Predicting the time-to-deliver of software changes

by

Sokratis Tsakiltsidis

Bachelor of Science in Information Technology, Alexander Technological Institute of Thessaloniki, 2012

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the Program of

Computer Science

Toronto, Ontario, Canada, 2016

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my dissertation may be made electronically available to the public.

Predicting the time-to-deliver of software changes

Master of Science 2016

Sokratis Tsakiltsidis

Computer Science

Ryerson University

Abstract

In this thesis we examine the application of survival analysis on time-to-deliver data. Successful prediction of the time necessary to deliver a new feature or fix a reported defect can assist in various phases and aspects of software development. We identify and try to overcome limitations when dealing with time-to-event data. Our proposed methodological framework includes use of survival analysis, utilization of incomplete information that might be available as censored data, and incorporation of random-effects through mixed-effects models for identification of hierarchical/clustered data within our dataset. We explore and experiment with a dataset from a large scale commercial software over a twelve year period of time. We show that we can successfully implement survival analysis, and that incorporation of random-effects provides a considerable advantage, however, incorporation of censored information is not proven to be advantageous in this case.

Acknowledgements

This Master dissertation is submitted to fulfill the requirements of the MSc of Computer Science at Ryerson University in Toronto, Canada. The work carried out in this dissertation has been has been supervised by Dr. Andriy Miranskyy.

Firstly, I would like to express my sincere gratitude to my supervisor for his continuous support, understanding, and valuable advice. His guidance and mentorship helped me in all the time of research and writing of this thesis.

Besides my supervisor, I would like to thank Dr. Petros Pechlivanoglou for his insightful guidance and amazing support during my period of study.

Additionally, I would like to thank my thesis examination committee: Dr. Chen Ding and Dr. Vojislav Mišić, for agreeing to read and provide their feedback on this thesis.

Last but not least, I would like to thank my family, my friends, and of course Lila for their support, but most importantly for their patience while fulfilling the requirements of this degree.

Dedication

To my father.

Contents

Declaration . . . iii
Abstract . . . v
Acknowledgements . . . vii
Dedication . . . ix
List of Tables . . . xv
List of Figures . . . xvii

1 Introduction **1**
1.1 Motivation and problem statement . . . 1
1.2 Objective . . . 2
1.3 Proposed Solution . . . 2
1.4 Novelty . . . 4
1.5 Contribution . . . 4
1.6 Outline . . . 5

2 Literature Review **7**
2.1 Software quality assurance . . . 7
2.2 Number of bugs remaining . . . 9
2.3 Time-to-deliver estimation . . . 10

3 Methodology and Implementation **15**
3.1 Theoretical framework . . . 15
3.2 Regression analysis . . . 17
3.2.1 Linear Regression . . . 18
3.3 Regression estimation . . . 19
3.3.1 Ordinary Least Squares . . . 19

3.3.2 Maximum Likelihood 20
3.4 Understanding the limitations 21
3.4.1 Censoring 21
3.4.2 Skewness 23
3.5 Survival Analysis 25
3.5.1 Survival function 26
3.5.2 Hazard function 27
3.5.3 Non-parametric estimators 27
3.5.4 Semi-parametric estimation 28
3.5.5 Parametric estimation 29
3.6 Mixed-effects models 32
3.6.1 Fixed-effect models 32
3.6.2 Random-effect models 33
3.7 Diagnostics and validation 34
3.7.1 Akaike Information Criterion 35
3.7.2 R-squared 36
3.7.3 Residual standard deviation 36
3.7.4 Kendall rank correlation coefficient 37
3.7.5 Accuracy of slow/fast classification 37
3.8 Censoring scenario analysis 38

4 Evaluation **41**
4.1 Data description 41
4.2 Non-parametric analysis 45
4.3 Semi-parametric analysis 46
4.3.1 Fixed-effect models 47
4.3.2 Mixed-effects/frailty models 48
4.4 Parametric analysis 48
4.4.1 Fixed-effect models 51
4.4.2 Mixed-effects/frailty models 54
4.5 Validation 57
4.6 Considering censoring 59
4.7 Discussion 61

4.7.1 Validation of Hypothesis 63
4.7.2 Stakeholder feedback 63
4.8 Threats to validity 65

**5 Conclusions,
Summary and Future Work 67**
5.1 Future work 68

Appendix A Results Tables 71

Appendix B Analysis scripts 87
B.1 Main analysis script 87
B.2 Data preprocess 93
B.3 Survival analysis helper script 95
B.4 Survival analysis helper script 106
B.5 External validation 111
B.6 Linear model metrics 119

References 127

Index 133

List of Tables

4.1 Number of observations per release. 42
4.2 Descriptive analysis table. 44
4.3 Regression estimates of Cox proportional hazard models. 49
4.4 Parametric regression estimates of fixed-effect models. 52
4.5 Performance metrics for fixed-effect models. 54
4.6 Parametric regression estimates of mixed-effect models. 56
4.7 Performance metrics for mixed-effects models. 57
4.8 Performance metrics for fixed-effect models, including *developer*. 57
4.9 Performance metrics for fixed-effects models for external validation. . . . 58
4.10 Performance metrics for mixed-effects models for external validation. . . 58

A.1 *Akaike information criterion. Fixed-effect models.* 73
A.2 *Standard deviation of the residuals for internal validation predictions. Fixed-effect models.* 74
A.3 *Kendall rank correlation coefficient for internal validation predictions. Fixed-effect models.* 75
A.4 *Standard deviation of the residuals for external validation predictions. Fixed-effect models.* 76
A.5 *Kendall rank correlation coefficient for external validation predictions. Fixed-effect models.* 77
A.6 *Akaike information criterion. Mixed-effects / frailty models.* 78
A.7 *Standard deviation of the residuals for internal validation predictions. Mixed-effects / frailty models.* 79
A.8 *Kendall rank correlation coefficient for internal validation predictions. Mixed-effects / frailty models.* 80

A.9 *Standard deviation of the residuals for external validation predictions. Mixed-effects / frailty models.* . . . 81
A.10 *Kendall rank correlation coefficient for external validation predictions. Fixed-effect models.* . . . 82
A.11 *Standard deviation of the residuals for internal validation predictions on filtered dataset. Fixed-effect models.* . . . 83
A.12 *Kendall rank correlation coefficient for internal validation predictions on filtered dataset. Fixed-effect models.* . . . 84
A.13 *Accuracy of fast/slow classification of the models.* . . . 85

List of Figures

3.1 Different types of censored observations. 23
3.2 Right censored data. Representation of the dataset under study. 24
3.3 Representation of skewed distributions. 25
3.4 True effect of fixed variable on the decision (from [6]). 33
3.5 True effect of random variable on the decision (from [7]). 34
3.6 Representation of the truncation process. 39

4.1 Kaplan-Meier and cumulative hazard curves on the empirical data of the first three releases. 45
4.2 Log transformation on the x-axis of Figure 4.1 for clarity. 46
4.3 Survival curves of each distribution compared to the empirical data. . . . 50
4.4 Cumulative hazard curves of each distribution compared to the empirical data. 51
4.5 Artificial truncation 59
4.6 Accuracy fluctuation over different fast/slow thresholds. 64

Chapter 1

Introduction

1.1 Motivation and problem statement

Computer software development consists of multiple stages, from requirements engineering, to implementation, initial release, and up until the end of further software support [36]. A significant amount of resources during software development is spent on bug fixing and feature implementation. The expected time necessary to fix a single bug or implement a specific feature — commonly referred as *time-to-fix* or *time-to-deliver*, respectively — is short. However, the large volume of bugs and features involved in software development create logistical problems to stakeholders involved in software development. Misestimating the amount of time needed for resolving reported issues might result in either delay of software launch and/or the launch of a software with degraded quality. Accurate estimation of time-to-deliver can result in efficient resource allocation, as problems expected to be lengthier to solve will have more resources allocated to them. Such improvements can result in significant cost savings, through reduced resource utilization, earlier launching, improved product quality and customer satisfaction.

A number of attempts to estimate the expected resolution time can be found in the computer science literature [4, 39, 30, 20, 54, 51, 3, 43, 1, 19]. However, the predicting accuracy of these studies is not adequate. The methodological rigor of these methods varies. Researchers studied the expected time-to-fix using methods as simple as descriptive analysis [33, 53], to sophisticated prediction models that incorporate regression modeling, neural networks, machine learning and survival analysis [39, 20, 54, 3, 43, 19].

An extensive review of the literature on the topic can be found in Chapter 2.

1.2 Objective

The primary objective of this study is to generate models that can predict with improved accuracy the development time necessary for a newly reported bug or feature to be resolved. Secondary objectives are (i) understanding which attributes affect development time the most, so that the processes can be altered (to reduce time-to-deliver) and (ii) predicting which reported bug or feature will be resolved below a time threshold (fast) and which above the threshold (slow). In order to reach these objectives, we answer the following research questions:

RQ1: How incorporation of random-effects and/or censored data influence performance of a model that predicts time-to-deliver?

RQ2: How geographical, churn, and complexity factors affect the duration of the time-to-deliver?

Time-to-deliver is the primary outcome of this study and is defined as the difference between the time-stamp that an issue is reported as delivered and the time-stamp that the report was submitted and assigned to a developer. From here on, we use the term time-to-deliver for both bug fixes and feature implementations.

1.3 Proposed Solution

Time-to-deliver data belong to the family of *time-to-event* data. Time-to-event or survival data, as they are also known [22, 35], can be found when the outcome of interest is the time elapsed from a starting point until an event of interest takes place. Consequently, the analysis and interpretation of this type of data is referred to as time-to-event or survival analysis. This type of analysis is mainly used in biostatistics and epidemiology and relates to the time between disease onset and death (or another specific event). Such analysis is also used in engineering, where it is referred to as reliability analysis, and studies the time until failure – mainly for mission critical equipment and material [17].

One of the distinctive characteristics of survival data is that observations of the outcome of interest (duration outcome) may be incomplete. In the case of the time-to-deliver variable, it is a reasonable assumption that prediction models can be applied at a certain time point where some of the issue reports are still open – i.e., ongoing. The development process might have started for them, however they have not been resolved yet. Such observations are termed as *censored* and essentially indicate incompleteness of the data. We are going to incorporate this incomplete information into our models, in order to enhance our sample size and contribute in the process of predictive modeling. More details regarding censoring and its formation is provided in Chapter 3.

Skewness of the variable of interest is also highly prevalent in time-to-event data. A symmetric (and, preferably, bell-shaped) distribution is what ideally a researcher would expect to see from the data under study. However, extremely large time periods, in combination with the fact that time-to-event are bound at zero, will likely result to an asymmetry. This is defined as the skewness of the distribution and may vary according to its direction (left/right skewness) and its degree (e.g., moderate, extreme). We will deal with it by using and experimenting with different parametric distribution models. More examples and discussion of how to address such issues is given in Chapter 3.

Finally, often in a sample, multiple time-to-event observations might belong to the same subject/individual. This results in some correlation between the observations. In the time-to-deliver literature, the use of explanatory variables have been tried to be accommodated to resolve this correlation, such as the experience or reputation of the developer [21, 4]. In addition to the fact that there is a difference of characteristics that describe every developer, we can try to estimate a model where we acknowledge that there are remaining differences between subjects (e.g., developers) which are unobservable. In practice, this is accomplished by incorporating *random-effects* parameters in survival models, which after they are combined with fixed-effect parameters, result in a mixed-effects models (or *frailty* models in the survival context). We are dealing with this issue by introducing the developer as the random-effects term into our models. Mixed-effects and frailty models are discussed in Chapter 3.

1.4 Novelty

To the best of our knowledge, this is the first study utilizing parametric survival analysis and mixed-effects modelling on time-to-deliver data in software engineering. Although other researchers have relied on survival analysis tools in the past, we are presenting a comprehensive and replicable methodological approach. We try to cover most scenarios of data structures and limitations that can be faced with this type of data. Namely (i) censoring of the time-to-deliver variable, (ii) skewness of the distribution of time-to-deliver and, (iii) hierarchical structure of the time-to-deliver variable.

i Censoring of the response variable is a factor that can frequently exist in time-to-event data, especially when dealing with real-world data. The amount of information that these incomplete observations include can potentially improve the predicting power of the model and should not be disregarded.

ii The skewness of the distribution of the response variable is also a considerable factor that might affect the final outcome. Transformations are a potential solution to the skewness problem which we will explore along with alternative options, such as assuming different distributional shapes for our data.

iii Mixed-effects models have not been extensively used in computer science. We are investigating in this study whether adoption of such methods significantly improves the models' predictive accuracy

More details and definitions of the terms that are discussed in the last two Sections are provided in Chapter 3.

1.5 Contribution

The contributions of this work can be summarized as:

- A way of successfully identifying and understanding the relationship between the different explanatory variables and the response variable, in the domain of defect prediction and quality assurance.

- A novel approach, of building a predicting regression model for time-to-deliver data while leveraging incomplete information and random-effects.
- A prototype tool implementing this novel approach, listed in Appendix B, that can be used by practitioners and academics.

1.6 Outline

In Chapter 2 similar work and other relevant studies are discussed. Chapter 3 introduces the concepts of the statistical methods used, describes the data and provides the methodological approach followed in the analysis. Chapter 4 presents the results of the analysis. Finally, in Chapter 5, a summary of this study is provided, along with a conclusion and a scope for intended future work.

Chapter 2

Literature Review

This chapter reviews relevant published studies, where researchers were tackling the same problem: predicting the time necessary to deliver an issue report. We reviewed studies that focused on the general aspect of quality assurance and the number of bugs remaining in a system, as well as studies with an objective to predict the time-to-deliver of a particular issue report. We focused our discussion on the studies' findings as well as the methodological approaches followed.

A number of different methodologies have been explored, and in many cases were compared to each other in the past. From machine learning algorithms, including Naïve Bayes, Bayesian Networks, and Decision Trees to neural networks and support vector machines, as well as more traditional methodologies and statistical tools for data analysis.

2.1 Software quality assurance

Traditionally, in software analysis and software testing, researchers focused on the number of remaining defects before launch [15, 37, 27]. Such prediction has been shown to improve quality in a variety of aspects, but at the same time it has also been identified as a fundamental challenge in software engineering; better development management, cost efficiency, improved resource allocation, better development coordination and, in general, final software quality improvement, are only some of the acknowledged advantages.

Two of the most cited studies in this area come from Fenton and Neil [15], and Lessmann et al. [37]. Both studies are comparing different defect prediction models, illustrating the importance of such an in advance knowledge.

Fenton and Neil [15] compared and criticized existing studies and the models amongst them. They pinpoint that novel and methodologically concrete approaches are essential for defect prediction, while empirical studies are of lesser importance in defect prediction domain. However, based on the mistakes that have been made in this domain, they try to lead future researchers into successfully deciding the appropriate model specifications and the data to work with, in the inevitably difficult field of defect prediction.

Lessmann et al. [37] tried to benchmark different classification models and then propose the best ones suitable for defect prediction. They compared a total of 22 classifiers, based primarily on the area under the receiver operating characteristics curve (AUC) and utilized static code metrics over 10 different open sourced datasets. Since their experimental results did not yield significant differences among the top performing models, additional characteristics, such as computational efficiency, ease of use, and comprehensibility were also included in model selection.

However, the information generated just by predicting the number of expected defects is not sufficient since there is always a need for further improvements in the software development domain. That is the reason why, more recently, researchers have extended their studies in predicting the time necessary for a defect to get fixed as well as the time needed for new functionality to be developed and incorporated.

Machine learning, more sophisticated artificial intelligence approaches, and statistical analysis techniques have been extensively used and compared for time-to-deliver prediction [4, 9, 20, 54]. The rationale behind any method used, is to be able to transfer previous knowledge coming from the already observed data into future similar occurrences. The rationale is also based on the assumption that the past collected data, from previous development cycles or even from different software systems, can assist in the prediction process, no matter the approach. Consequently, we rely on the assumption that the main contributing factors and conditions are reasonably homogeneous and robust.

Although the approaches might differ, the attributes utilized to achieve this goal across studies are, to a great extend, similar. The information provider in the majority of previous studies is the *bug tracking system* [47], also commonly referred as *issue tracking system*, which is an essential component in a well organized software development infrastructure. The records of a tracking system are organized in a database and contain information, such as, the type of the issue, title, description, the time reported, the user submitting the report, the developer that it was assigned to, various metrics related to the changes that took place, the priority and severity of the issue. This type of software is usually integrated with other project management tools, being able to provide an even richer set of attributes.

In a recent study, Canforna et al. [9] used a survival analysis technique known as the Cox proportional hazard model (see Chapter 3). However, instead of estimating resolution times, they tried to predict the hazard of the survival of a bug, from injection until resolution. The time interval that they studied, includes the time when the bug is reported, but does not necessarily consider it as the starting point of measure. This work has been a valuable step in modeling expected resolution-time within a survival analysis framework. However, the study failed to account for a number of methodological challenges in the presence of properties in the data structure. Examples of such violations include the presence of censoring, the extreme skewness of the duration data, the limitations of proportional hazard models in generating time predictions, and the presence of within group dependence (e.g. multiple bugs fixed by the same developer). Although the proportional hazard models do account for the limitations mentioned above, they were not incorporated or mentioned in this study. These violations can have important consequences on the estimation of expected resolution-time and, therefore, their impact needs to be considered.

The impact of design and code reviews on software was studied by Kemerer et al. [27]. Although the scope of their work is different, they utilized regression models and introduced mixed models having the developer as the random-effects attribute; acknowledging the superiority and the advantages of such use.

