Two Approaches to Survival Analysis of Open Source Python Projects

Derek Robinson
drobinson@uvic.ca
University of Victoria
Computer Science
PO Box 1700 STN CSC
Victoria, British Columbia, Canada V8W 2Y2

Keanelek Enns
keanelekenns@uvic.ca
University of Victoria
Computer Science
PO Box 1700 STN CSC
Victoria, British Columbia, Canada V8W 2Y2

Neha Koulecar
nehakoulecar@uvic.ca
University of Victoria
Computer Science
PO Box 1700 STN CSC
Victoria, British Columbia, Canada V8W 2Y2

Manish Sihag
manishsihag@uvic.ca
University of Victoria
Computer Science
PO Box 1700 STN CSC
Victoria, British Columbia, Canada V8W 2Y2

ABSTRACT

A recent study applied frequentist survival analysis methods to a subset of the Software Heritage Graph and determined which attributes of an OSS project contribute to its health. This paper serves as a replication of that study. In addition, we apply Bayesian survival analysis methods to the same dataset and investigate an extra project attribute. Both analyses focus on the effects of certain attributes on the survival of open-source software projects as measured by their revision activity. Methods such as the Kaplan-Meier estimator, Cox Proportional-Hazards model, and the visualization of posterior survival functions were used for each of the project attributes. The results show that projects which publish major releases, have repositories on multiple hosting services, possess a large team of developers, and make frequent revisions have a higher likelihood of survival in the long run. The findings were similar to the original study; however, a deeper look revealed quantitative inconsistencies.

CCS CONCEPTS

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

KEYWORDS

data science, survival analysis, open source, python, Kaplan Meier, Cox Proportional-Hazards model, Bayesian analysis, frequentist

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 [21]. Evangelopoulos *et al.* [9] and the original authors [2] considered a project dead once there were no revisions at all. 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 [10] and the Cox Proportional-Hazards model [8]. Though frequentist approaches are considered to be unbiased, minimal in variance, efficient, and generally sufficient, some consider them to lack robustness [20]. 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 [20].

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. In addition to studying the same attributes as the original authors, we also study the revision frequency of a given project. Furthermore, this paper analyzes the same data set using a Bayesian approach to survival analysis as outlined in [11] 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 those of Bayesian survival analysis?

The remainder of the paper is structured as follows. 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, Bayesian analysis, and the additional attribute analysis. Section 5 shows the results for each analysis. Section 6 discusses the results, implications, and limitations of the analyses, and gives suggestions for future work. 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 effect of application domain and developer count on OSS project health, which was measured using project duration [21]. 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 [12, 14, 16]. 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 [14]. 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 maintained files created by both themselves and others have a higher survival probability than developers who only maintain their own files or only maintain others files [12]. 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 [12]. Ortega and Izquierdo-Cortazar analyzed the survival of OSS committers and Wikipedia editors and found that OSS committers have higher mean survival times than Wikipedia editors [16].

Survival analysis can also be applied to the software itself, this has been demonstrated by Aman *et al.* [4] and Caivano *et al.* [6]. Aman *et al.* used survival analysis to analyze the time to a bug-fix for files modified by developers of different experience levels [4]. This analysis determined that files which were most recently modified by less experienced developers had an increased probability of needing a bug fix within a shorter time frame. Caivano *et al.* explored the effect of dead code within OSS projects using survival analysis [6]. They found that dead methods are present in Java code, persist for a long time 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 [24]. Norick *et al.* analyzed OSS projects using code quality measures and observed no significant evidence that the number of committing developers affects software quality [15].

3 DATA SET

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 (referred to as revisions from here on) and major releases (revisions of note, often with a specific name and release date) [2]. The *popular-3k-python* subset [1] of the Software Heritage Graph [18] contains the necessary information and was used in the original paper and also in this paper. This data set contains information on 3,052 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 [2], 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

---

[1]https://annex.softwareheritage.org/public/dataset/graph/latest/popular-3k-python/sql/
[2]https://docs.softwareheritage.org/devel/swh-dataset/graph/postgresql.html

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 the replication package [1].

