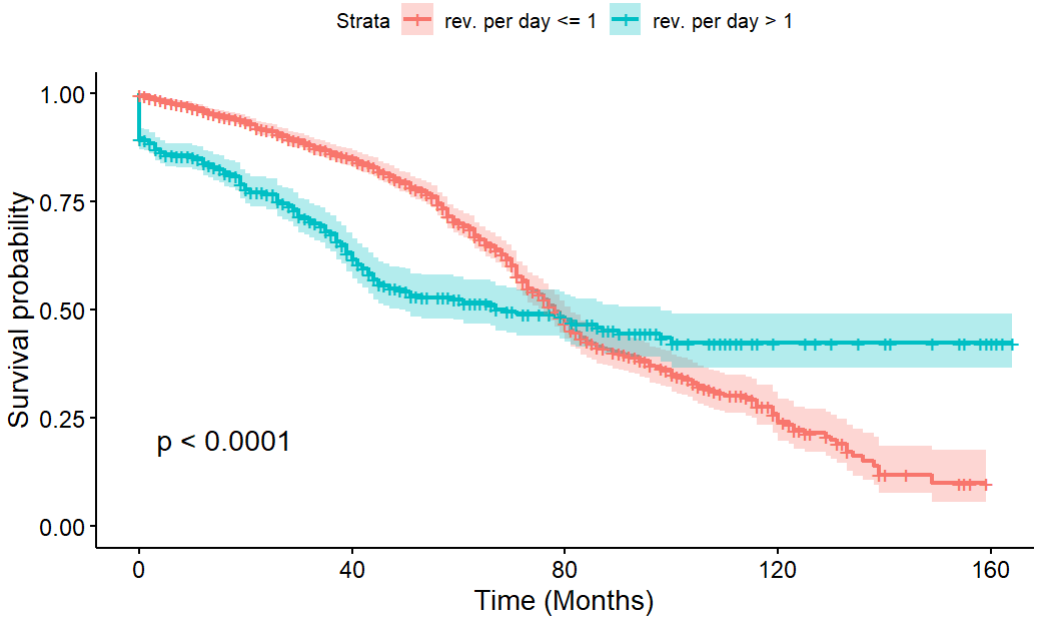

Fig. 4. KM curve for the data by revision frequency **[Keanu: need a better caption]**

5.3 Bayesian Survival Analysis

Shown in figure 5 are the posterior survival functions for variations of the selected project attributes. The dotted lines represent the 2.5% and 97.5% quantiles, while the solid middle line represents the posterior mean of β (the prior on the project attribute of interest). The remaining lines all represent valid posterior survival functions. Fig. 5a clearly indicates that the survival function of the projects with no major releases decreases much faster than projects with releases. Survival probability for projects with no releases was lower than 25% after 150 months compared to 55% for projects with releases. Fig. 5b illustrates that projects hosted on Github have the highest survival chances compared to those hosted on Pypi and Debian. At 150 months, the predicted survival probabilities for Github, Pypi and Debian were 87, 65 and 25, respectively. Fig. 5c shows that projects hosted on more than 1 version control system had significantly higher survival chances than projects hosted on a single hosting service. Above 65% survival probability for multiple repositories compared to slightly over 25% for single repository projects at the end of 150 months. Fig. 5d demonstrates the importance of the number of contributing developers on the survival chances of open-source python projects. For projects with more than 20 authors, the predicted survival probability was around 55%, while less than 25% for projects with less than 20 authors at the 150-month mark. Additionally, for all project attributes, the beta value for the 97.5% quantile was found smaller than zero, which ensures that estimates are highly certain [9].