Finally, Schalken et al. [45] also proved the superiority of the mixed-effects models, while they investigated the success of a software process improvement program, in a large financial institution. Using hierarchical linear models they noticed a great enhancement in the sensitivity of analysis of the empirical data they utilized.

2.2 Number of bugs remaining

The majority of studies in this area focused on the number of bugs remaining or are yet to be discovered, within a future time span [15, 37]. These studies have extensively utilized tree-based classification methods, Bayesian belief networks, neural networks, analogy-based approaches, and statistical procedures. Sometimes though, researchers mention the importance of being able to predict the time-to-deliver as supplementary to the number of bugs that are expected in the future [43]. We will be focusing on the estimation of the former. Furthermore in our study, in addition to the bug reports, we consider the functionality reported issues as well.

In 2010, D'Ambros et al. [14] tried to provide a benchmarking pattern for comparison of defect prediction approaches in terms of accuracy, complexity, and the type of data these require to make predictions. The need for an established comparison and benchmarking methodology across different methods, illustrates the plethora of studies in the field of defect prediction and the necessity of prediction reliability.

2.3 Time-to-deliver estimation

In an attempt to provide more detailed predictions around bug resolution, researchers focused on the prediction of time-to-deliver rather than predicting the number of remaining bugs. Different approaches, others dealing with the time-to-deliver itself, or in an attempt to simplify the problem just classify a new report as a slow or quick fix. Nevertheless, all of them dealing with the duration of the resolution of a single reported issue [1, 4, 39, 30, 20, 54, 51, 3, 43, 19].

In particular, Bhattacharya and Neamtiu [4], based on prior studies and the methodologies that were followed, tried to illustrate that the models constructed in these studies cannot be easily generalized and adapted into other (external) systems. They used multivariate and univariate regression testing, to assess the predicting power of previously built models, while predicting defects on external datasets. The datasets that were used belong to open source projects that are commonly used for defect prediction. The results of this study show that the predictive power of the models (that they assessed) ranges from 30% to 49% and identified the necessity of finding more appropriate explanatory variables, other than bug severity, bug dependencies, number of developers involved and patches applied for a fix.

The significance of Bhattacharya's and Neamtiu's [4] findings, that different software have significant differences in terms of resolution times, can be observed in two other studies; the work of Anbalagan with Vouk [3] and Panjer [43]. The best explanatory variables, in terms of predicting the time-to-deliver, differed in each study; Anbalagan and Vouk identified the number of involved developers as the most relevant one, while for Panjer, a combination of the commenting activity in the bug report along with severity, component, and version[1] are the most important ones. According to these findings, we take under consideration the necessity of more relevant and correlated to the time-to-deliver explanatory variables and study their correlation, as well as their performance on the models built. Additionally, we consider within group dependence on some of the explanatory variables and decide to act accordingly.

Kim and Whitehead [30] use the time needed to fix a bug as a criterion for the software's quality. They identified and related the quality of the component that a fix was necessary, the file that the component belongs to, up to the quality of the software in general. They used this information solely to measure the quality of the software components, and did not try to estimate the bug-fix time. The more time is needed for a bug to get resolved, the worst the quality of the component. They concluded that the time-to-deliver is an important measure for quality assurance, after analyzing bug fix statistics for two projects.

In a similar study, Koru et al. [33] consider the characteristics of the components and the relationship that they might have with bug existence. They found a strong correlation between component size and bug proneness. Interestingly, they focused on the size of the classes themselves instead of the size of the file, which is similar to the classification of components and functions that we have in our dataset. They utilized a Cox proportional hazard model to estimate the effect of the size of a component on defect proneness. Although we follow the same methodological approach, by utilizing Cox proportional hazard models, we focus on the time-to-deliver instead.

Marks et al. [39] identified that the most correlated attributes with the time needed to fix a bug, are different between two separate open source software (Eclipse and Mozilla). Additionally, even within the same software, time progression can change the most correlated attributes. The most relative ones in this study were identified by performing sensitivity analysis on the attributes of each software. Components related to the bug, developer, and reporter, as well as the description of the bug are the most important "di-

[1] Variables are listed in the order of their significance.

mensions" affecting resolution time. They also split the fix time into classes, representing fast, medium, and slow fixes. The random tree classifier that they used could correctly classify 65% of the bugs. We are also evaluating the effectiveness of each attribute and consider the differences that inevitably exist, not only between different software, but at the same time within the same software. Although in this study we will not be evaluating the effectiveness of a model in an external dataset, we acknowledge this limitation and plan to tackle this in the future.

The importance of time progression is also discussed in the study of Habayeb [19]. The author studied the effect of temporal characteristics by building a Hidden Markov model that, compared to previous approaches, performs better in predicting the time-to-fix on a Firefox dataset. We take Habayeb's study as one of the most relevant in terms of estimating the time-to-deliver and try to show some considerable differences in the time-to-event family of the data that this and other studies have been dealing with. However, we are using different dataset, attributes, and tooling.

In an empirical study, Hewett and Kijsanayothin [20] used various machine learning and computational intelligence techniques in order to achieve optimal results in predicting time-to-fix. In the same way as Marks et al. [39], they tried to understand the reason and the source of the reported issue and categorize them as low, medium, or high duration. For attribute selection and evaluation, they used a *wrapper method* based technique, while we will be statistically analyzing the correlation of each attribute to the time-to-fix. Interestingly, they discuss the importance of the pre- or post-release information, that we are also considering in our hypotheses, as discussed in Section 3.1.

Another empirical study, on commercial software this time, was conducted by Zhang et al. [54]. Their goal was to classify a newly reported bug in two classes: slow or quick, based on a preset threshold. They rely on a Markov model-based method to predict the number of bugs that will be fixed in the future, and they also introduce a time prediction approach. Afterwards, they propose a method based on k-Nearest Neighbour and compare its efficiency to other commonly used machine learning techniques (Bayesian Networks, Naïve Bayes, Radial Basis Function Network, and Decision trees). Using three different attribute evaluation techniques for categorical data (Chi-square, Gain ratio, and Information gain) they conclude that bug submitter/originator and developer have the most predicting relevance.

An other interesting approach, but not so related in terms of methodology, was followed by Weiß et al. [51] They were able to achieve good prediction estimates of fixing effort by utilizing text similarity techniques; fixing effort is defined as the actual person-hours that will be needed to deliver the fix. They used *Lucene*, an Apache text similarity measuring engine, to identify similar past issue reports and be able to categorize a new one.

In addition to the generalization problem that was discussed above, researchers have also identified factors impacting bug fixing time that are hard to determine or extract from the issue tracking systems. One of the most common issues in this category is the time that intervenes between the report of the issue and the actual time that a developer starts working on it [53]. Also, duplicate reports is another considerable cost inefficiency [24].

Finally, in the Software Engineering domain, regression analysis has been used to estimate and show dependability of various attributes with the quality of software [29], bug prediction [14, 5], defect density [41], as well as time-to-fix duration [4, 9, 20, 30].

Chapter 3

Methodology and Implementation

Based on the literature discussed in the previous chapter, we provide the methodology followed to achieve our goal. We first state the hypotheses in Section 3.1. Subsequently, we discuss the general principles underlying regression modelling in Section 3.2. Next we continue with regression estimation approaches and the limitations associated with using regression analysis to test our hypotheses in Sections 3.3 and 3.4 respectively. Finally, we introduce the concept of survival analysis as the methodological framework we followed to address the limitations of regression analysis when using time-to-event data, in Sections 3.5, 3.6 and 3.7.

3.1 Theoretical framework

In order to answer the main research question of this thesis (the prediction of time-to-deliver) we first need to identify the parameters that will constitute a bug report or a feature request as more likely to be resolved. Can the management take corrective actions towards a faster, more efficient, and more qualitative resolution? For that purpose we formed hypotheses on the relation of a number of explanatory variables with the time-to-deliver a bug or feature, based on the literature and input from the provider of the data. On that basis we hypothesized the following:

The binary attribute Pre/Post release shows if the issue report was completed before or after the current release that our dataset comes from. If this information is available from the time that the bug is reported, we would expect that bugs reported as *post release* would need more time to be delivered, mainly because there are no time constraints. Respectively, *pre release* indicated reports, have a specific due date that need to be resolved. Because of this time constraint, we would expect them to get resolved faster.

H1 *Issue reports that are delivered as pre-release, are expected to get resolved faster than post-release.*

As shown in [4, 20] the size and the number of components involved in a report can affect the time-to-deliver. Therefore, we would expect bug reports that involve more complex or larger components to require more time to be resolved. Consequently, a bigger number of components involved in a single issue report is also more likely to require longer time to resolve.

H2 *The number of components involved in the development if an issue report, is expected to be positively correlated to the time-to-deliver. The more the components – the longer the time necessary.*

Severity and priority of the issue report is a required field in most of the bug tracking systems. This information may also connect to the first element (*H1*) of this list. If the manager/developer know that this task has low severity/priority and therefore can wait until the next release, might lead to a longer duration of resolution.

H3 *Higher severity and/or priority of the issue report is expected to result in a faster resolution.*

As it was described before, we are investigating all the possible reported issues from the tracking system. Differences might occur and be expected for different types of reports. Features in general are not always highly ranked in the priority list, in contrast to bugs, that need to get resolved most of the times before the next release.

H4 *We would generally expect bugs to be resolved faster than features.*

A small predefined tag/description is given to each issue report. This can be considered as a subcategory of the bug/feature category.

H5 *We would expect some issue reports, given their symptom tag, to be more complicated – hence take more time to get resolved.*

Experience, workload, development type, are only some of the characteristics that can differentiate between developers. As in previous studies [4, 18], we expect that developers are also a factor affecting the time-to-deliver. We would also expect that within group dependence of the developers, is also a significant factor on the duration of the time-to-deliver, therefore we will be incorporating this assumption into our models.

H6 *The developer assigned to an issue report is expected to be a significant factor affecting the time-to-deliver.*

Since the development team of the product we are studying is distributed around the world, the country of origin might also affect the time-to-deliver. Differences in the culture, characteristics – as defined in *H6*, as well as differences in the size of the team in each country, can be contributing in the resolution time variations.

H7 *Issue reports developed in different countries are expected to have variations in the time-to-deliver.*

As most of the previous studies did, we will use multivariate models towards our goal. However, univariate models will also be built initially, in order to form a baseline model estimation.

3.2 Regression analysis

Regression analysis is a statistical process for estimating the relationship between two (or more) variables [32, 40]. Regression analysis allows us to understand the effect of the explanatory variable(s) on the response variable. Depending on the functional relationship assumed between the response and the explanatory variables, models can be described as *linear*, *generalized linear* and *nonlinear*.

In regression analysis, variables are categorized into two sets; the response variable and the explanatory variable(s). Response or dependent variable is the variable of interest that is being measured in the experiment. Explanatory or independent are those variables that are assumed to be associated with, or can be used to inform predictions of, the response variable. Regression modelling estimates the level of association between the response and the explanatory variable(s).

This relationship can be used for predictions. However, as many researchers showed in the past [3, 43, 39, 20], being able to draw inference and understand a relationship that describes a situation might be worth more than using this relationship for prediction purposes.

3.2.1 Linear Regression

Linear regression is the simplest form of regression analysis, where the response and explanatory variables are assumed to be linearly associated. In linear regression, the objective is to find the (straight) line that has the smallest distance from every data point of the response variable [40]. Linear regression is also defined as the study of the linear and additive relationships between variables. Assuming n is the number of observations in a sample, y is a vector of size n capturing the response variable and x is a vector of size n capturing the explanatory variable. Subsequently the linear regression model can be written as:

$$y = \beta_0 + \beta_1 x + \epsilon, \tag{3.1}$$

where, β_0 is the intercept and captures the value of y when x is equal to zero, β_1 is the slope and shows the change on y for a unit change on x, and ϵ is the residual or error term which captures the distance of the observed from the estimated value of y. β_0 and β_1 are also referred as the regression coefficients of the linear regression model.

In case of p explanatory variables, the simple linear regression can be extended to a multivariable regression model:

$$\begin{aligned} y &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \epsilon \\ &= \boldsymbol{x}\beta + \epsilon, \end{aligned} \tag{3.2}$$

where $\boldsymbol{x}$ is the explanatory variable matrix of size $n \times (p+1)$ and β is a vector of regression coefficients of size $p+1$.

Given estimates of β, denoted $\hat{\beta}$, one can generate predictions of the response variable

$\hat{y}$, given $\boldsymbol{x}$, using the equation:

$$E(\boldsymbol{y}|\hat{\beta}, \boldsymbol{x}) = \hat{\boldsymbol{y}} = \boldsymbol{x}\hat{\beta}. \tag{3.3}$$

Using 3.2 and 3.3, the error term ϵ can be defined as $\epsilon = y - \hat{y}$.

3.3 Regression estimation

There are several estimation approaches for regression models. Although the majority of regression approaches are solved using likelihood - based methods, simple problems like the linear regression can be solved with Ordinary Least Square (OLS) regression methods.

3.3.1 Ordinary Least Squares

There are multiple ways of obtaining the best linear unbiased estimates (BLUE) of β. The simplest estimation method is the Ordinary Least Squares method. This method estimates the regression coefficient β that minimizes the sum of the squared distances between the observed response variable y and the predicted variable $\hat{y}$. In other words, the OLS method minimizes the sum of the squared errors (SSE):

$$\hat{\beta} = \arg\min_{\beta} \left[\sum_{i=1}^{n} (y - \boldsymbol{x}\beta)^2 \right]$$

With OLS method, we are able to get an estimate of the regression parameter β. As an estimate, it is accompanied with statistical properties. One of these properties is the variance, which can tell us how precise is the estimate. The variance of the errors assists in the estimation of this variance.

The coefficients can be estimated using the function:

$$\hat{\beta} = (\boldsymbol{x}^T \boldsymbol{x})^{-1} (\boldsymbol{x}^T \boldsymbol{y}), \tag{3.4}$$

where ^{T} denotes the transpose of a vector/matrix; the variance is approximated by:

$$\hat{\sigma}^2 = \frac{\epsilon^T \epsilon}{n-p} = \frac{(y - \boldsymbol{x}\hat{\beta})^T (y - \boldsymbol{x}\hat{\beta})}{n-p}, \tag{3.5}$$

where $\hat{\sigma}^2$ is the variance of the sample and p is the number of parameters being estimated for the model.

Estimation of the linear regression parameters with OLS requires a number of assumptions hold:

(i) The residuals are independent, there is no statistical correlation between them.

(ii) The residuals follow a normal distribution.

(iii) The relationship between the response and the explanatory variables is linear. For every explanatory variable there is a function that explains the response variable as a straight line. The slope of each of these lines is not related to the others. The effect of each straight line function to the response variable is additive to each other. This is also referred as *additivity*.

(iv) Homoscedasticity of the residuals. Also known as homogeneity of variance of the errors.

Some of these assumptions can be relaxed (e.g normality) with some loss of inference (e.g., no p-values).

3.3.2 Maximum Likelihood

If we assume that the residuals are normally distributed, we could try to estimate the parameters that explain best the distribution of the residuals. The probability that a residual term comes from a normal distribution with mean 0 (since the residuals are centred around zero) and variance σ^2 is:

$$f(\epsilon|\beta, \sigma) = \frac{1}{\sigma_\epsilon \sqrt{2\pi}} e^{-(y - x\beta)^2 / 2\sigma^2}. \tag{3.6}$$

The probability that *all* residual terms come from the same distribution is:

$$MLE = f(\epsilon_1|\beta_1,\sigma)\cdot f(\epsilon_2|\beta_2,\sigma)\cdots f(\epsilon_n|\beta_n,\sigma) = \prod_{i=1}^{n} f(\epsilon_i|\beta_i,\sigma), \tag{3.7}$$

which is referred to as the likelihood function. The parameters β and σ that maximize 3.7 are called the maximum likelihood estimates. The maximum likelihood estimator differentiates and sets the function equal to zero, in order to find the maximum value. It can be shown [46] that the maximum likelihood estimates are:

$$\beta_{MLE} = (\boldsymbol{x}^T\boldsymbol{x})^{-1}\boldsymbol{x}^T y \tag{3.8}$$

and

$$\sigma^2_{MLE} = \frac{(y-\boldsymbol{x}\hat{\beta})^T(y-\boldsymbol{x}\hat{\beta})}{n}. \tag{3.9}$$

3.4 Understanding the limitations

Time-to-event variables have often characteristics that are inherently not in-line with the assumptions of the linear regression model. Examples of such characteristics are:

- The presence of censored, incomplete, or missing data;
- Clustering of observations in hierarchical way (i.e violating the assumption of independence of observations);
- Skewed, zero-constrained distribution.

We will be referring to each item of the list above later in this chapter and provide some insights and explanations on the approaches that we followed in order to overcome them.

3.4.1 Censoring

Censoring exists when some of the observations of the response variable are incomplete due to some cause. Due to this incompleteness, the time-to-event is not accurately known. Although these observations are not “complete”, they still include some useful information. There are statistical models that can incorporate this information. Although exclusion of the censored observations enhances simplicity, it might lead to [38]:

- biased results,
- loss of efficiency (smaller sample size),
- increased variance of the estimated values.

Types of censored data

Below, we use as an example a software to illustrate the different types of censoring. We acknowledge two major time points during the release's development: T_A as the start time of our observation period and T_B as the end of the observation period. Furthermore, t_a is the date of reporting and t_b is the date of delivery of the issue report.

- In a case that an issue is reported between these two time points $T_A \leq t_a \leq T_B$ and is also resolved within them $T_A \leq t_b \leq T_B$, the observation is *complete* (i.e., not censored).