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, as these were not given in the original paper. Using January 1, 2018, as a strict end date and maintaining the study duration as 4,620 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 2,066 projects and their associated information was obtained.

4 METHODS

This section outlines how the data was transformed (Sections 4.1.1 and 4.3) and gives brief introductions to each of the statistical methods used (Sections 4.1.2 and 4.2). All research artifacts can be found in the replication package [1].

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 by first defining two special revisions for each project: *last revision* and *last observed revision*. *Last revision* is defined as the last recorded revision of a project within the scope of the data set (i.e., 1980 - 2019). *Last observed revision* is defined as 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.

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) [20]. 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 Figure 1, which is a more detailed rendition of
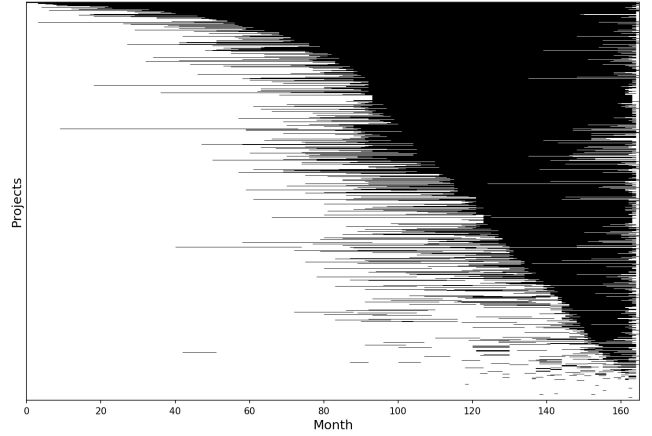

Figure 1: Project durations within the studied time frame. The projects are ordered by duration and plotted from and to their respective start and end dates. Month 0 begins on May 9, 2005, and Month 165 concludes on January 1, 2018. The black portion of the horizontal lines indicates the active period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

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. Overall, 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 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 until a particular time t. At $t = 0$, the K-M estimator is 1 and as t approaches infinity, the K-M estimator approaches 0. 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 ith time step [10]. The K-M estimator produces curves that approach the true survival function of the data. In other words, K-M curves plot the K-M survival probability against time. The curve always goes in "steps" as the cumulative survival remains the same until the time of another event (in this case project abandonment). Furthermore, the censored observations are indicated by a vertical dash.

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 at a certain point in time given that the subject has survived to that time [7]. 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 [8]. 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.

Table 1: Data Set Column Descriptions

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

4.2 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 [11]. This study replicates the methods outlined in [11] to apply Bayesian survival analysis to the data set. 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.

This study applies the same methods as found in the section titled *A detailed example* of [11]. A parametric exponential model that assumes the survival times $y = (y_1, y_2, \ldots, y_n)$ for the set of projects are exponentially distributed with parameter λ was created as shown in Equation 1.

$$f(y_i|\lambda) = \lambda \exp(-\lambda y_i) \text{ for } i = 1, \ldots, n \quad (1)$$

The censoring indicators are denoted as $v = (v_1, v_2, \ldots, v_n)$ where $v_i = 0$ if y_i is right censored (project death is not observed) and $v_i = 1$ if y_i is a failure time (project death is observed). The survival function, which is the probability of surviving past the time point y_i, is given by Equation 2.

$$S(y_i|\lambda) = P(T \geq y_i|T \geq 0) = 1 - [1 - \exp(-\lambda y_i)] = \exp(-\lambda y_i) \quad (2)$$

The combination of Equations 1 and 2 yields the survival model given by Equations 3 through 6.

$$y_i|v_i \sim f(y_i|\lambda)^{v_i} + S(y_i|\lambda)^{1-v_i} = [\lambda exp(-\lambda y_i)]^{v_i} + [exp(-\lambda y_i)]^{1-v_i} \quad (3)$$

$$\lambda \sim p(\lambda) \quad (4)$$

$$\lambda = exp(x_i^T \beta) \quad (5)$$

$$\beta = normal(0, 10) \quad (6)$$

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.