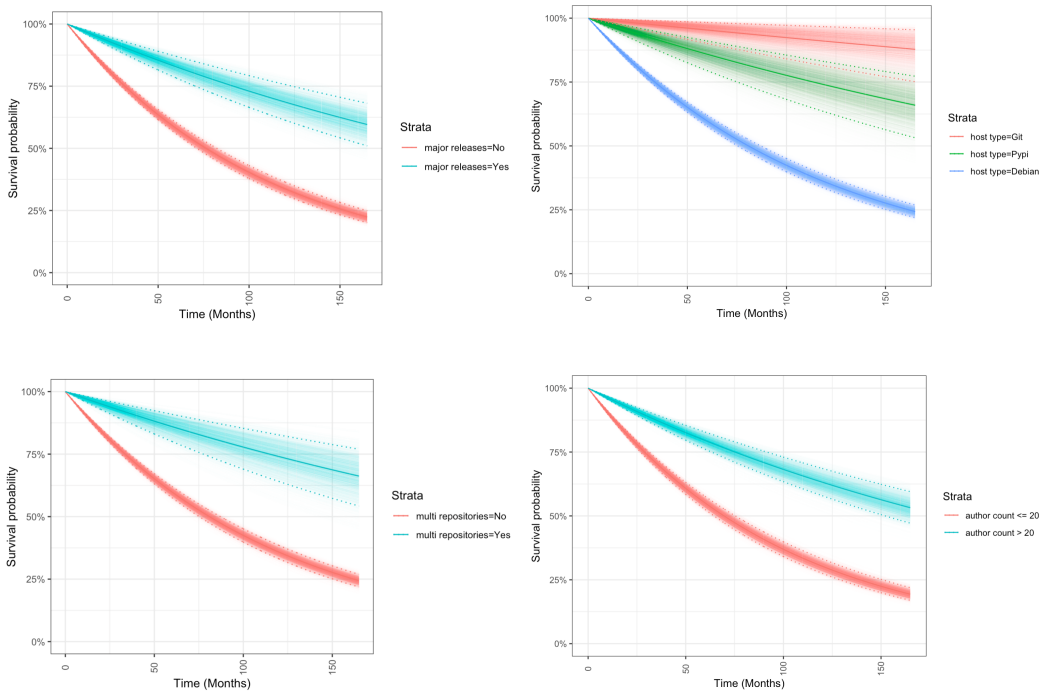

Fig. 5. Bayesian posterior survival functions of the python projects analyzed by major releases, host type, multiple repository hosting, and number of authors

6 DISCUSSION

6.1 Implications

6.2 Comparison of Analyses

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

6.3 Limitations

6.3.1 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.

The original authors left out many details regarding the methods they used for data extraction, manipulation, and analysis. This led to assumptions being made when replicating their study. Hence, there are discrepancies in the values obtained compared to the original paper.

Being a replication, this study was limited to choosing the methods used by the original paper. These methods are also unanimous in the area of survival analysis, and there are few alternatives to them. This study attempts to address these limitations by providing an alternative analysis method, namely Bayesian analysis.

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 [18]. Additionally, the K-M estimator does not account for confounding factors [18]. 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 [19]. 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 hazard ratio between two groups [19]. Looking at the K-M curves for this study, it can be concluded that the proportional hazards assumption does not hold as the survival functions diverge over time and cross over each other rather than running in parallel [14]. This explains the discrepancies between the results of the cox regression model and the K-M curves. Future studies should perform tests to determine whether the assumptions for their models hold and should seek methods for mitigating such errors through identifying time-dependant covariates.

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 [16]. 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]**

6.3.2 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 [15]. 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 revision frequency. Additionally, the potential confounding factors mentioned in section 6.3.1 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 the original authors 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.

6.4 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.

[Derek: Future work should include a more in depth analysis of whether or not the assumption of the cox proportional hazards model holds]

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] 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.

[5] 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.

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

[7] 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 arXiv:https://doi.org/10.1080/03610910802140224

[8] E. L. Kaplan and Paul Meier. 1958. Nonparametric Estimation from Incomplete Observations. *J. Amer. Statist. Assoc.* 53, 282 (1958), 457–481. https://doi.org/10.1080/01621459.1958.10501452 arXiv:https://www.tandfonline.com/doi/pdf/10.1080/01621459.1958.10501452

[9] 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.

[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.

[11] 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.

[12] 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.

[13] 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.

[14] Inger Persson and Harry Khamis. 2007. A Comparison of Graphical Methods for Assessing the Proportional Hazards Assumptions in the Cox Model.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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).

APPENDICES

A ARTIFACTS

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

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

B WORK DISTRIBUTION