- In a similar case, an issue that is reported between these two time points $T_A \leq t_a \leq T_B$, but, due to time constraints or low priority/severity of the issue, the report could not be resolved before T_B. In this case we have a *right censored* observation in our response variable.

- Respectively, a *left censored* observation exists in the case where the time of resolution is $T_A \leq t_b \leq T_B$, but we do not know the submission time of the report.

- Finally a combination of right and left censoring, results to an *interval censored* observation, where it is known that the report and/or resolution of the issue happened within an interval time period, however the actual time point is not exactly known.

Figure 3.1 outlines the different types of censoring, described above.

The most common censoring type is right censoring. Especially in the type of problem that we will be dealing with, it is unlikely to have an issue report that the submit date is unknown. On the other hand, it is very likely to have a number of reports still undergoing

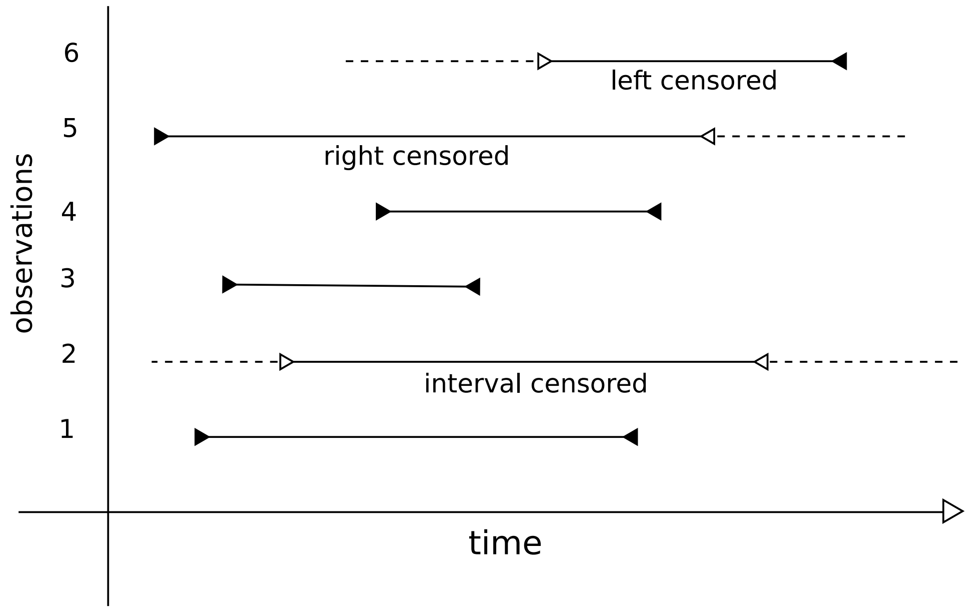

Figure 3.1: Different types of censored observations.

development and therefore having their resolution time as unknown. Figure 3.2 represents the nature of the data that we will be studying.

For this study, we created an artificial sample based on the original dataset, where a certain percentage of the data was assumed to be censored. We tested methods that account for censoring to understand the discrepancy between the "True" findings of the model when censoring is not present vs. the "real-world" scenario where a proportion of the data is censored.

3.4.2 Skewness

As we have seen above, one of the assumptions of the regression model is the normality of the residuals. Normally distributed variables are accompanied with the following properties:

- The normal curve (bell) is symmetrical around its mean μ,
- The mean divides the area into two equal parts,

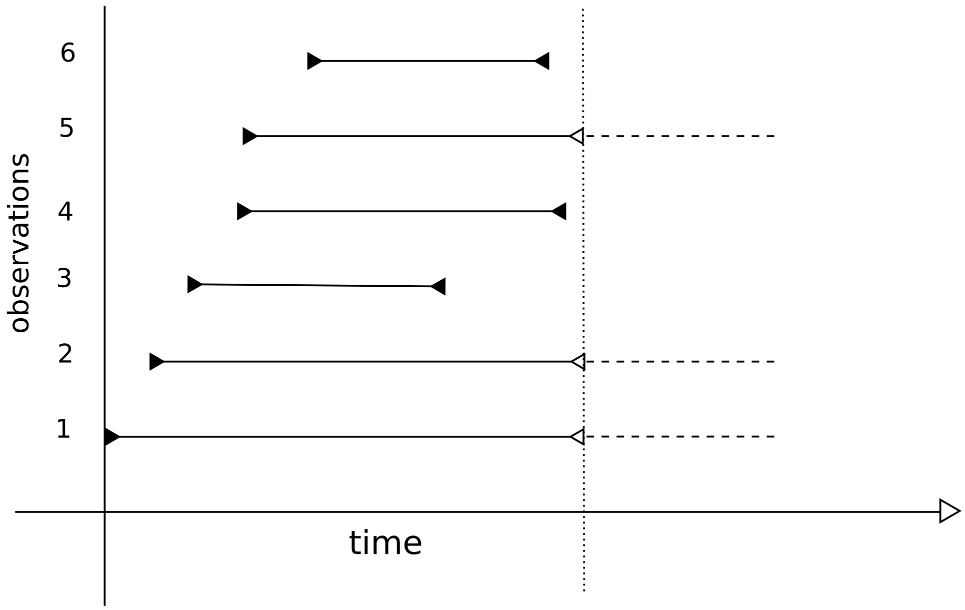

Figure 3.2: Right censored data. Representation of the dataset under study.

- The total area under the curve (integral) is equal to 1,
- It is completely defined by the mean and standard deviation σ.

In time-to-event data, this assumption is often violated: Despite that most events occur in short durations, there are often a few events occurring in disproportionately long durations. In addition, as time cannot be negative, time-to-event data are bound to zero. Distributions that do not follow the symmetry of a normal distribution are referred to as skewed or asymmetric distributions. A distribution can be described as:

- *Left* or *Negative skewed* - because its tail extends to the left or to the negative values of the x-axis;
- *Right* or *Positive skewed* - because its tail extends to the right or to the positive values of the x-axis;
- *Extreme tail* (positive or negative) - as by its name, this is an extreme condition of the previous categories, in a case where the tail stretches to the left or right of the horizontal axis;

An example of a left and right skewed distribution is given in Figure 3.3 compared to the normal distribution (or bell curve).

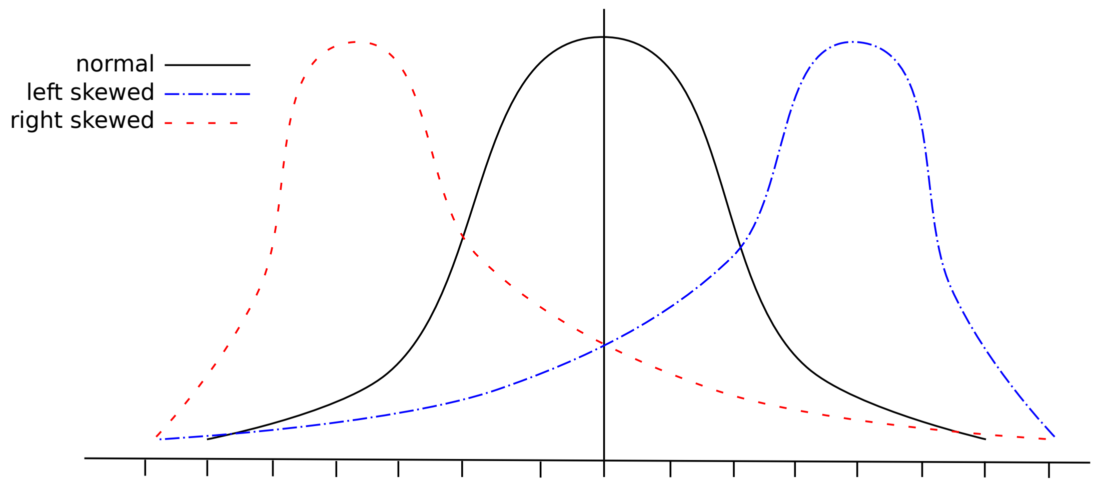

Figure 3.3: Representation of skewed distributions.

Transformations

As skewness results in violation of the assumptions of regression models, methods have been proposed that rely on transformations of the response variable in order to resemble more closely a normal distribution. These transformations can help with visually inspecting the data as well as applying to the (transformed) data more standard regression approaches. Despite their usefulness, transformations have some caveats too. Transformations imply that the studied relationships now, is that between the explanatory variables and the transformed response variable. In addition, transformations are not always easily invertible. Being able to revert back to the measuring scale that you started with is a very useful feature, most importantly because it gives the capability of easy comparison of the results.

3.5 Survival Analysis

Survival analysis (also known as duration, failure or reliability analysis in engineering) focuses on analyzing time-to-event data. Questions like: “How much time until an event (e.g., death) will occur?", "How much time will it take for a new bug to be reported?" or more appropriately for our case: "How much time will it take to resolve a reported issue?", as well as: "What is the proportion of bug reports that will be closed after a certain time threshold?", fall under this category.

Survival time or lifetime, is defined as the duration from a specific time point at which we start the observation for an event to occur until the time the event of interest occurs. In our case, the former is the time point that an issue is being reported, which can either be a bug or a functionality request, and the latter is the time point that this report is submitted as resolved.

Below we introduce concepts that are central to survival analysis; the survival function, Kaplan-Meier estimator of the survival function, the hazard function, the Cox proportional hazards models, and the accelerated failure time models.

3.5.1 Survival function

The survival function is a monotonic, non increasing function and is defined as the probability that a specific subject will not experience the event of interest until a specific time t. The survival function, for a given time t is defined as:

$$S(t) = P(T > t) = 1 - F(t), \tag{3.10}$$

where T is the time when the event occurs, and $F(t)$ is the cumulative distribution function (c.d.f.), i.e., $F(t) = P(T \leq t)$. The survival function is the complementary of the c.d.f. $F(t)$.

The properties of the survival function are as follows:

$$\begin{gathered} S(t) \in [0, 1], \\ S(0) = 1, \\ \lim_{t \to \infty} S(t) = 0, \\ S(t_1) \geq S(t_2) \Leftrightarrow t_1 \leq t_2. \end{gathered}$$

3.5.2 Hazard function

The instantaneous risk of an event at time t is called *hazard*. The hazard function or hazard rate is defined as $h(t)$, which is a conditional probability; i.e., it is conditional for the event to survive until time t. The formal definition is:

$$h(t) = \lim_{dt \to 0} \frac{Pr(t \leq T < t + dt | T \geq t)}{dt}. \tag{3.11}$$

In other words, we can denote the hazard function as the probability of an event to happen within a fraction of time – a small time interval $[t, t + dt)$.

Like the hazard function, the cumulative hazard function $H(t)$ is also not a probability. It represents the accumulation of hazard over time and is given by:

$$H(t) = \int_0^t h(t)dt. \tag{3.12}$$

The relation between the cumulative hazard function 3.12 and the survival function 3.10 is:

$$H(t) = -\ln S(t). \tag{3.13}$$

3.5.3 Non-parametric estimators

Non-parametric estimation is a statistical approach on fitting the empirical data without any theoretical constraints or assumptions. The Kaplan-Meier survival and the Nelson-Aalen cumulative hazard are both different techniques to graphically visualize the distribution of time-to-event data. Since the cumulative hazard and the survival functions are related, based on equation 3.13, the Kaplan-Meier and the Nelson-Aalen estimators can be used interchangeably. It has also been proven that they are asymptotically equivalent [16]. However, there are some differences and advantages that depend on the sample size that is being studied and other factors [11]. We will not be giving any further details regarding their differences, because we are only using these estimators for a graphical representation of the empirical data.

Kaplan-Meier estimator of the survival function

The most common way of estimating the survival function non-parametrically is the Kaplan-Meier survival estimator (product-limit estimator) [26]. It is a non-parametric or empirical method of estimating $S(t)$ for right-censored data (or non censored data).

A Kaplan-Meier estimator plot is a strictly non-increasing step curve, that can incorporate right censored observations. This plot is built by sorting all the records by their duration, from shortest to longest. Then, cumulatively sum the events and subtract them from the total number of subjects at risk of experiencing the event. Although the event information for censored observations is not available, they contribute to the at-risk population until they are censored. Graphically, they are illustrated as cross-points. As time progresses, the number of observations that remain survived keeps decreasing.

Nelson-Aalen cumulative hazard estimator

A non-parametric estimator of the cumulative hazard rate, which can also incorporate the presence of censored data, is the Nelson-Aalen cumulative hazard estimator. In contrast with the Kaplan-Meier survival estimate, the Nelson-Aalen estimate is a strictly non-decreasing, step curve. The curve starts from zero, since the hazard at time zero is equal to zero and accumulates to infinity as time progresses. This plot is essentially accumulating the hazard at every given time.

3.5.4 Semi-parametric estimation

Cox proportional hazard models

Cox proportional hazard models [13] is another family of (semi parametric) models for time-to-event data. More specifically, Cox proportional hazard models facilitate identifying a relationship between the hazard rate of an event and one or more explanatory variables. Cox models rely on the assumption that each explanatory variable x has a proportional effect β on some baseline hazard h_0. The model is referred to as a semi-parametric one since h_0 is estimated non-parametrically while the β's are estimated under the assumption that they follow some distribution (i.e., parametrically). Mathematically, the model can be expressed as:

$$h(t) = h(0) \cdot e^{\beta_1 x_1} \cdot e^{\beta_2 x_2} \cdots e^{\beta_n x_n}. \tag{3.14}$$

This relation is usually converted in a natural logarithm, to get advantage of its properties and transform the multiplicative equation 3.14 to an additive one:

$$\ln(h(t)) = \ln(h(0)) + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n. \tag{3.15}$$

The coefficient β in the Cox model concept is interpreted as the effect in the hazard rate, which comes in contrast with the concept of the coefficient in a survival model. Therefore a positive β implies higher risk, i.e., shorter survival time, while a negative β implies lower risk, i.e., longer survival.

The Cox model is fitted using a partial likelihood. Partial likelihoods are useful for the estimation of semi-parametric models. This likelihood function is maximized using the Newton-Raphson method [25].

3.5.5 Parametric estimation

Accelerated failure time models

Until now, non-parametric and semi-parametric estimating methods for time-to-event data have been discussed. As already mentioned above, the Kaplan-Meier estimate of the survival function and the Nelson-Aalen cumulative hazard is the best way of graphically representing empirical data (non-parametrically) with incorporation of censored observations. For the estimation of associations between explanatory variables and censored time-to-event observations we introduced the Cox model; a semi-parametric estimator that relies on the hazard function. As we discussed above, the limitation of the Cox models is that inference is drawn on the hazard rather than the time-to-event level.

An alternative method is the use of parametric survival models. As by their name, parametric models have all parameters of the models specified to be following a parametric distribution. This is however considered as one of their main disadvantages, the fact that a distribution has to be assumed for the values of the explanatory variables and consequently follow all of the distribution's properties. On the other hand, the main advantage is that the estimated value is no longer a hazard, but the time-to-deliver that we mainly want to estimate. An additional advantage is the ability to extrapolate in the presence of censored observations.

In parametric survival models, the residuals are assumed to be following a distribution that is more appropriate to the distribution of the data. While numerous distributions have been proposed for the use in time-to-event analysis [31], in this study we will be focusing on four of the most commonly used distributions: the exponential, the Weibull, the lognormal, and the loglogistic.

The regression models for a matrix of explanatory variables $\boldsymbol{x}$ under each of the distribution assumptions can be generally specified as:

$$\log(y) = \beta_{AFT}\boldsymbol{x} + \gamma W \tag{3.16}$$

where β_{AFT} is the vector of coefficients and γ is a scale parameter whose interpretation is dependent on the distribution assumed. Finally W is the vector of residuals following a distribution that is dependent on the assumed distribution of y. Below we provide the specific model assumptions and parameter interpretation for each distribution assumed. In all cases y is conditional on the explanatory variables $\boldsymbol{x}$.

- *Exponential*

  Let the time-to-deliver variable y follow an exponential distribution with a probability density function (p.d.f) that is equal to:

  $$p(y; \lambda) = \lambda e^{-\lambda y},$$

  where λ represents the rate parameter. Under that assumption, it follows that the residuals W, in 3.16, follow a one parameter extreme value distribution. As the scale parameter γ is fixed and constant over time at the value of 1 ($\gamma = 1$), β_{AFT} is the only vector of parameters to be estimated. $\beta_{AFT}\boldsymbol{x}$ captures the rate parameter, for the appropriate values of $\boldsymbol{x}$.

- *Weibull*

  If y follows a Weibull distribution:

  $$p(y; \lambda, k) = \frac{k}{\lambda}\left(\frac{y}{\lambda}\right)^{k-1} e^{-(y/\lambda)^k},$$

  with λ as the scale and k as the shape parameter, then the residuals W, in 3.16, follow a two parameter extreme value distribution. In this case, the parameter γ is the scale parameter ($\gamma = \lambda$) and the shape parameter is equal to $\beta_{AFT}\boldsymbol{x}$ ($\beta_{AFT}\boldsymbol{x} = k$).

- *Lognormal*

  When we assume a lognormal distribution for the time-to-deliver y, the p.d.f. is given by:

  $$p(y; \mu, \sigma) = \frac{1}{y\sigma\sqrt{2\pi}} e^{-\frac{(\ln y - \mu)^2}{2\sigma^2}},$$

  where μ represents the location and σ the scale parameters of the distribution. The regression model 3.16 has γ representing the scale parameter σ ($\gamma = \sigma$), $\beta_{AFT}\boldsymbol{x}$ the location parameter μ ($\beta_{AFT}\boldsymbol{x} = \mu$), and W is assumed to follow a standard normal distribution.