4.3 Revision Frequency Analysis

The original paper mentions that "The health of a project could be computed by the number and frequency of contributions..." [2] 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 frequentist and Bayesian analysis methods to the data to stratify the effects of high revision frequency on the overall health of an open-source project.

5 RESULTS

This section describes the results for each portion of the study. Section 5.1 describes the results for the replication portion of the study. Sections 5.2 and 5.3 describe the results of the Bayesian survival analysis and the revision frequency analysis respectively.

5.1 Replication

The replication study performed in this paper yielded extremely similar results to those shown in the original paper. Figure 2 depicts K-M curves along with their confidence intervals and p-values. The p-values imply that the difference in survival probability for projects within each group is statistically significant. As seen in Figure 2a, this study found that *projects with at least one major release have higher chances of survival.* The curve for projects with at least one major release plateaus around 65% survival probability around the 120-month mark, whereas the survival probability of projects with no major releases ends up less than 20% by the end of the study period. Figure 2b represents the significance of the type of hosting service used. *It was observed that projects that are hosted on GitHub have higher survival chances in the long run,* though the curves suggest all three hosting services have a similar trend for the first 55 months, which is within the average duration of projects hosted on these services. In addition, *having multiple repositories hosted on multiple services significantly increased the chances of a project's survival.* As seen in Figure 2c, the survival rate for such projects is close to 70% by the end of the study period, whereas it is around 20% for projects with only one package repository system. The curve in Figure 2d illustrates the effect of the size of the network of developers, *the projects with more than 20 different authors end up having a survival probability of about 60%*
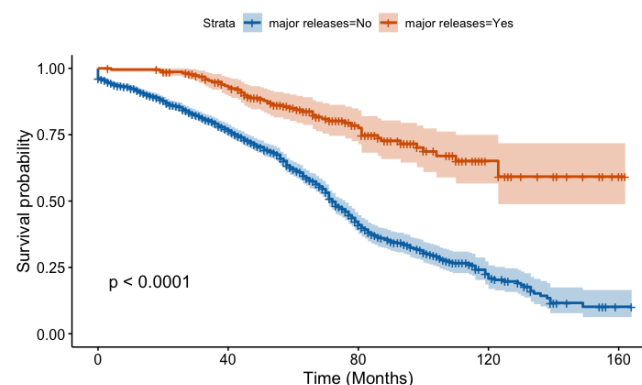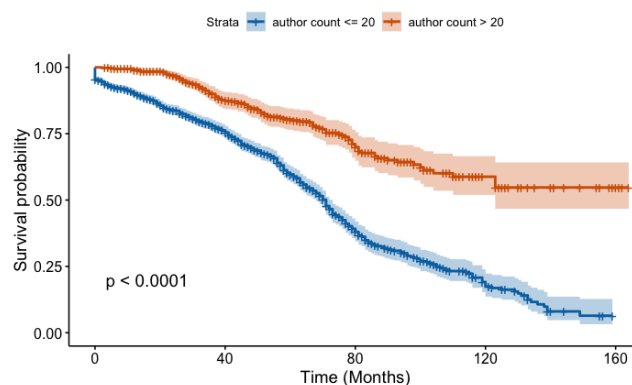
(a) K-M curves for projects which publish major releases and those which do not
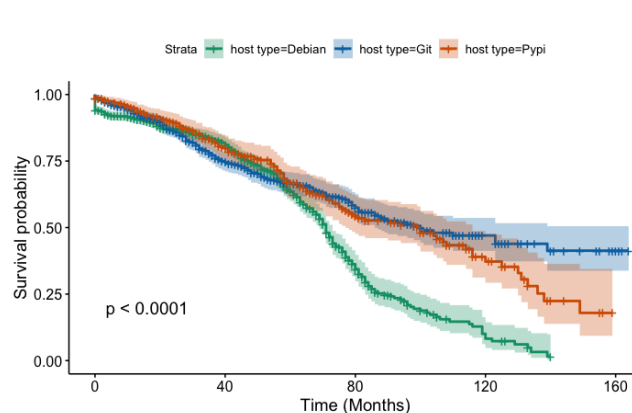
(b) K-M curves for projects with different hosting services
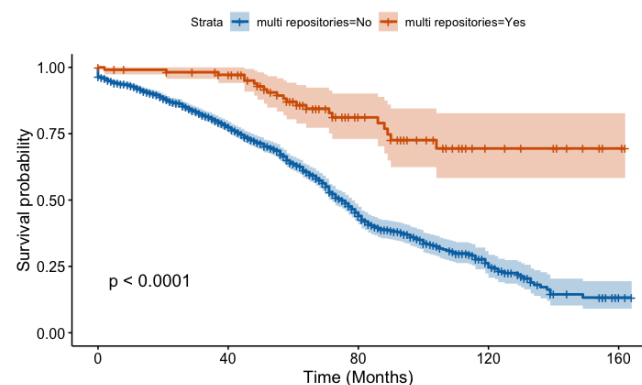
(c) K-M curves for projects with repositories on multiple hosting services

(d) K-M curves for projects with high and low author count

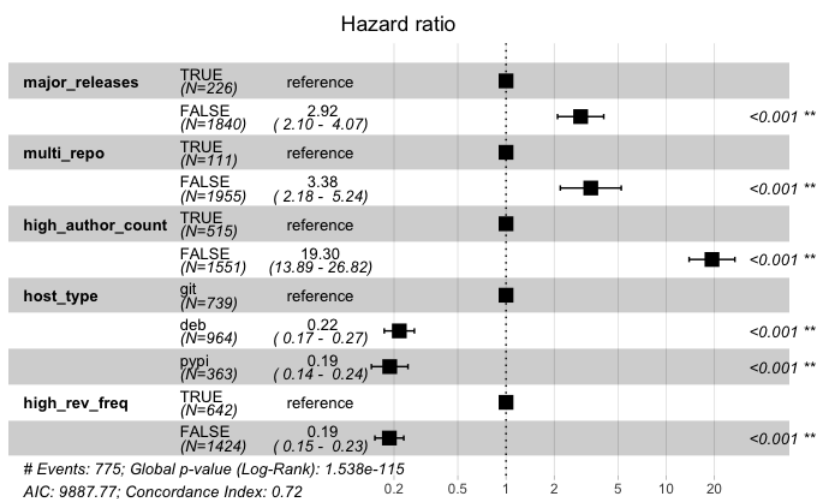Figure 2: K-M curves for each project attribute

Figure 3: The Cox Proportional-Hazards model. From left to right: project attribute, attribute value with counts, hazard ratio, box plot of hazard ratio, and p-value of log rank test.

compared to a 20% survival probability for projects with a small team of developers. Figure 3 presents the results of the Cox Proportional-Hazards model which quantifies the effect of these attributes on project survival probability. The third column shows the hazard ratio which indicates the probability of abandonment with respect to the reference feature. In the first row, the hazard ratio for projects which do not publish major releases with respect to projects that do, is nearly 3. This 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 with repositories on a single hosting service are 3.3 times more likely to be abandoned.* The third row highlights that *the projects with fewer developers are 19.3 times more likely to be inactive.* For the type of hosting service used, the ratio implies that *projects hosted on PyPI or Debian are less likely to be abandoned compared to projects that are hosted on GitHub.* This appears to contradict the results of the K-M curve and will be further discussed in Section 6.

5.2 Bayesian Survival Analysis

Figure 4 shows 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 represent all valid posterior survival functions. Figure 4a 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 major releases was lower than 25% after 150 months compared to 55% for projects with major releases.* Figure 4b 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. Figure 4c shows that *projects with repositories on more than one hosting service had significantly higher survival chances than projects with repositories 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. Figure 4d 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 smaller than zero, which ensures that the estimates for λ have low uncertainty [11].