- *Loglogistic*

  Finally, if we assume that y follows a loglogistic distribution, the p.d.f. is:

  $$p(y; \alpha, \beta) = \frac{(\beta/\alpha)(y/\alpha)^{\beta-1}}{(1 + (y/\alpha)^{\beta})^2},$$

  where $\alpha > 0$ and $\beta > 0$ are the scale and shape parameters, respectively. In this case, in 3.16, γ is the inverse of the scale parameter (i.e., $\gamma = \alpha^{-1}$) and the intercept captures the product of the scale and the shape parameters along with the coefficients of the model (i.e., $\beta_{AFT}\boldsymbol{x} = \alpha\beta$). W is assumed to be following a logistic distribution.

All parametric models are estimated with the appropriate likelihood function.

3.6 Mixed-effects models

One of the assumptions of the regression models we have seen so far is that the observations in the study sample are independent of each other. For example, in our sample it is assumed that all issue reports are independent. However, reports that are dealt by the same developer are expected to be correlated as developer skills can vary. Hence the assumption of independence made in Chapter 3.3 is likely to be violated.

A conventional solution would be to introduce dummy covariates to capture the effect of each developer on the time-to-deliver. However, this would imply the estimation of a large number of additional covariates which would complicate and potentially bias our regression estimates. In addition, it is often the case that we are not interested in the effect of the developers on the time-to-deliver, but only want to adjust or quantify the variation across developers. Finally, if the model is to be used as a prediction tool, introducing covariates for the observed developers would make the model unsuitable for predicting the time-to-deliver of a newly hired developer (i.e., out of sample prediction).

A potential solution to the problem of independence assumption is the use of mixed-effects models [7]. Mixed-effects models utilize both parameters that their true effect, which is the final effect to the model, is assumed to be fixed across levels (the fixed-effect) as well as parameters that the effect is assumed to vary across levels with a given distributional pattern (the random-effects). The random-effects parameters can capture potential unobserved variation within levels (e.g., unobserved variation across developers) attributed to nesting or hierarchical data structures.

Before moving forward and trying to explain the usefulness and the idea behind the mixed-effects models we further elaborate on the distinction between fixed-effect and random-effects models.

3.6.1 Fixed-effect models

In a fixed-effect model, it is implicitly assumed that the true time-to-deliver for each developer only differs due to the variation on characteristics of the issue report (e.g., pre/post release date). Any excess observed variation across developers is attributed only to the sample variation. In other words, every developer, no matter the differences that inevitably exist between them, will affect the duration estimation the same.

In Figure 3.4 we can observe the effect of three different developers. The triangle on the x-axis represents the true average time-to-deliver in the model shown as μ and equal to $\mu = \beta\boldsymbol{x}$. The true effect for each developer is the circle on the x-axes and it is common across developers. As described before, the effect of every developer on the time-to-deliver is the same on the model. However, the observed values for each developer are different – shown by the squares. The assumption that a fixed-effect model makes is that, despite the true effect is the same, there might be variation across developers that is only attributed to the sample size. If we had enough (infinite) information, the observed effect (squares) would perfectly match their true effect (circles).

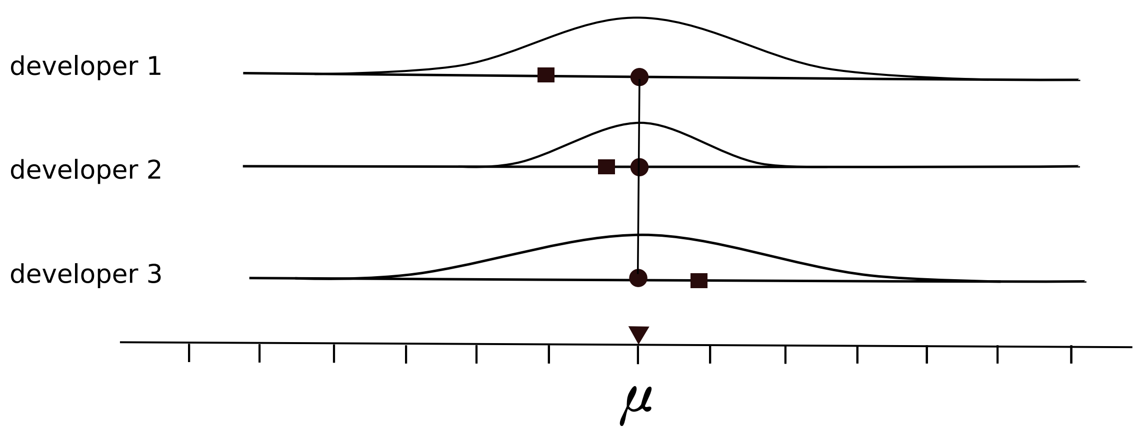

Figure 3.4: True effect of fixed variable on the decision (from [6]).

3.6.2 Random-effect models

The fixed-effect model, as discussed above, assumes that all developers have the same effect on the duration of the time-to-deliver that we are studying. However, differences in characteristics (such as experience, maturity, development skills, work load) might cause differences in the time that every individual needs to deliver a resolved reported issue (different effect). In such cases, we decide to use this information as random-effects. The assumption that we make in this case is that the final effect of the developers is a normal distribution, shown at the bottom of Figure 3.5. In comparison with the fixed-effect approach, although our sample is still limited, we expect the mean of the existing sample to match the mean of the case of an infinite sample size. However, what we observe on Figure 3.5 is the final normal distribution of the final effect of the developers to the time-to-deliver, as well as the observed values (squares) for every developer.

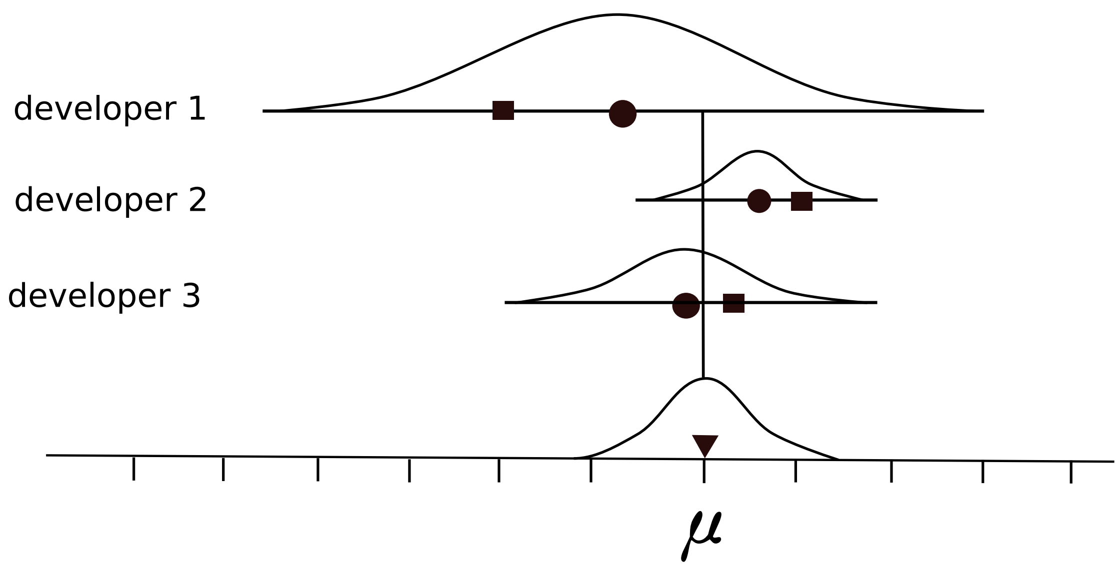

Figure 3.5: True effect of random variable on the decision (from [7]).

Frailty models

The way we are introducing the random-effects of a variable on our case, is through the concept of frailty [23]. Frailty models are an implementation of random-effects in survival analysis, for introduction of association and unobserved heterogeneity within the variable that is applied to. As described, we will be assuming a Gaussian distribution of the variable used in the frailty models.

3.7 Diagnostics and validation

After a model, or a series of models, have been fitted, it is essential to be able to assess them in various manners. Initially we want to make sure that the assumptions that were defined before fitting a model have not been violated. For example, in the simple case of a linear regression, the residuals should follow a normal distribution. This is easy to evaluate just by plotting the residuals and optically assessing the plot and its normality. However, for more complex cases, there are some additional tools that can help us compare different models.

Diagnostics and validation practices presented in this section are only applied to the parametric models. The main reason of this decision is also the major advantage of the parametric models; the fact that the estimates are on the response variable and not hazards.

3.7.1 Akaike Information Criterion

The Akaike Information Criterion (AIC) [2] is a measure used to test the goodness-of-fit across models that are applied on the same outcome of interest and on the same data sample. It is defined as:

$$AIC = 2k - 2\ln(L), \tag{3.17}$$

where k is the number of explanatory variables used in the model incremented by 1 (i.e., number of variables + the intercept), and L is the maximum value of the likelihood function. When comparing two models, the one with the smaller AIC value is the one that fits better.

As a basic notion while fitting a model, we might say that adding more explanatory variables will always make a model fit better, but will be trading against overfitting and overparameterizing our model. AIC calculates a trade-off between the number of parameters used and the incremental amount of variation explained by adding more parameters. The AIC of a model on its own does not provide a qualitative metric of quality of fit. AIC will only provide comparative information for a collection of models around the same variable of interest. For example, one can use AIC to select the best among parametric models, or among semi-parametric models. However, since the outcome is different for these two types of models (time-to-deliver for the fully-parametric and hazard for the semi-parametric), an AIC-based comparison between them is not valid.

AIC is also used to assess the predicting contribution of each explanatory variable through the stepwise algorithm. A stepwise algorithm, is an automated process of variable selection, based on the goodness-of-fit (AIC). The algorithm identifies and returns the variables that are contributing sufficiently to the improvement of the goodness-of-fit.

3.7.2 R-squared

Another statistic that measures the goodness-of-fit of a model is the R-squared (R^2 or r^2) [42]. R-squared measures the closeness of the data to the fitted regression line and is also known as the coefficient of determination. It is defined as the fraction of the response variable variation that can be explained by the fitted linear model and is given by:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}, \tag{3.18}$$

where $\bar{y}$ indicates the mean of the n real values y_i, and $\hat{y}_i$ are the predicted n values.

R-squared values range between 0 and 1. The higher the value of R^2, the more variability is explained, leading to better fit of the model to the data ($R^2 = 1$ suggests perfect fit). However, R^2 is not a perfect measure of goodness-of-fit: e.g., it cannot detect overfitting of the model, indicate whether the explanatory variables of a model have an effect on the response variable, or assess statistical significance of explanatory variables. It can also be misleading in case of non-linear models [34, 50]. Therefore, we resort to other goodness-of-fit measures described below.

3.7.3 Residual standard deviation

After fitting the data with regression analysis, a way to quantify the goodness-of-fit, is by calculating the standard deviation of the residuals [12]. The definition of the residuals is given by equation 3.1; in essence, it represents the vertical distance of the actual value from the fitted curve. The measurement of the standard deviation is used to evaluate the variation, or dispersion, of a sample. Low standard deviation indicates that the predicted values are closer to the mean (which is zero for the residuals) of the sample. On the other hand, high standard deviation indicate a scattered spread of the predicted values. The standard deviation of the residuals of a fitted model is defined as:

$$\sigma_\epsilon = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n-1}} = \sqrt{\frac{\sum_{i=1}^{n}(\epsilon^2)}{n-1}}. \tag{3.19}$$

The standard deviation of the residuals is also referred as the standard error of estimate.

3.7.4 Kendall rank correlation coefficient

Another statistical method that was utilized for measuring the performance of the models, is the Kendall rank correlation coefficient (RCC) [28], also referred as Kendall's τ (tau) coefficient. This statistical method is used to measure the ordinal association between two measured sets. As per its name, the statistical method is a measure of rank correlation, which is defined as the similarity on the orderings of the predicted against the true sets of data. The Kendall correlation is high when the two sets have a similar rank – equal to 1 for identical ranking, and low when the rank is no similar – equal to -1 for an inverse ranking. Finally, zero valued correlation coefficient denotes a random chance ranking.

In this study, we utilized Kendall rank correlation coefficient by assessing the ranking of the model's predicted values, against the real observations. The correlation test function takes as input these two numerical vectors and yields a numerical output $[-1, 1]$.

3.7.5 Accuracy of slow/fast classification

A common practice when dealing with time predictions and more specifically with time-to-deliver, is simplification of the outcome. As it has been described in Chapter 2, previous studies classified newly reported issue reports as fast or slow based on a pre-defined time threshold. Although we did not build classification models, but focused on predicting times, we assessed their predicting effectiveness by setting a threshold and marking the predicted times as fast or slow. In cases of balanced data, *accuracy* (ACC) is a reliable metric for evaluation of the performance of the model [49]. Accuracy provides a measurement that describes the closeness to the true value and is often referred as *trueness*. It is defined as the proportion of the true results, among the total number of observations under study, and is given by [49]:

$$ACC = \frac{TP + TN}{P + N}, \tag{3.20}$$

where TP is the number of the true positive occurrences – correctly classified as fast, TN is the number of true negative cases – correctly classified as slow, and $P + N$ represents the total number of samples referred to as positive and negative.

3.8 Censoring scenario analysis

One of our objectives was to examine appropriate methodology in the presence of censoring. To investigate the properties of survival analysis models when the data are censored, and to illustrate the methods that can be used in such circumstances, we designed scenario analyses where different proportions of our dataset were considered as censored. We utilized *truncation* – a common approach to artificially generate censored observations. With truncation, we specify preset censoring cut-off points, in order to illustrate conditions of censored information. We defined time points within our dataset that have different given proportions of censored information; from 0 to 20% $[0, 0.2]$ with a step of 0.01. For each censored dataset we fitted parametric models with all distributions assumed above and with explanatory covariates and applied diagnostic and validation procedures descibed in Section 3.7 (residual standard deviation, Kendall rank, and classification accuracy).

A drawback of the truncation methodology is that the available information is significantly reduced, in order to achieve the desired ratio of censored against non-censored records. This contradicts with the theoretical advantages that consideration of censored information comes with – enhanced sample size. However, we are just examining the influence, by reproducing the presence of censoring in our data.

In Figure 3.6 we visually represent the truncation process, which is based on Figure 3.2 presented in Section 3.4.1. In this figure, we included the observations that are being removed due to the cut-off point. As we will be noticing in the next chapter, non-normality of the data is the main reason of this excessive loss of information as we are truncating.

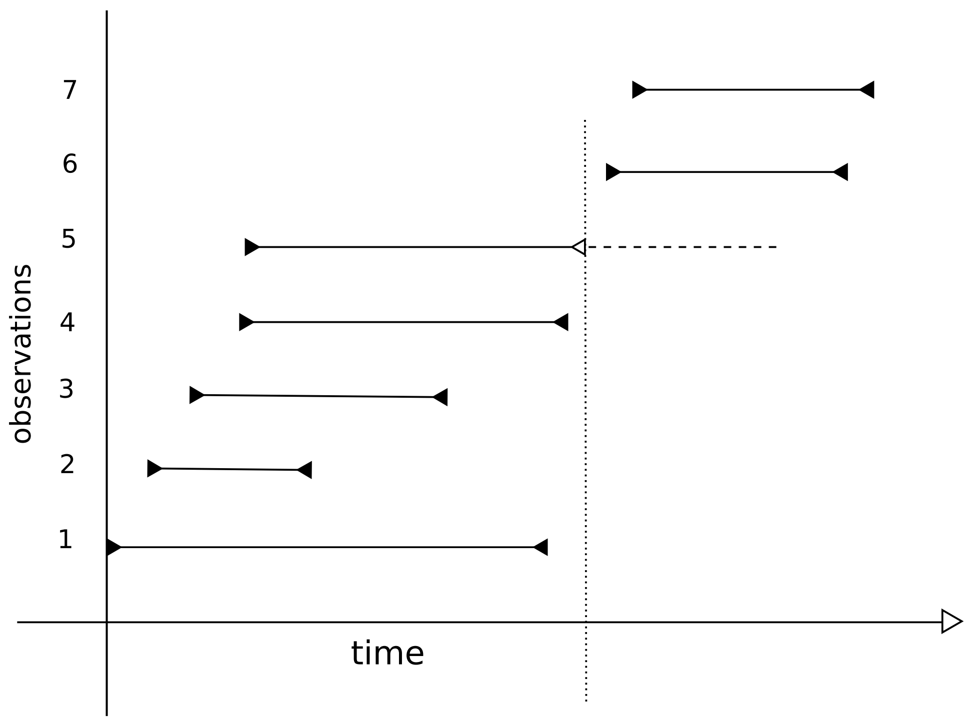

Figure 3.6: Representation of the truncation process.
In this case, in order to artificially reproduce 20% of censoring in our data, we have to remove two observations that their submit time is greater than the selected cut-off time point.

Chapter 4

Evaluation

In this chapter, the methodology provided above is applied on a set of real world, time-to-deliver issue report data. We will be presenting results from a generalizable approach that can be applied in similar cases. The structure of this chapter is as follows: in Section 4.1 we will be giving a description of the dataset that we leveraged for this study. Non-parametric analysis on the empirical data is presented in Section 4.2, semi-parametric in Section 4.3, and parametric in Section 4.4. In both Sections 4.3 and 4.4 we explore the influence of the random-effects term after we first evaluate the fixed-effect models. External validation of the models is presented in Section 4.5. Finally, in Section 4.6 influence of censoring is being discussed.

4.1 Data description

The dataset used for this study comes from a commercial software that is still actively being developed. Due to confidentiality issues and in order to preserve anonymity we are not allowed to reveal the name of the company or the software. This is also the reason why many values of the dataset were anonymized or presented in a scale different than this of the real values.