5.3 Revision Frequency Analysis

The K-M curves for revision frequency are different from the graphs generated for the other attributes, as seen in Figure 5a. A curve drop was observed for the projects with more than one revision per day in the beginning of the study period. In addition, the projects with lower revision frequencies out performed projects with higher revision frequencies during the mean duration period of the studied projects. Figure 5b shows the posterior survival functions for projects with more than one revision per day and those with revision per day less than or equal to one. Although a significant difference was not observed in predicted survival probabilities,

projects with more than one revision per day had a slightly higher chance of survival by the end of the study period. As shown in Figure 3, the results of the regression model indicate that the projects with a higher revision frequency are more likely to be abandoned.

6 DISCUSSION

The replication was deemed successful in that the results were consistent with the qualitative findings of the original authors with respect to the first research question. As in other works [21], the number of developers was found to be a significant indicator of a project's health in terms of duration (note the smaller confidence interval for Figure 2d), and this result lends support to "Linus's Law", which claims that a larger number of developers will make bug detection and resolution simpler [19], though this study does not make claims about a project's quality. Interestingly, Norick *et al.* observed no significant evidence that software quality is affected by the number of developers [15]. This suggests a higher number of developers has an effect on the longevity of a project that is unrelated to the quality of the software.

The Cox Proportional-Hazards model operates under the assumption that the hazard ratio between the hazard functions of each group remains the same throughout the period of the study [23]. 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 [17]. This means the hazard ratios are not applicable to the entire studied time frame. After inspecting the hazard ratios in Figure 3, it is assumed the calculations in the Cox model give more weight to the portion of the study containing a larger number of subjects (i.e., the left most portion of the K-M curves). This partially explains why there is such a large discrepancy between the hazard ratio for the author count attribute found in this paper and the one found in the original paper (19.3 compared to 5.95).

When comparing the results of the Bayesian analysis to those of the frequentist analysis, it is clear that the Bayesian survival functions reveal less information about the data set. This is to be expected, however, as the models require the data to conform to the chosen prior distribution function and cannot reflect more complex detail. This comparison is especially apparent when looking at Figure 5. One of the advantages of Bayesian models is their ability to create predictions in the presence of new data. Whereas the frequentist approach attempts to reveal exactly what the data set is describing, this would be considered overfitting in the context of Bayesian analysis. Though the approaches are distinct and serve different purposes, it is clear that the survival curves generated by both approaches share the same sentiment with respect to the project attributes.