A total of 60,293 issue reports, delivered in a period of approximately 12 years, with all their attributes, are being analyzed. These observations were recorded through an issue tracking system. Issue or bug tracking systems [47], have been the main source of data for similar studies [4, 18]. The amount of information available comes from the successful development of four sequential releases of the same software, starting from 2003 until early 2016. As previously mentioned in Chapter 1, we consider as an "event" both bug reports and feature requests (collectively referred to as *issues*).

We chose to split our dataset into two parts, in order to achieve a test/train split that could represent real-world behaviour. Therefore, the first three releases were used as the train set, while the last release played the role of the test set. Table 4.1 shows the number of reported issues per release; in total we analyzed 46,296 observations for training and 13,997 observations for testing purposes.

Table 4.1: Number of observations per release.

Release ID	No. of observations
r	13,681
r + 1	16,956
r + 2	15,659
r + 3	13,997

'*r*' represents the first release that we have available, '*r + 1*' the one that followed, etc. '*r*', '*r + 1*', and '*r + 2*' are used as the train set, while '*r + 3*' is used as the test set.

The attributes that were used for model creation are described below.

- *Time-to-deliver*: This is our response variable. Although it is not directly provided from the issue tracking system, it is simple to calculate, by subtracting the issue submit time-stamp from the resolution/deliver time-stamp. We converted these dates to UNIX times for easier data manipulation. Due to data disclosure restrictions, only relative time-units are provided.

- *Defect*: This binary variable specifies if a given issue report is related to a defect or not. Essentially, a non-defect report infers a feature implementation.

- *Pre/Post release*: Submission, development, and completion of an issue report within the same release time window results in *pre* release. Inability for the issue report to be completed within this time-window gets the issue report to be marked as *post* release. We can argue that some issue reports, due to their *priority/severity* can be categorized as mandatory for completion as *pre* release in advance. Therefore we assume that this flag is a proxy for the *priority/severity* of the report.

- *Components involved*: This numerical attribute specifies the number of components that had to be modified for a given report to get resolved. Although one would argue that this is also information not available a priori, the originator of the report has to specify the main component against which the issue is reported. Based on that information and from previous knowledge, we can get a rough estimate of the components that will be involved.

- *Functions involved*: Similar to the components, another numeric attribute that is being used in this study, is the number of functions that were involved to the resolution of a single report. Again, although the number of the functions modified during development of the issue is not known in advance, we can say that this number is correlated with the main function that the issue report is connected.

- *Developer country*: The software that we are studying was developed by developers around the world. Since we are dealing with a commercial software that belongs to a well established company, development departments are spread all around the world. Although we notice a considerable amount of work contributed from a single country for all of the releases under study (>50%), we are studying the influence that country differences might have on the time-to-deliver. This is also one of the anonymized fields of the dataset. For the four releases we have information for, 14 unique countries are involved in the development process.

- *Symptom tag*: The person reporting an issue on the tracking system has to select a single symptom tag that briefly describes the issue. The selection has to be made among pre-defined unique tags, that briefly describe the problem or the feature that the assigned developer will be dealing with. A total of 27 tags were used for the total of the issue reports that we have available.

- *Developer*: This variable captures the developer who was assigned and resolved the reported issue. The software has 1,236 unique developers involved at the time that we are studying.

A descriptive analysis of the attributes in the dataset is presented in Table 4.2. Train set is represented by the first three releases and the test set by the last release.

Table 4.2: Descriptive analysis table.

Attributes	**Train set**	**Test set**
time-to-deliver[1]		
mean	865	551
range	0.01 - 46798	0.03 - 50000
standard deviation	2126	1417
defect, n(%)		
yes	44,100 (95.3)	12,863 (91.9)
no	2,196 (3.7)	1,134 (8.1)
pre/post release, n(%)		
pre	29,610 (64.0)	8,528 (60.9)
post	16,686 (36.00)	5,469 (39.1)
components involved		
mean	2.45	2.49
range	0 - 136	0 - 228
standard deviation	5.90	6.38
functions involved		
mean	29.47	32.40
range	1 - 21,740	1 - 31,630
standard deviation	291.55	403.26
developer country[2], n(%)		
country 1	30,965 (66.9)	7,966 (56.9)
country 2	7,596 (16.4)	2,429 (17.4)
country 3	3,732 (8.1)	1,044 (7.5)
country 4	2,272 (4.9)	1,032 (7.4)
country 5	866 (1.9)	735 (5.3)
symptom tag[2], n(%)		
program defect	9,558 (20.6)	2,830 (20.2)
test failed	9,486 (20.5)	3,811 (27.2)
function needed	8,783 (19.0)	2,775 (19.8)
incorrect i/o	4,079 (8.8)	482 (3.4)
core dump	2,312 (5.0)	378 (2.8)

[1] Time-to-deliver is presented in time units due to confidentiality.
[2] Only the five most common values presented in the table.

4.2 Non-parametric analysis

After the descriptive analysis of the data, we generated Kaplan-Meier and Nelson-Aalen survival and cumulative hazard estimators, respectively. The steep drop of the Kaplan-Meier curve presented in Figure 4.1 indicates that a big proportion of the issue reports do not survive for a long time. In Figure 4.2 we applied a logarithmic transformation on our data (which is a standard visualization enhancement technique that preserves the order of the observations while making outliers less extreme). Figure 4.1 and 4.2 suggest that almost 60% of the issue reports suggest that get resolved relatively quickly. However, the flattening of the Kaplan-Meier curve implies that those that survive beyond 150 time units are expected to take a significantly longer time to be resolved. A small proportion of the observations stretches the survival curve to the right side of the x-axis; for the first three releases (which is our train set), 1% of the issue reports were resolved in more than 174 time units.

Accordingly, in the cumulative hazard graph on the right of Figure 4.1, there is also a

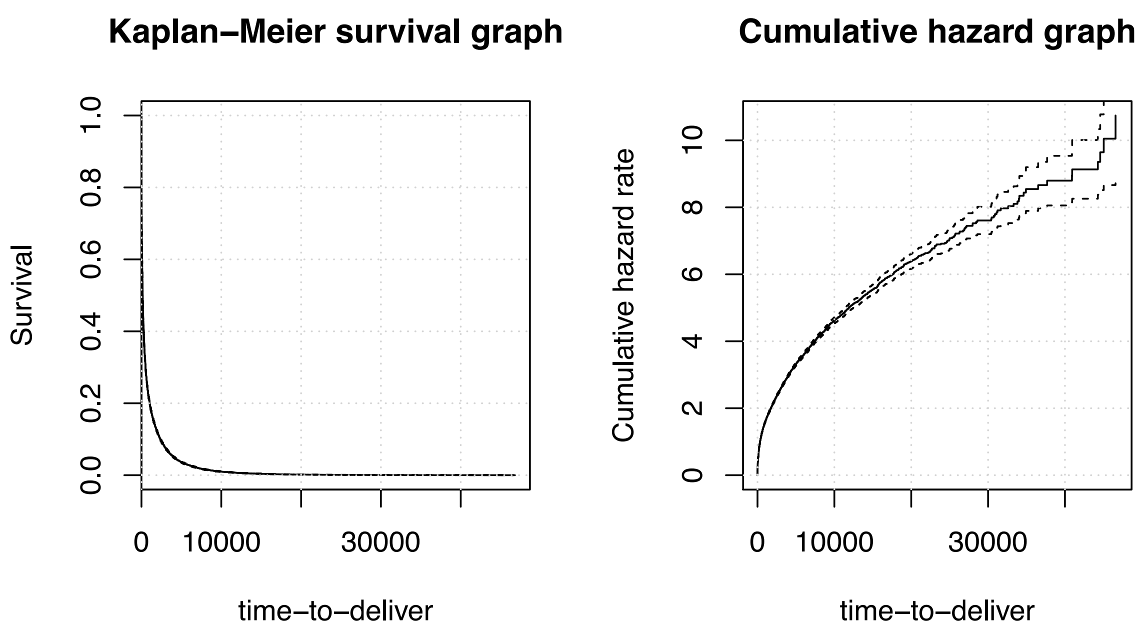

Figure 4.1: Kaplan-Meier and cumulative hazard curves on the empirical data of the first three releases.
Dotted lines represent confidence intervals.

steep slope at the beginning of the curve, proving that there is a time dependency on the hazard of issue reports being resolved. The earlier the distance from the date of an issue being reported, the more likely is for the issue to be solved. However, the slope/gradient of the curve is decreasing over time, hence the hazard as the time progresses lowers. Similar to the Kaplan-Meier curve, reported issues that survive after a certain time are more likely to remain unresolved for a long time.

4.3 Semi-parametric analysis

As discussed earlier, an alternative approach to modeling the effect of explanatory variables on a time-to-event variable is by measuring their effects on the hazard of the event. In this section we will investigate the application of the proportional hazards model on the time-to-deliver variable. We will differentiate between two models: one that ignores the variation between developers (the *fixed-effect* model) and one that assumes that there is variation across the developers due to their unobserved characteristics (the *random-*

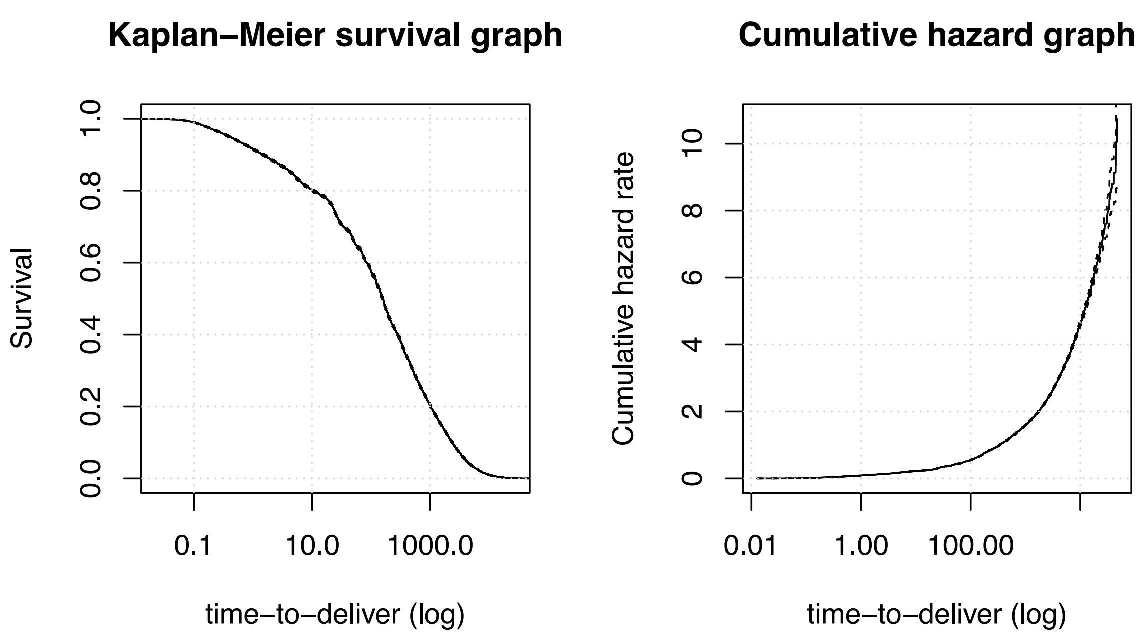

Figure 4.2: Log transformation on the x-axis of Figure 4.1 for clarity. Dotted lines represent confidence intervals.

effects model). In both cases the fixed-effect attributes that we used are the following:

- defect or feature,
- pre or post release,
- components involved,
- functions involved,
- developer country,
- symptom tag.

Detailed description of each attribute has been given in Section 4.1.

4.3.1 Fixed-effect models

The effect of each explanatory variable on the (log) hazard is given in Table 4.3. In general, the higher the hazard ratio, presented as $\log(HR)$ in the table, the greater the likelihood of the issue to be resolved (and in consequence the time-to-deliver will be smaller). A negative $\log(HR)$ implies a negative effect of the variable on the likelihood of resolution.

The coefficients in a proportional hazards model can also be interpreted as the logarithmic ratio of the hazard, given the characteristics of x, over the baseline hazard. $\beta = \log(HR) = \log((h|x = 1)/(h|x = 0))$ for x being a binary variable; when $x = 0$ the hazard rate is equal to the baseline hazard $h(0)$.

Issue reports that correspond to defects have lower hazard against the ones that correspond to feature implementation, hence defects will need more time to get resolved. Reports that have been submitted before the date that the next release is launched have a higher hazard to be delivered, which comes in line with our hypothesis *H1* in Chapter 3.1. The larger the number of components involved in the resolution of an issue report – the larger the hazard. Contrarily, although with a smaller effect, the number of functions involved reduce the hazard of the issue report resolution. The model estimated considerable differences between the hazard rates across the different countries the development took place (with county #1 being the reference one). For all categorical variables — developer country and symptom tag — an interpretation approach would suggest sorting of the different category levels based on their relative effect on the hazard of delivery compared to the baseline developer country. We observe a negative hazard rate for all symptom tags (Table 4.3). This implies that issues with a symptom tag of *build failed*, against which all the other tags are compared, has a very high hazard rate of being resolved. The symptom with the lowest log hazard ratio compared to the *build failed* tag, was the *docs incorrect* symptom tag. Similar, the rest of the symptom tags can be interpreted.

4.3.2 Mixed-effects/frailty models

The differences between the fixed-effect and the mixed-effects models with respect to the estimated hazard ratios is reflected in Table 4.3, where both fixed-effect and random-effects models are presented.

By observing the hazard ratio estimates of the two models, we infer that adding the random-effect component has limited effect on the fixed-effect estimates. It should be noted however that the interpretation of the fixed-effect terms are now conditional on the random-effect value. Although there are some differences in their values, the degree and statistical significance of the change is minor.

Therefore, inclusion of the random-effect variable did not result in serious implication on the rest of the coefficients. Although the inference from the hazard ratios has not changed, the magnitude of the variance of the random-effects term (σ^2), shown in Table 4.3, indicates that there exist considerable heterogeneity across the individual developers. Finally, the reduction on the AIC value provides a better fit for the random-effects model against the fixed-effect model.

4.4 Parametric analysis

As discussed in Section 3.5.5, transformations on the response variable might be able to assist in dealing with our extremely skewed distribution that we identified in Figure 4.1. Additionally, a significant advantage of the fully parametric models, that we are utilizing, is that the model is fitted directly on the duration variable rather than the hazard. Especially in the case of time-to-deliver estimation, predicting time, instead of hazard rate, is an important advantage of these models.

Table 4.3: Regression estimates of Cox proportional hazard models.

Variable name	fixed-effect log(HR) (SE)	mixed-effects log(HR) (SE)
defect	-0.058 (0.022)	-0.057 (0.024)
pre release	0.401 (0.010)	0.283 (0.013)
component count	0.018 (0.001)	0.005 (0.001)
function count	-0.0001 (-0.00003)	-0.00005 (0.00002)
developer country (baseline: 1)		
10	-0.543 (0.236)	-0.840 (0.464)
11	0.460 (0.057)	0.300 (0.609)
12	0.200 (0.069)	0.430 (0.437)
14	-0.961 (0.700)	-1.27 (0.841)
19	-0.899 (0.707)	-1.249 (0.932)
2	0.223 (0.012)	0.112 (0.057)
3	-0.208 (0.020)	-0.349 (0.084)
4	-0.152 (0.094)	-0.243 (0.237)
5	-0.201 (0.017)	-0.172 (0.069)
6	0.178 (0.069)	-0.180 (0.205)
8	0.551 (0.310)	0.484 (0.685)
9	0.041 (0.034)	0.149 (0.115)
symptom tag (baseline: build failed)		
core dump	-1.400 (0.034)	-1.369 (0.036)
corrupt dbase	-1.470 (0.061)	-1.385 (0.06)
docs incorrect	-1.801 (0.12)	-1.760 (0.125)
function needed	-1.230 (0.029)	-1.177 (0.031)
incorrect i/o	-1.511 (0.031)	-1.439 (0.033)
incorrect xlat	-1.610 (0.201)	-1.537 (0.204)
install add remove files	-1.516 (0.142)	-1.237 (0.151)
install configuration	-1.682 (0.101)	-1.443 (0.106)
install failed	-1.460 (0.063)	-1.270 (0.067)
intgr problem	-1.128 (0.085)	-1.129 (0.091)
lost data	-1.714 (0.091)	-1.671 (0.093)
mixed code releases	-0.812 (0.164)	-0.651 (0.175)
non standard	-1.678 (0.062)	-1.711 (0.064)
not to spec	-1.484 (0.037)	-1.519 (0.040)
obsolete code	-1.685 (0.060)	-1.635 (0.067)
performance	-1.725 (0.030)	-1.700 (0.037)
planned xlat	-0.753 (0.500)	-0.708 (0.503)
plans incorrect	-1.450 (0.091)	-1.434 (0.096)
program defect	-1.398 (0.029)	-1.390 (0.031)
program loop	-1.545 (0.068)	-1.466 (0.069)
prog suspended	-1.453 (0.041)	-1.449 (0.043)
reliability	-1.574 (0.043)	-1.534 (0.045)
test failed	-1.367 (0.028)	-1.354 (0.031)
usability	-1.575 (0.034)	-1.516 (0.037)
AIC	894785	887928
σ^2	-	0.368

SE denotes standard error.
σ^2 – variance of the random-effects term.