The survival functions displayed in Figure 5 are distinct when compared with the other results of the study. There is a noticeable drop in the K-M curve belonging to projects with high revision frequencies within the first few months of the studied time period. This is likely due to the presence of many short-lived projects with rapid development times. If such projects were removed from the study, the projects with high revision frequencies would perform even better. Additionally, it can be seen that, within the mean
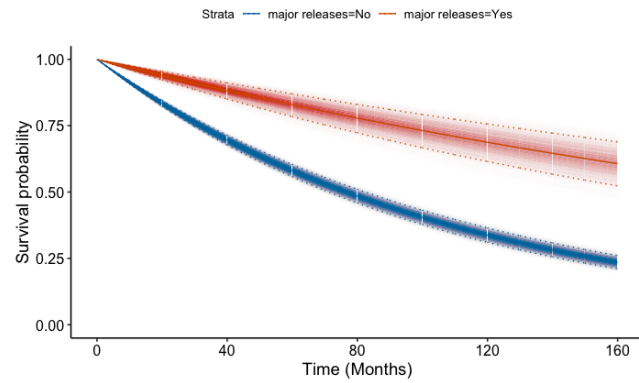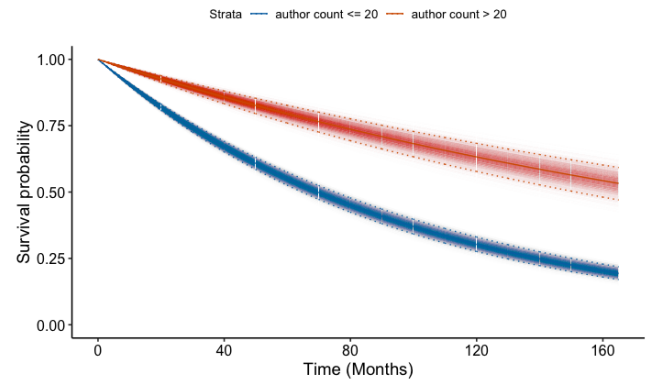
(a) Posterior survival functions for projects which publish major releases and those which do not
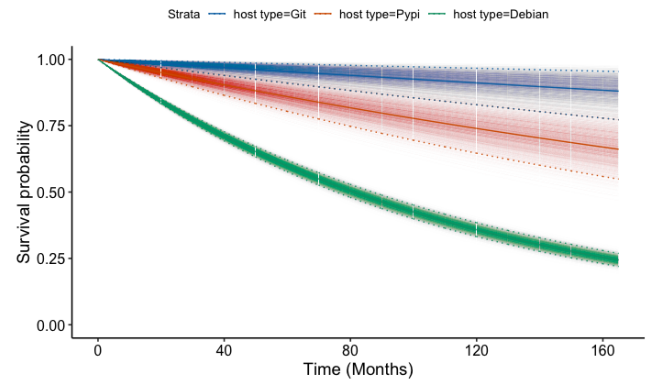
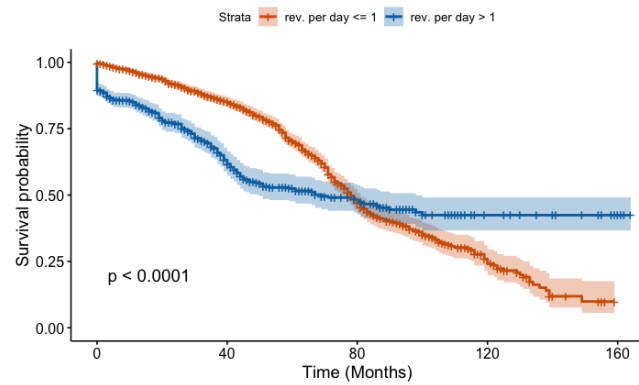(b) Posterior survival functions for projects with different hosting services
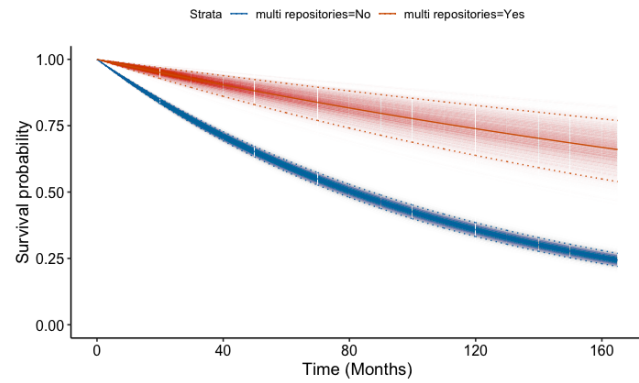
(c) Posterior survival functions for projects with repositories on multiple hosting services
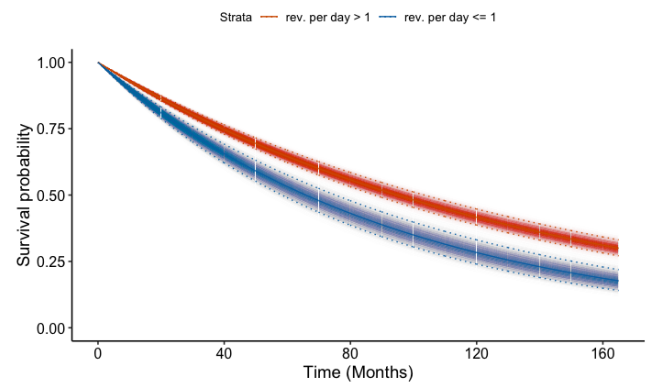
(d) Posterior survival functions for projects with high and low author count

Figure 4: Bayesian posterior survival functions for each project attribute

Figure 5: K-M (a) and Bayesian posterior survival function (b) curves for revision frequency

project duration (about 46 months), the projects with higher revision frequencies are abandoned more frequently, but after this point their curve begins to plateau while the other curve begins to decline more rapidly. This suggests that *having a higher revision frequency benefits a project's probability of survival in the long run and may indicate that consistency is more important*, though the metric used for this study does not consider how evenly distributed the revisions of a project are.