We applied four different distributions on the empirical time-to-deliver data (exponential, Weibull, lognormal, and loglogistic). Figure 4.3 illustrates the fit for the four distributions against the Kaplan-Meier curve. Graphically, we notice that the Weibull curve is almost identical to the non-parametric Kaplan-Meier curve.

Formal comparison between the models using the AIC indicates that the best fitting model with no explanatory variables is the model assuming a Weibull distribution, with an AIC of 644,482. At the same time, the worst AIC value for the estimated models, is the exponential, with a value of 718,783. The lognormal and loglogisitc distribution had an AIC of 649,455 and 650,031 respectively.

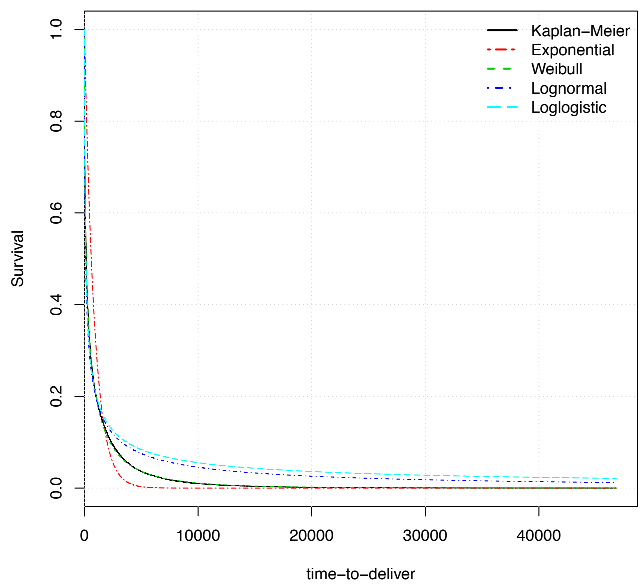

Figure 4.3: Survival curves of each distribution compared to the empirical data.

Parametric regression

Subsequently, we follow a parametric regression approach to understand the impact of different explanatory variables on time-to-deliver as well as to generate time-to-deliver predictions. The variables we consider as explanatory are the same to those in the semi-parametric Section 4.3. Additionally,we built a basic linear model to investigate the extent of deviation from the parametric models from a simple regression solution.

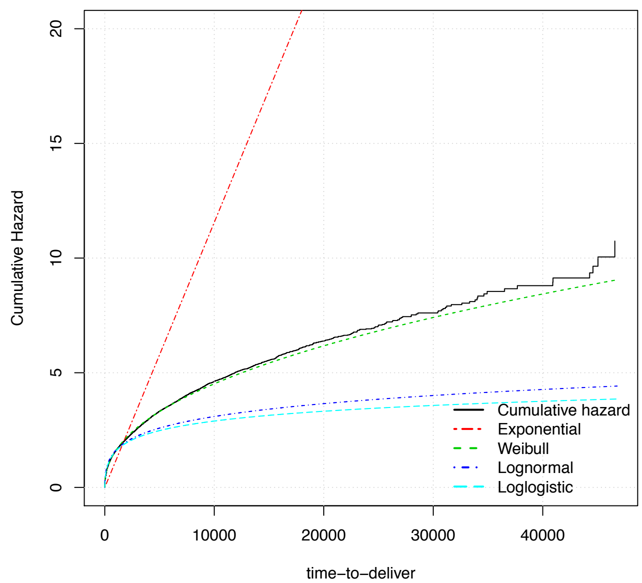

Figure 4.4: Cumulative hazard curves of each distribution compared to the empirical data.

The contribution of each of these variables is considered significant in the predictive efficiency of the model. This was derived by the p-values (< 0.01 in all cases) that the models yield upon creation. At the same time application of a stepwise elimination algorithm (based on the AIC values of the models) on the full model did not eliminate any of the explanatory variables.

For accuracy metric, due to the restriction of a balanced dataset, we set the median of the time-to-deliver as our time threshold. This resulted in an even categorization of our data between the two classes – fast below the threshold and slow above.

4.4.1 Fixed-effect models

Table 4.4 presents the results of the best fitting models for each of the distribution assumptions along with their corresponding γ parameters. We can observe that even after explanatory variable inclusion, the Weibull is the best fitting model based on the AIC. The reason AIC was preferred as an assessment of goodness-of-fit against R^2 is because we cannot use R^2 values as a measure for parametric models (as discussed in

Table 4.4: Parametric regression estimates of fixed-effect models.

Variable name	Exponential β (SE)	Weibull β (SE)	Lognormal β (SE)	Loglogisitc β (SE)
(Intercept)	4.920 (0.036)	3.449 (0.076)	2.354 (0.089)	2.12 (0.089)
defect	0.018 (0.023)	0.128 (0.048)	0.180 (0.056)	0.245 (0.059)
pre release	-0.463 (0.010)	-0.841 (0.021)	-1.395 (0.025)	-1.315 (0.024)
component count	-0.030 (0.001)	-0.040 (0.002)	-0.045 (0.002)	-0.051 (0.003)
function count	0.0001 (0.00002)	0.0002 (0.00005)	0.0003 (0.00004)	0.0003 (0.00004)
developer country (baseline: 1)				
10	0.526 (0.236)	1.159 (0.492)	1.537 (0.575)	1.587 (0.569)
11	-0.868 (0.058)	-0.986 (0.120)	-1.279 (0.140)	-1.352 (0.146)
12	-0.362 (0.070)	-0.419 (0.145)	-0.704 (0.170)	-0.628 (0.179)
14	1.407 (0.707)	2.015 (1.474)	2.894 (1.724)	2.745 (1.455)
19	1.574 (0.707)	1.882 (1.474)	2.368 (1.724)	2.218 (1.487)
2	-0.468 (0.013)	-0.472 (0.027)	-0.458 (0.032)	-0.45 (0.031)
3	0.248 (0.022)	0.434 (0.046)	0.814 (0.053)	0.708 (0.049)
4	-0.211 (0.095)	0.312 (0.198)	1.162 (0.231)	1.088 (0.209)
5	0.295 (0.018)	0.421 (0.037)	0.561 (0.043)	0.499 (0.041)
6	-0.488 (0.07)	-0.378 (0.146)	-0.270 (0.171)	-0.272 (0.161)
8	-2.184 (0.317)	-1.177 (0.662)	-0.823 (0.773)	-0.612 (0.760)
9	-0.325 (0.035)	-0.09 (0.072)	0.155 (0.084)	0.200 (0.082)
symptom tag (baseline: build failed)				
core dump	2.131 (0.034)	2.978 (0.071)	3.454 (0.083)	3.695 (0.079)
corrupt dbase	2.214 (0.062)	3.127 (0.129)	3.551 (0.150)	3.813 (0.141)
docs incorrect	2.984 (0.123)	3.815 (0.257)	3.788 (0.301)	4.187 (0.301)
function needed	2.019 (0.029)	2.615 (0.060)	2.428 (0.071)	2.737 (0.070)
incorrect i/o	2.407 (0.031)	3.204 (0.065)	3.426 (0.076)	3.736 (0.074)
incorrect xlat	2.333 (0.202)	3.410 (0.421)	3.866 (0.492)	4.319 (0.450)
install add remove files	2.413 (0.143)	3.215 (0.298)	3.734 (0.348)	3.797 (0.320)
install configuration	2.449 (0.102)	3.562 (0.212)	4.291 (0.248)	4.538 (0.221)
install failed	2.261 (0.063)	3.099 (0.131)	3.565 (0.153)	3.773 (0.143)
intgr problem	1.755 (0.085)	2.405 (0.178)	2.671 (0.208)	2.946 (0.202)
lost data	2.732 (0.091)	3.623 (0.190)	4.029 (0.223)	4.239 (0.209)
mixed code releases	1.025 (0.165)	1.725 (0.343)	1.557 (0.401)	1.755 (0.436)
non standard	2.878 (0.062)	3.544 (0.129)	3.286 (0.150)	3.676 (0.156)
not to spec	2.406 (0.038)	3.148 (0.078)	3.254 (0.092)	3.594 (0.091)
obsolete code	3.095 (0.065)	3.572 (0.135)	3.13 (0.158)	3.437 (0.169)
performance	2.677 (0.035)	3.651 (0.073)	4.061 (0.085)	4.384 (0.082)
planned xlat	0.560 (0.501)	1.619 (1.044)	1.822 (1.220)	2.340 (1.236)
plans incorrect	2.280 (0.092)	3.081 (0.192)	3.254 (0.224)	3.657 (0.222)
program defect	2.165 (0.029)	2.965 (0.060)	3.225 (0.07)	3.567 (0.068)
program loop	2.304 (0.068)	3.275 (0.142)	3.822 (0.165)	4.074 (0.153)
prog suspended	2.134 (0.042)	3.081 (0.087)	3.795 (0.101)	3.959 (0.095)
reliability	2.426 (0.043)	3.334 (0.090)	3.727 (0.105)	4.049 (0.100)
test failed	2.018 (0.029)	2.902 (0.060)	3.320 (0.070)	3.605 (0.068)
usability	2.525 (0.034)	3.337 (0.072)	3.521 (0.084)	3.870 (0.082)
γ parameter[1]	1	2.0849	2.4373	1.3632
AIC	702514	636975	638941	638716

SE denotes standard error.

Section 3.7.2). Nevertheless, the R^2 of the linear model that we built (where response variable was $\log(y)$) with the same explanatory variables, was 0.204. This implies that the model explains $\approx 20\%$ of the variability. The number is low, but is not uncommon in modelling complex systems (e.g., in economics, medicine, and psychology [34, 8]). Nonetheless, the statistical significance of the coefficients allows us to draw important conclusions on the effect of each explanatory variable.

We can, therefore examine the effect of each explanatory variable on the (log) time-to-deliver as well as the differences of the coefficients among the four different parametric models in Table 4.4. A first general observation is that, for the majority of the explanatory variables, the direction of the effect is the same among all the different parametric models. The explanatory variable that seems to have the greater effect on the time-to-deliver is the symptom tag that is associated with the issue report. The *non standard* tag is the one that affects the time-to-deliver the most in a negative way; always compared with the baseline tag which is the *build failed*. For both categorical variables (developer country and symptom tag), the results that were observed in the proportional hazards models concur in most cases. An interesting comparison can be made among the components and the functions that need to be adjusted for a report to get resolved. For the former, the greater the number of the components – the shorter the time until resolution. Contrary, more functions involved in the resolution of a report, more time necessary – although the significance of the functions variable is much less significant. In addition, when a report is flagged as needed to be resolved before the next release is launched, the time-to-deliver of this report is shorter.

Although the differences among the four different distributions are not identical, their effect is in all cases in a similar direction.

Table 4.5 illustrates the performance metrics that we utilized for model comparison.The results presented in this section reflect performance metrics while evaluating the models internally, on the train set. Considering the standard deviation of the residuals (σ_ϵ) and the Kendall rank correlation coefficient (RCC) the simple linear model yields competitive results. While assessing the residual standard deviation, the exponential model is the best performer among the parametric models. Regarding the Kendall rank correlation coefficient, the Weibull model, which also has the best AIC value, provides the better result. However, the superiority of the simple linear model is still obvious. Finally, when using the models as fast/slow classifiers of the time-to-deliver, the lognormal model is the one that has the best performance based on the accuracy metric. In this case, the lognormal model significantly outperforms the simple linear model as well, being able to correctly classify approximately 65% of the observations.

Table 4.5: Performance metrics for fixed-effect models.

	Linear model	Exponential	Weibull	Lognormal	Loglogistic
σ_ϵ	2084	2087	2095	2106	2112
RCC	0.197	0.157	0.161	0.126	0.128
ACC	0.562	0.553	0.586	0.646	0.618
AIC	-	702514	636975	638941	638716

Based on these findings, an interesting insight is the inconsistency of the best candidate model. This is based on the criteria we measure performance. Naively, we can say that these metrics "challenge" the models in a different manner - from the more difficult challenge of the residual standard deviation, to the less difficult of rank correlation coefficient, with accuracy being the less challenging one. The residual standard deviation assesses the predicting accuracy of each observation, while Kendall rank correlation "forgives" non-detailed prediction, as long the ranking is the same. Finally, fast/slow classification of the issue reports, as captured by accuracy, ignores both prediction precision close to the real values and ranking, and focuses on the time threshold that is set.

4.4.2 Mixed-effects/frailty models

In a similar approach as in the proportional hazards models, we introduce the developer as the random-effects attribute in our models.

The results of these mixed-effects models are presented in Table 4.6. Along with the coefficients of the fixed-effect terms and their standard errors, the variance of the random-effects variables is also presented. Based on the values of the coefficients there are no significant changes compared to what we have seen in the previous results discussed. The effect of the defect/feature differentiation, as well as the number of functions involved are the least significant variables. The impact however is considerable, in the direction of making the report to get resolved faster, for both pre release reports and proportional to the number of components involved. Additionally, we observe that the introduction of the random-effects term resulted in a notable and some times in a directional change on some of the developer's countries. For example, by noticing the values of country 11 in Table 4.4 gives us the impression that issue reports developed in this country tend to get resolved faster than the baseline country (#1), as well as compared with the majority of the rest of the countries. However, in the mixed-effects model, this is no longer true; the standard error around the coefficient is high in all cases, therefore we are driven to the result that the variability within this country is high. The explanation could be based on the definition of mixed-effects models given on Section 3.6 and their difference with the fixed-effect models.

The AIC values propose a different distribution as the best candidate model, always compared to the fixed-effect models. The loglogistic distribution has the best AIC value, followed by the lognormal. The Weibull model, which had the best AIC values so far, comes third while assessing the goodness-of-fit based on AIC value. The variance of the random-effects term for the loglogistic distribution is also considerably higher than the one of the Weibull model. This implies that in the loglogistic model the dispersion of the developers is assumed to be higher.

Accordingly, Table 4.7 represents performance metrics of the mixed-effects models for internal validation (train set). In this case, inclusion of the random-effects term results in superiority of the parametric models, against the simple linear model we fitted for comparison purposes. In this case, the exponential model is the one performing best, in terms of residual standard deviation, as well as Kendall rank correlation coefficient. However, the loglogistic model is the one that can classify more accurately as fast or slow fix.

By comparing the performance metrics of the fixed-effect with the mixed-effects models, Tables 4.5 and 4.7 respectively, we can observe an improvement in the latter. Although the values of the residual standard deviation are not significantly improved, both Kendall rank correlation coefficient and classification accuracy are substantial. However, it is also worth mentioning that, better fit of the models is expected anyway since the number of explanatory variables is increasing (addition of the developer). This is also the reason of the results that are presented in Table 4.8. Although we have not discussed inclusion of the developer as a fixed-effect variable, we conducted this additional experiment and presenting the results in this section only, to show how information about

Table 4.6: Parametric regression estimates of mixed-effect models.

Variable name	Exponential β (SE)	Weibull β (SE)	Lognormal β (SE)	Loglogistic β (SE)
(Intercept)	4.220 (0.070)	3.371 (0.088)	2.293 (0.103)	2.087 (0.104)
defect	-0.018 (0.025)	0.121 (0.048)	0.272 (0.055)	0.330 (0.057)
pre release	-0.284 (0.014)	-0.561 (0.026)	-0.972 (0.030)	-0.936 (0.029)
component count	-0.012 (0.001)	-0.012 (0.002)	0.001 (0.003)	0.000 (0.003)
function count	0.00003 (0.00005)	0.0001 (0.00004)	0.0001 (0.00004)	0.0001 (0.00004)
developer country (baseline: 1)				
10	1.488 (0.867)	1.421 (0.827)	1.825 (1.066)	1.997 (1.048)
11	-0.345 (1.324)	-0.851 (0.950)	-1.423 (1.217)	-1.411 (1.271)
12	-0.266 (0.938)	-0.713 (0.682)	-0.836 (0.871)	-1.011 (0.907)
14	1.909 (1.182)	2.251 (1.545)	3.019 (1.780)	2.916 (1.560)
19	2.222 (1.500)	2.168 (1.672)	2.557 (1.975)	2.451 (1.845)
2	-0.258 (0.112)	-0.291 (0.092)	-0.199 (0.115)	-0.190 (0.118)
3	0.462 (0.164)	0.506 (0.134)	0.796 (0.167)	0.781 (0.171)
4	0.064 (0.461)	0.271 (0.387)	0.953 (0.480)	0.965 (0.484)
5	0.121 (0.126)	0.318 (0.107)	0.265 (0.130)	0.232 (0.135)
6	0.345 (0.407)	0.163 (0.347)	0.456 (0.438)	0.393 (0.44)
8	-1.563 (1.360)	-1.235 (1.129)	-1.084 (1.398)	-0.926 (1.445)
9	-0.451 (0.220)	-0.283 (0.179)	-0.164 (0.221)	-0.114 (0.229)
symptom tag (baseline: build failed)				
core dump	2.166 (0.038)	2.755 (0.070)	3.117 (0.078)	3.332 (0.076)
corrupt dbase	2.194 (0.067)	2.783 (0.125)	3.062 (0.140)	3.312 (0.132)
docs incorrect	3.054 (0.127)	3.522 (0.245)	3.240 (0.275)	3.467 (0.282)
function needed	2.036 (0.034)	2.374 (0.061)	2.160 (0.068)	2.419 (0.068)
incorrect i/o	2.352 (0.035)	2.888 (0.065)	3.030 (0.072)	3.294 (0.072)
incorrect xlat	2.512 (0.206)	3.083 (0.398)	3.361 (0.450)	3.689 (0.420)
install add remove files	1.932 (0.156)	2.506 (0.293)	2.943 (0.344)	3.115 (0.316)
install configuration	2.192 (0.108)	2.918 (0.206)	3.345 (0.232)	3.566 (0.209)
install failed	2.023 (0.071)	2.566 (0.132)	2.744 (0.147)	2.985 (0.140)
intgr problem	1.803 (0.092)	2.274 (0.177)	2.439 (0.207)	2.658 (0.201)
lost data	2.733 (0.096)	3.330 (0.182)	3.665 (0.205)	3.845 (0.194)
mixed code releases	0.962 (0.183)	1.314 (0.338)	1.003 (0.385)	1.098 (0.424)
non standard	3.101 (0.067)	3.401 (0.125)	3.041 (0.139)	3.401 (0.146)
not to spec	2.574 (0.042)	3.038 (0.078)	2.998 (0.088)	3.268 (0.088)
obsolete code	3.010 (0.070)	3.272 (0.132)	2.877 (0.149)	3.194 (0.157)
performance	2.748 (0.039)	3.414 (0.072)	3.684 (0.081)	3.943 (0.079)
planned xlat	0.569 (0.504)	1.453 (0.982)	1.883 (1.109)	2.278 (1.099)
plans incorrect	2.396 (0.099)	2.879 (0.188)	3.003 (0.212)	3.305 (0.209)
program defect	2.248 (0.033)	2.790 (0.06)	3.088 (0.067)	3.336 (0.067)
program loop	2.267 (0.071)	2.946 (0.135)	3.387 (0.152)	3.591 (0.142)
prog suspended	2.286 (0.046)	2.908 (0.085)	3.378 (0.095)	3.499 (0.091)
reliability	2.479 (0.048)	3.073 (0.088)	3.273 (0.098)	3.548 (0.096)
test failed	2.160 (0.033)	2.722 (0.060)	3.017 (0.067)	3.231 (0.066)
usability	2.508 (0.039)	3.038 (0.072)	3.158 (0.080)	3.436 (0.079)
scale	1.000	1.951	2.206	1.230
σ^2	1.745	0.889	1.462	1.593
AIC	682761	631740	631266	630908

SE denotes standard error.
σ^2 – variance of the random-effects term

Table 4.7: Performance metrics for mixed-effects models.

	Linear model	Exponential	Weibull	Lognormal	Loglogistic
σ_ϵ	2026	2025	2048	2077	2070
RCC	0.305	0.314	0.289	0.255	0.260
ACC	0.569	0.605	0.671	0.693	0.703
AIC	-	682761	631740	631266	630908

Table 4.8: Performance metrics for fixed-effect models, including *developer*.

	Linear model	Exponential	Weibull	Lognormal	Loglogistic
σ_ϵ	2084	2041	2036	2068	2064
RCC	0.197	0.297	0.289	0.241	0.247
ACC	0.562	0.606	0.675	0.695	0.703
AIC	-	683334	632099	631702	631276

developer can be leveraged by a linear model. However, the main disadvantages of the linear fixed-effect model including developer, is that (i) it cannot make predictions for new developers and (ii) the model will return null values as predictor estimates when the records of a single developer are not sufficient in number. Therefore, by comparing Tables 4.7 and 4.8, we can see that the goodness-of-fit of the mixed-effects models is better based on the AIC values. The residual standard deviation and the accuracy metrics are very close in most cases, however, for rank correlation we observe a slight superiority for the mixed-effects models.

4.5 Validation

A common way of validating the fit of the model is by evaluating its predicting performance on a different dataset. Although there are different ways of conducting this validation, we used data splitting, based on the chronological order of the releases that we have available; we can refer to this split as quasi-chronological or pseudo-temporal split. As discussed before, we used the first three releases as the train set and the fourth as the validation/test set. Although there are difficulties and restrictions while validating the results, data partitioning is a common way of assessing the predictive performance.

Especially in this case and due to the nature of the data, there might still be overlapping information among the different releases. For example, since the development process is continuous, there are issue reports that were submitted during the first three releases but closed sometime within the time-window of the fourth release.

Using the same parametric models that were built on the train set, we evaluated their predictive performance by applying them on the test set and then comparing the predicted values with the real ones. The results for both fixed-effect and mixed-effects models are presented in Tables 4.9 and 4.10 respectively.

Table 4.9: Performance metrics for fixed-effects models for external validation.

	Linear model	Exponential	Weibull	Lognormal	Loglogistic
σ_ϵ	1409	1414	1490	1540	1693
RCC	0.192	0.191	0.098	0.048	0.035
ACC	0.530	0.544	0.575	0.663	0.663

Table 4.10: Performance metrics for mixed-effects models for external validation.

	Linear model	Exponential	Weibull	Lognormal	Loglogistic
σ_ϵ	1418	1447	1406	1406	1406
RCC	0.197	0.183	0.197	0.180	0.183
ACC	0.569	0.551	0.611	0.677	0.675

Although the fit looks better when comparing the standard deviation values with these from the internal validation, we have to be careful with data interpretation. As already mentioned above, the nature of the data and the selection of the train/test set could be the reason of the residual standard deviation decrease. Overlap of the data and differences in the descriptive analysis of the response variable that was identified in the descriptive analysis Table 4.2, is primarily the reasons of the differences that we observe (compare standard deviation of time-to-deliver for train vs test sets in Table 4.2).

The fit of the simple linear model still looks competitive in most cases, except from the accuracy. Although the results of the fixed-effect models seem inconsistent, in the mixed-effect models the efficiency is improved, or at least consistent, between the different paramteric models. As a final observation we can conclude to the following: loglogistic models yields the best accuracy for both training and validation,; mixed-effects are performing better in all cases and finally, accuracy in validation is slightly worse than that for training – which was expected based on the similarities of the datasets.

4.6 Considering censoring

In the presence of censoring we re-applied all of the validation processes that were described until now. As mentioned in Section 3.8, artificial truncation resulted in a significantly reduced dataset. Figure 4.5, represents this impact on the sample size, for each censoring proportion (achieved by truncation); from the full dataset — 46,296 records — with no censoring, to a reduced one — 2,257 records — in order to achieve 20% of censoring. The long tailed distribution of the response variable, in combination with the majority of issue reports being resolved very quickly (compared with the mean value of the response variable) results in this sharp reduction. In order to be able to achieve the maximum censoring proportion (20%) we had to go back nine years – in the twelve year time span we are studying.

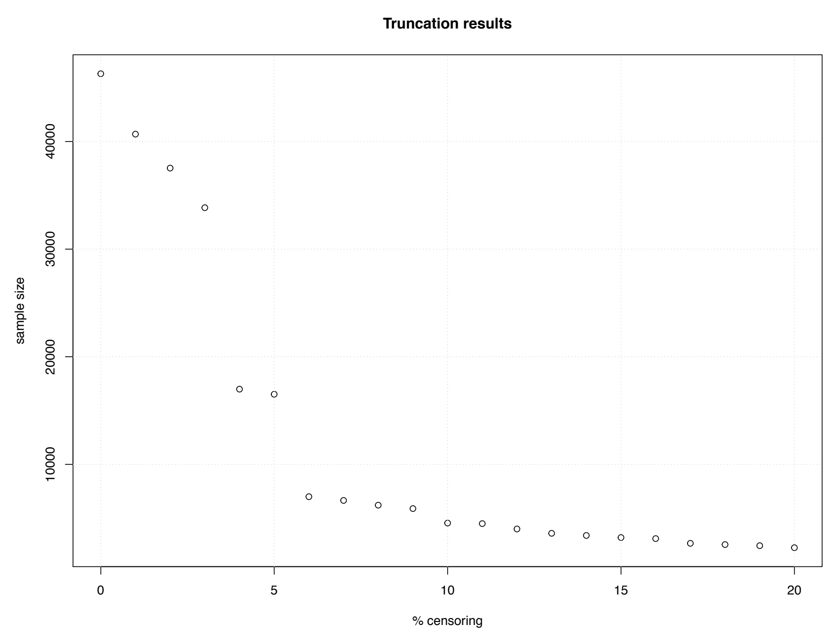

Figure 4.5: Artificial truncation
Impact of the censoring simulation on the sample size.

The results of these experiments are presented in the Appendix A and Tables A.1 to A.10. The Tables can be read as follows:

- First column represents the censoring proportion that we artificially achieved by truncation of the data.
- Second column indicates the sample size for each censoring proportion. The reason of the sample degradation as the censoring increases is given above.
- In the remaining ten columns (eight columns for Tables A.1 and A.6 since we did not calculate AIC for basic linear models because we cannot use them for model comparison, as discussed in Section 3.7.1) the values of each metric are presented – residual standard deviation, Kendall rank correlation coefficient, and AIC.
  - For standard deviation and Kendall rank tables, the third column provides results of a simple linear model that was built and presented as a baseline criterion. However, and since a linear model cannot incorporate censored information, the deliver time-stamp of the censored observations was set as the censoring cut-off point. For the AIC tables, this column is missing since AIC comparison between parametric and linear models is not valid (as discussed in Section 3.7.1).
  - Similarly, for standard deviation and Kendall rank tables only (AIC tables are missing this column as well), the fourth column presents results of a simple linear model with the censored observations filtered out. The number of censored observations can be calculated by multiplying the censoring proportion with the sample size.
  - Columns five, seven, nine, and eleven (three, five, seven, and nine for AIC) represent results of the fully parametric models: exponential, Weibull, lognormal, and loglogistic respectively.
  - Consequently, the even columns from six to twelve (four to ten for AIC) represent the fully parametric models that completely drop/filter out the censored observations.

For the AIC values, note that we can only compare them on each row independently and at the same time among the same type of columns (*cens* or *drop* columns). The definition of the AIC does not allow it to be a criterion when the sample is different – which is true for censored against dropped columns in all tables.

Some of the results values in the tables are missing, the reason of this inconsistency is the decreased sample size, which leads to reduced information, hence the inability to generate predicted values. Despite this numerical instability of the models, we are presenting the results regardless, with the goal to get as much information as possible.

The exponential distribution models are performing slightly better than the rest, which contradicts with the results that were observed on the empirical data (Weibull looked to fit better). However, and as we already identified above, the differences are minor between the majority of the models. Models that drop the censored observations are reducing their standard deviation as the censoring proportion increases – sample size decreases significantly. At the same time, the models that incorporate large number of censored observations are unstable - the residual standard deviation is increasing extremely in these cases. This is an additional evidence of the right skewed distribution of our data and echoes the fact of the long but, at the same time, thin tail. The results are similar for the Kendall rank correlation coefficient.

While trying to compare differences between fixed and mixed-effects models, in respect to the effect of censoring, we can argue the following. Inconsistencies seem to align, after the censoring proportion is greater than 12% for internal validation. However, these inconsistencies show up earlier (i.e., censoring proportion greater than 5%) when it comes to external validation, due to the smaller sample. In both cases though, internal and external validation, the mixed-effects models seem to be more efficient based on all metrics. Although the differences are minor in respect of residual standard deviation, rank correlation coefficient is significantly improved in all mixed-effect models. Comparison of AIC values between Tables A.1 and A.6 is valid, since the response variable is not altered and the explanatory variables of the fixed-effect models is a subset of the mixed-effects ones. Therefore, all mixed-effects models have a better goodness-of-fit than the fixed-effect.

4.7 Discussion

Regarding *RQ1* "How incorporation of random-effects models and/or censored data influence performance of a model that predicts time-to-deliver?". We extensively analyzed and compared fixed against mixed-effects models, as well as differences in the proportion of censored information. For the former, we can confidently say that inclusion of the random-effect term is having a positive impact, enhancing the predicting power and the goodness-of-fit of the models. We compared the performance of the models in various aspects, and in all cases the superiority of the mixed-effects models was proven.

Additionally, mixed-effect models will be superior against fixed-effect ones, in cases where prediction is necessary on new data, that potentially new developers have been introduced. In such cases, fixed-effect models will not be able to use historical information about developers and make predictions at all.

However, the decision is not only based on AIC values and goodness-of-fit metrics but also on expert's opinion. If the long tails make no sense then even if the AIC is correct the model makes little sense in practice.

For incorporation of censored data, the results are unclear. As extensively analyzed and illustrated, non-normality of our data is the main reason of the inconsistent, and in some cases, incomplete results. We argue that the dataset under study is not ideal for artificial truncation. The long tail dominates in all cases, especially in those of increased censoring proportions, which resulted in the extreme values that show up in Appendix A result tables. The phrase "the ends justify the mean" might be appropriate as an explanation to these results – meaning that the long tail is having a great impact on the dataset.

For *RQ2* "How geographical, churn, and complexity factors affect the duration of the time-to-deliver?", we thoroughly explored the effect of each attribute on the response variable. From the effect of the hazard ratio to the extend of the time-to-deliver on the accelerated failure time models. Our results were consistent and in most cases concur with the theoretical hypothesis that we formed before proceeding with the practical implementation, as discussed in Section 4.7.1 below.

As for a final selection of the best model, we showed the advantages of parametric analysis. The ability to overcome all the violations and obstacles that were extensively analyzed is definitely an asset. However, based on the measured performance, the results were not considerably better than in the simpler case of the linear models. Lastly, selection of the most appropriate model might also depend on the desired outcome. Inconsistencies between the different parametric distribution models have also been analyzed.

4.7.1 Validation of Hypothesis

Based on the hypotheses that were preliminarily conducted and presented in Section 3.1, we re-iterate and compare with the findings after the theoretical methodology has been applied to the data.

For *H1* we showed that our hypothesis comes in line with the results for all models that were fitted. Issue reports marked as *pre* release are most likely to be resolved faster than *post* ones. At the same time, we utilized the information that we get from this variable as a proxy for severity/priority of the issue report which was hypothesized in *H3*.

Based on the coefficients of the components involved in an issue report, we observe a negative correlation – the more the components involved, the less the time necessary for delivery. This finding disproves our original hypothesis in *H2*. However, for the functions involved, our findings converge towards this direction – although the significance is reduced in the case of the functions.

Differentiation of defect or feature did not make a significant difference based on our findings, although we hypothesized that defects tend to get resolved faster than feature implementations in *H4*.

In our first categorical variable, we expected differences among different symptom tags *H5*. We were able to identify some differences among them, as well as to provide an order, from faster to slower, in terms of the duration effect on the time-to-deliver. Based on their brief description and on expert input, we could have also hypothesized the expectation of their difficulty.

For the random-effect term in our models, we anticipated a within group dependence and variation on the effect among different individuals *H6*. We discussed the final effect of the variable itself, as well as the effect on the rest of the explanatory variables. Additional insights, such as the variance among the developers, are also a considerable advantage that we considered when using this variable as a random-effects.

4.7.2 Stakeholder feedback

As part of this study, we not only evaluated the results ourselves, but also provided frequent feedback to the team responsible for the current development of the software under study. The prosperous communication and suggestions on further steps and improvements, resulted in some additions and re-iterations on the results presented, as well as in our future work suggestions. Additionally, feedback and information on details that are not readily available from the issue tracking system and the raw data, provided some adjustments and calibrations on the existing models. For example, shifting the slow/fast threshold to the current standards and the development process being followed, resulted in a significant improvement in classification accuracy. In Figure 4.6 the fluctuation of the accuracy performance is shown; shifting the threshold to the right of the median value, results in better accuracy. As illustrated in the same graph, the "current slow/fast threshold" represents the feedback we received from the development team and how they currently classify resolutions as fast or slow. However, additional measures and metrics are necessary, especially when the ratio between the two classes becomes imbalanced.

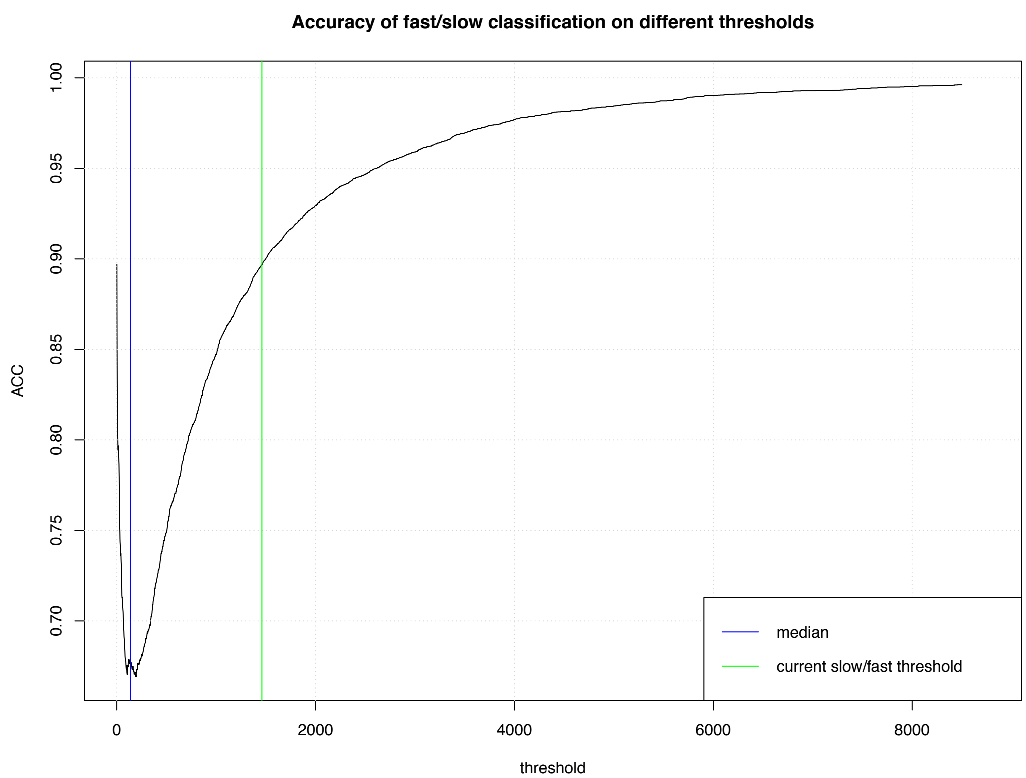

Figure 4.6: Accuracy fluctuation over different fast/slow thresholds. Thresholds above 8,500 get closer to 1 monotonically.