Finally, it is hypothesized from the K-M curves of every binary project attribute (Figures 2 and 5) that somewhere around the 60 month point represents a critical threshold for projects, after which, the attributes outlined in this study have an increasingly significant effect. More work should be done to investigate whether this is truly the case.

6.1 Implications

6.1.1 Implications for Software Practitioners.
The results of this study, and the studies it is based on, have practical implications. They give developers of open-source software insights into which projects are likely to receive long term attention from other developers, they help project coordinators understand the attributes that their projects should possess if they want a long-lasting project, and they give companies and organizations looking to invest in, utilize, or rely on OSS projects confidence that such projects will remain active.

6.1.2 Implications for Researchers.
Though this paper's results were qualitatively similar to those of the original authors, it is suspected that a variation in approaches for data retrieval and the analysis tools used caused quantitative discrepancies (as seen in the hazard ratio for the author count attribute). The authors of this paper therefore call for researchers to improve the reproducibility of their studies by providing research artifacts.

The application of Bayesian survival analysis presented in this paper is meant to serve as a preliminary investigation of the approach in the context of software duration. It is suggested that researchers looking to use this method in the future take a deeper look into priors, likelihoods, and covariates. Researchers may also wish to test the validity of the models produced in this study by using them to predict software duration for new data sets.

6.2 Limitations

6.2.1 Limitations of the Methods.
The original paper [2] and the corresponding 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 [22]. Additionally, the K-M estimator does not account for confounding factors [22]. 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.

As discussed in the start of Section 6, the Cox Proportional-Hazards model is used with the assumption that, over the period of observation, the hazards within each group are proportional. If the assumption is not true, then the Cox Proportional-Hazards model will lead to incorrect estimates of the hazard ratio between two groups [23]. The assumption does not hold in this study, which explains the discrepancies between the results of the Cox regression model and the K-M curves. It is unclear whether these assumptions are reasonable in the context of software duration. 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 [20]. To mitigate this bias, prior selection requires both epistemological and ontological reasoning. Prior distributions were chosen based on survival analysis done in other domains [11, 13], but more investigation should be done as to whether these priors are appropriate for this domain and whether the models from this study accurately predict durations of different data sets.

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 [18]. 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 [5].

There are inherent differences in the ways developers use the different hosting services. Traditionally, services such as PyPI and Debian are 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 6.2.1 are not represented in this data set.

It may also be worth noting that this data is only for Python projects and 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. The results of the revision frequency analysis in section 5.3 seem to indicate a large number of short lived projects.

The data set 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 K-M 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. Finally, the data set contained many revisions (over 4 million) that were not associated with project URLs, the cause of this is unclear.

6.3 Future Work

It can be said with high confidence that this is the first application of Bayesian survival analysis in the context of OSS. As such, prior distributions were uninformative and priors and likelihoods alike were chosen based on the application of Bayesian survival analysis done in other domains. More investigation should be conducted to determine if the priors and likelihoods applied in this paper are suitable for OSS. Furthermore, future work should investigate the predictive power of the Bayesian survival models presented here to determine their performance on other data sets. Similarly, an evaluation of the assumptions present in the Cox Proportional-Hazards model should be performed to determine if these assumptions are reasonable for the domain of OSS.

In regards to the study design, future work should increase the time frame of the study to determine how the attributes analyzed in this paper play out over a longer time period. Similarly, as only OSS written in Python was studied here, future work should evaluate the survival of OSS written in other popular languages such as Java.

As noted in the discussion, a higher number of developers has some effect on the longevity of a project that is not related to the quality of the software being developed. Future works should investigate the reason behind this increased longevity and attempt to determine a reason for it. In line with Samoladas *et al.* [21], distinguishing between core developers and sporadic ones could lead to an increased understanding of the aforementioned phenomena.

7 CONCLUSION

The work presented in this paper aimed to replicate that of the original authors [2]. In addition, one more project attribute was analyzed, the effect of revision frequency on open-source software project duration. Furthermore, Bayesian survival analysis was used to study the attributes in the original paper as well as revision frequency. While the Kaplan Meier curves are consistent with those of the original paper, the Cox Proportional-Hazards Model indicated a quantitative discrepancy with regard to the author count hazard ratio. This discrepancy is likely due to a difference in methods which could have been mitigated had the original paper provided research artifacts. It was observed that a higher revision frequency leads to a better chance of survival by the end of the study duration. The Bayesian survival analysis yielded similar results to the frequentist approach. However, it is worth noting that the results obtained from the Bayesian approach were much less granular and are intended to predict new data rather than accurately describe the studied data.

REFERENCES

[1] 2022. Two Approaches to Survival Analysis of Open Source Python Projects. Zenodo. https://doi.org/10.5281/zenodo.6040657
[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*. Association for Computing Machinery, New York, NY, USA, 6–10. https://doi.org/10.1145/3379597.3387511
[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 *Product-Focused Software Process Improvement*, Michael Felderer, Daniel Méndez Fernández, Burak Turhan, Marcos Kalinowski, Federica Sarro, and Dietmar Winkler (Eds.). Springer International Publishing, Innsbruck, Austria, 80–88.
[5] Christian Bird, Peter C Rigby, Earl T Barr, David J Hamilton, Daniel M German, and Prem Devanbu. 2009. The promises and perils of mining git. In *2009 6th IEEE International Working Conference on Mining Software Repositories*. IEEE, Vancouver, Canada, 1–10.
[6] 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)*. Association for Computing Machinery, New York, NY, USA, Article 10, 11 pages. https://doi.org/10.1145/3475716.3475773
[7] Taane G Clark, Michael J Bradburn, Sharon B Love, and Douglas G Altman. 2003. Survival analysis part I: basic concepts and first analyses. *British journal of cancer* 89, 2 (2003), 232–238.
[8] David R Cox. 1972. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)* 34, 2 (1972), 187–202.
[9] 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
[10] 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
[11] 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.
[12] 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, Buenos Aires, Argentina, 66–75. https://doi.org/10.1109/ICGSE.2017.11
[13] Richard McElreath. 2020;2016;2015;. *Statistical rethinking: a Bayesian course with examples in R and Stan* (second;1; ed.). Vol. 122. Chapman & Hall/CRC, Boca Raton.
[14] 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 *Open Source Systems*, Francis Bordeleau, Alberto Sillitti, Paulo Meirelles, and Valentina Lenarduzzi (Eds.). Springer International Publishing, Cham, 116–129.
[15] 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 (ESEM '10)*. Association for Computing Machinery, New York, NY, USA, Article 62, 1 pages. https://doi.org/10.1145/1852786.1852864
[16] 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, Washington, DC, USA, 7–12. https://doi.org/10.1109/FLOSS.2009.5071353
[17] Inger Persson and Harry J Khamis. 2007. A comparison of graphical methods for assessing the proportional hazards assumptions in the Cox model. *Journal of Statistics and Applications* 2, 1-4 (2007), 1.
[18] 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, Montreal, QC, Canada, 138–142. https://doi.org/10.1109/MSR.2019.00030
[19] Eric S. Raymond. 2001. *The cathedral and the bazaar: musings on Linux and Open Source by an accidental revolutionary* (rev. ed.). O'Reilly, Beijing;Cambridge, Mass;.
[20] Vinaitheerthan Renganathan. 2016. Overview of frequentist and bayesian approach to survival analysis. *Applied Medical Informatics*. 38, 1 (2016), 25–38.
[21] 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.

[22] 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.

[23] 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.

[24] 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).