4.8 Threats to validity

The threats to validity for this study, as well as the ways to overcome them, are categorized and presented in this section.

Construct validity

We construct our dataset based on the data collected from the issue reporting system. The system captures a variety of events and activities happening in the organization. However, only a subset of this information was used to built our predictive models. To overcome this threat, we utilized a large subset of the entire software dataset – 4 releases developed over approximately 12 years.

Statistical validity

We utilized the R software environment for statistical computing and graphics [44]. In order to validate our results, we utilized different libraries wherever possible. To prevent biased results, we utilized quasi-chronological or pseudo-temporal (as defined in Section 4.5) as a validation technique.

Internal validity

In order to avoid researcher bias, we derived and followed strict automated processes for data extraction and processing. One of the most critical internal parts of our study is the artificial truncation process. Although the results are not consistent and optimal in the way we would expect incorporation of censored information to assist in the predicting performance, the truncation process replicates real conditions in the best way possible.

External validity

Generalizing our findings from a single software to different situations is one of our main future interests towards extending this work, although this might not be possible under different circumstances. However, the design of this study is based on the rationale of the critical case [52]. The methodological approach is easily reproducible for other software and at the same time readily available to be applied on software with similar attributes recorded by the issue tracking system.

Chapter 5

Conclusions, Summary and Future Work

In this study, we utilized survival analysis for estimation of the time-to-deliver, within a large scale commercial software.

We successfully leveraged three techniques, that although are well known and commonly used in other academic disciplines (i.e., engineering, financial, and health economic studies), have not been widely utilized in the Software Engineering field of study. Namely, we introduced use of survival analysis for time-to-deliver data, incorporation of random-effects models, and consideration of censored observations. We identified that these techniques are appropriate for our case and also encourage researchers to utilize them in similar cases, by providing the theoretical background and the limitations that these techniques help to overcome. Moreover, we provide the prototype tool implementing this approach.

We thoroughly described the advantages for the proposed methodologies: survival models to overcome linear regression limitations, incorporation of censored observations to enhance the sample size, and identification of within group dependence for the random-effects attribute. However, limitations still exist in the use of these techniques. In the case of survival analysis (since the most common way of dealing with survival data is the Cox proportional hazards models), interpretation of the hazard rate might be difficult when dealing with time-to-deliver data. For random-effects models, either these are incorporated in proportional hazards models or accelerated failure time models, the computational complexity increases dramatically – especially when compared to simple linear models. Finally, in censored data consideration, their inclusion in the predictive models can also be complex. Missing values and attributes of the censored observations that are not available when in the state of "censored" might change the model fitting "strategy".

If these limitations can be avoided though, integration of these practices can significantly improve the robustness and prediction power of the models. Additionally, they are an efficient way of avoiding the limitations that simple linear models will face in equivalent situations.

We believe that our approach can help practitioners to improve prediction of the time-to-deliver, simplifying resource planning. Our results are also of interest to theoreticians, showing applicability of survival analysis, incorporation of censored information, and introduction of a random-effects term, in a new domain.

5.1 Future work

We consider this work as a first step of application of survival analysis in the time-to-deliver estimation. Our plans for future work include: (i) application of the same methodology to other similar datasets, in order to be able to validate our methodological approach. As proposed in other studies, ability to replicate findings (i.e. successfully apply the same methodology or prediction models to other datasets), is one of the most challenging endeavours of predictive modeling [48, 10]. Additionally, (ii) further study of the effect of censored information, especially in comparison with dropping out the censored observations is also of our interest – we set as a constraint though, that the dataset should be censored itself and not apply artificial truncation, as we did in this study. (iii) Finally, regarding random-effects models, more in depth analysis of the final effect of the frailty object in the predictive efficiency as well as introduction of multi-level frailty objects is also a consideration. More complicated and hierarchical structures of group dependence (multi-level models, individual growth models or hierarchical linear models) are a methodological improvement of the single level frailty object. An example on the case we studied would be the incorporation of the hierarchical connection among the developer and the manager of the developer.

We look forward to contributing our work on these challenging problems relevant to quality assurance.

Appendix A

Results Tables

Interpretation of the results is discussed in depth in Chapter 4. Constructional explanation of the tables has already been provided in Chapter 4, however, for easier reference it is repeated here as well:

- First column represents the censoring proportion that we artificially achieved by truncation of the data.
- Second column indicates the sample size for each censoring proportion. The reason of the sample degradation as the censoring increases is given in Section 4.6.
- In the remaining ten columns (eight columns for Tables A.1 and A.6 since we did not calculate AIC for basic linear models because we cannot use them for model comparison, as discussed in Section 3.7.1) the values of each metric are presented – residual standard deviation, Kendall rank correlation coefficient, and AIC.
  - For standard deviation and Kendall rank tables, the third column provides results of a simple linear model that was built and presented as a baseline criterion. However, and since a linear model cannot incorporate censored information, the deliver time-stamp of the censored observations was set as the censoring cut-off point. For the AIC tables, this column is missing since AIC comparison between parametric and linear models is not valid (as discussed in Section 3.7.1).
  - Similarly, for standard deviation and Kendall rank tables only (AIC tables are missing this column as well), the fourth column presents results of a simple

linear model with the censored observations filtered out. The number of censored observations can be calculated by multiplying the censoring proportion with the sample size.

- Columns five, seven, nine, and eleven (three, five, seven, and nine for AIC) represent results of the fully parametric models: exponential, Weibull, lognormal, and loglogistic respectively.
- Consequently, the even columns from six to twelve (four to ten for AIC) represent the fully parametric models that completely drop/filter out the censored observations.

Table A.1: *Akaike information criterion. Fixed-effect models.*

Cens %	Sample Size	cens exp	drop exp	cens weib	drop weib	cens ln	drop ln	cens lglg	drop lglg
0	46296	702514	702514	636975	636975	638941	638941	638716	638716
1	40680	607726	606377	544782	543729	546001	545106	546053	545166
2	37524	555729	552426	495042	492663	495820	493727	495984	493943
3	33847	497109	489710	440197	435846	440389	436682	440501	436933
4	16986	241505	236796	209207	206534	209234	206966	209458	207270
5	16513	232484	227122	200646	197538	200571	197928	200798	198237
6	6994	96536	93425	82233	80620	82411	81032	82476	81161
7	6646	91031	87577	77280	75470	77395	75847	77463	75977
8	6207	84497	81211	71771	69931	71869	70267	71928	70387
9	5891	79487	76197	67603	65814	67707	66158	67759	66271
10	4545	61350	58171	51644	49933	51635	50145	51703	50269
11	4494	60061	56876	50429	48666	50414	48860	50484	48988
12	3995	53030	49480	44044	42202	43957	42346	44024	42474
13	3590	47366	43835	39058	37247	38943	37361	39010	37482
14	3391	44191	40758	36254	34433	36136	34521	36203	34641
15	3198	41161	37845	33562	31816	33441	31883	33506	32002
16	3102	39481	36147	32127	30386	32009	30458	32070	30574
17	2662	33694	30325	26933	25208	26783	25238	26846	25351
18	2547	31916	28673	25408	23714	25259	23728	25322	23838
19	2445	30290	27103	23983	22314	23833	22316	23894	22424
20	2257	27740	24634	21712	20077	21557	20064	21615	20165

Table shows AIC values of the fixed-effect models including all covariates.

Table A.2: *Standard deviation of the residuals for internal validation predictions. Fixed-effect models.*

Cens %	Sample Size	full linear model	drop linear model	cens exp	drop exp	cens weib	drop weib	cens ln	drop ln	cens lglg	drop lglg
0	46296	2084	2084	2087	2087	2095	2095	2106	2106	2112	2112
1	40680	2105	2076	2134	2079	2147	2087	2160	2097	2158	2101
2	37524	2140	2093	2199	2096	2212	2104	2226	2115	2224	2118
3	33847	2168	2029	2284	2032	2295	2040	2310	2051	2308	2055
4	16986	1932	1710	2420	1717	2431	1725	2445	1733	2444	1736
5	16513	1935	1706	2439	1713	2446	1722	2461	1730	2460	1730
6	6994	1826	1439	2722	1441	2742	1451	2766	1467	2767	1464
7	6646	1838	1422	2708	1425	2752	1435	2748	1449	2751	1447
8	6207	1861	1454	2785	1456	2839	1467	2820	1481	2822	1479
9	5891	1878	1480	2863	1484	2972	1493	2880	1509	2884	1507
10	4545	1957	1492	3321	1493	4706	1506	3519	1523	3448	1521
11	4494	1958	1496	3388	1496	5807	1511	3636	1527	3564	1525
12	3995	1984	1457	3571	1460	7137	1473	3706	1488	3674	1487
13	3590	2006	1387	4234	1389	1883213	1402	255371	1413	471061	1412
14	3391	2002	1352	4358	1354	993379	1367	160786	1379	279907	1378
15	3198	2004	1333	4203	1335	77371	1348	40442	1359	43035	1358
16	3102	2011	1304	4249	1306	64210	1319	33066	1330	35100	1330
17	2662	2048	1292	6392	1294	692005	1308	426261	1319	482294	1318
18	2547	2052	1313	6737	1316	745833	1329	461323	1341	530464	1340
19	2445	2064	1294	8552	1297	16064820	1312	1911965	1324	4275393	1324
20	2257	2074	1303	9199	1302	96112146	1321	6673515	1334	5714514	1333

Table shows the standard deviation of the residuals, while predicting values on the train set.

Table A.3: *Kendall rank correlation coefficient for internal validation predictions. Fixed-effect models.*

Cens %	Sample Size	full linear model	drop linear model	cens exp	drop exp	cens weib	drop weib	cens ln	drop ln	cens lglg	drop lglg
0	46296	0.197	0.197	0.157	0.157	0.161	0.161	0.126	0.126	0.128	0.128
1	40680	0.190	0.188	0.189	0.183	0.159	0.161	0.127	0.131	0.132	0.135
2	37524	0.188	0.186	0.190	0.182	0.160	0.158	0.130	0.127	0.134	0.131
3	33847	0.191	0.193	0.185	0.187	0.162	0.165	0.131	0.136	0.134	0.139
4	16986	0.223	0.211	0.199	0.197	0.162	0.165	0.137	0.134	0.140	0.137
5	16513	0.223	0.205	0.201	0.192	0.166	0.158	0.141	0.124	0.144	0.128
6	6994	0.274	0.227	0.228	0.220	0.191	0.206	0.168	0.158	0.155	0.175
7	6646	0.277	0.216	0.233	0.209	0.176	0.196	0.163	0.135	0.147	0.153
8	6207	0.278	0.218	0.241	0.211	0.910	0.197	0.174	0.133	0.166	0.151
9	5891	0.280	0.220	0.243	0.212	0.194	0.201	0.183	0.138	0.175	0.157
10	4545	0.304	0.249	0.247	0.247	0.184	0.224	0.136	0.171	0.142	0.181
11	4494	0.305	0.229	0.246	0.248	0.184	0.224	0.142	0.171	0.148	0.179
12	3995	0.315	0.212	0.261	0.223	0.200	0.198	0.179	0.122	0.183	0.129
13	3590	0.322	0.212	0.218	0.206	0.002	0.170	0.003	0.095	0.002	0.101
14	3391	0.317	0.219	0.232	0.213	0.004	0.180	0.005	0.099	0.003	0.105
15	3198	0.321	0.217	0.276	0.211	0.061	0.174	0.026	0.097	0.025	0.101
16	3102	0.322	0.221	0.276	0.214	0.071	0.179	0.031	0.102	0.030	0.107
17	2662	0.337	0.224	0.197	0.219	0.022	0.181	0.015	0.106	0.015	0.111
18	2547	0.347	0.228	0.219	0.220	0.029	0.184	0.017	0.110	0.016	0.115
19	2445	0.344	0.235	0.176	0.228	0.011	0.197	0.013	0.108	0.012	0.112
20	2257	0.362	0.235	0.201	0.237	-0.006	0.200	-0.003	0.111	0.007	0.116

Table shows estimation of the Kendall rank correlation coefficient. The test compares the ranking of the predicted values compared to the real values (on the train set).

Table A.4: *Standard deviation of the residuals for external validation predictions. Fixed-effect models.*

Cens %	Sample Size	full linear model	drop linear model	cens exp	drop exp	cens weib	drop weib	cens ln	drop ln	cens lglg	drop lglg
0	46296	1409	1409	1414	1414	1490	1490	1540	1540	1693	1693
1	40680	1410	1410	1421	1411	1437	1421	1442	1430	1488	1459
2	37524	1412	1412	1425	1412	1440	1426	1442	1432	1487	1471
3	33847	1414	1413	1429	1456	1445	1472	1455	1462	1531	1483
4	16986	1464	1456	1667	1452	1802	1473	1657	1464	1685	1490
5	16513	1468	1459	1715	1530	1916	1543	1719	1491	1740	1530
6	6994	1927	1492	3152	19092	153194	167523	44715	14445	115265	54939
7	6646	2064	1472	3686	6919	223888	55530	51092	8175	112509	23867
8	6207	2022	1475	3316	4966	144665	37078	62394	8607	130970	24110
9	5891	2026	1479	4869	25441	290582	40227	95227	7330	190777	18252
10	4545	2100	1481	7043	1816	204870	3187	94517	2657	145339	4230
11	4494	2508	1481	7744	1696	245761	2706	107483	2427	169778	3671
12	3995	2291	1474	8658	9579	354085	4864	162046	2721	176903	4615
13	3590	2158	1462	19146	15795	698547	4505	185526	1963	312457	2786
14	3391	2046	1463	27332	3683	1052286	1803	172214	1580	296561	1803
15	3198	1978	1463	9891	1473	763103	1496	325189	1645	777420	2095
16	3102	1946	1462	9833	1480	690576	1495	283588	1630	666758	2058
17	2662	1890	1460	39090	1507	86005838	1511	41044174	1539	43744867	1669
18	2547	2146	1463	45125	1534	166498057	1532	65338997	1544	77943281	1639
19	2445	2092	1464	57291	1519	1016885684	1486	101860794	1514	251401933	1617
20	2257	2127	1470	65873	1480	1746469745	1658	177497788	1708	424541895	2309

Table shows the standard deviation of the residuals, while predicting values on the test set.

Table A.5: *Kendall rank correlation coefficient for external validation predictions. Fixed-effect models.*

Cens %	Sample Size	full linear model	drop linear model	cens exp	drop exp	cens weib	drop weib	cens ln	drop ln	cens lglg	drop lglg
0	46296	0.192	0.192	0.191	0.191	0.098	0.098	0.048	0.048	0.035	0.035
1	40680	0.191	0.189	0.191	0.184	0.144	0.150	0.098	0.105	0.074	0.080
2	37524	0.186	0.185	0.187	0.180	0.142	0.140	0.101	0.100	0.075	0.071
3	33847	0.185	0.183	0.186	0.161	0.140	0.129	0.090	0.107	0.062	0.098
4	16986	0.154	0.139	0.142	0.160	0.112	0.127	0.092	0.106	0.086	0.094
5	16513	0.150	0.129	0.143	0.129	0.117	0.102	0.095	0.093	0.089	0.082
6	6994	0.149	0.073	0.101	-0.001	0.077	-0.003	0.074	-0.002	0.070	-0.003
7	6646	0.152	0.080	0.096	0.003	0.061	-0.003	0.068	-0.002	0.061	-0.003
8	6207	0.151	0.078	0.093	0.005	0.059	-0.003	0.065	-0.002	0.060	-0.003
9	5891	0.152	0.078	0.088	-0.001	0.054	-0.003	0.060	-0.002	0.057	-0.002
10	4545	0.146	0.085	0.069	0.037	0.027	0.005	0.015	0.001	0.020	0.000
11	4494	0.151	0.081	0.067	0.043	0.025	0.006	0.014	0.002	0.018	0.000
12	3995	0.147	0.091	0.059	0.043	0.011	0.006	0.008	0.002	0.011	0.000
13	3590	0.142	0.087	0.046	0.043	0.001	0.006	0.003	0.002	0.004	0.000
14	3391	0.142	0.084	0.044	0.043	0.000	0.006	0.002	0.002	0.003	0.000
15	3198	0.135	0.083	0.045	0.087	0.001	0.034	0.003	0.005	0.003	0.001
16	3102	0.135	0.087	0.048	0.005	0.002	0.086	0.004	0.036	0.005	0.002
17	2662	0.117	0.093	0.043	0.076	-0.002	0.037	-0.003	0.011	-0.003	0.007
18	2547	0.135	0.090	0.043	0.072	-0.001	0.035	-0.003	0.012	-0.003	0.009
19	2445	0.135	0.085	0.034	0.073	0.022	0.037	0.022	0.006	0.022	0.003
20	2257	0.116	0.074	0.036	0.065	0.021	0.010	0.022	0.001	0.022	-0.001

Table shows estimation of the Kendall rank correlation coefficient. The test compares the ranking of the predicted values compared to the real values (on the test set).

Table A.6: *Akaike information criterion. Mixed-effects / frailty models.*

Cens %	Sample Size	cens exp	drop exp	cens weib	drop weib	cens ln	drop ln	cens lglg	drop lglg
0	46296	682761	682761	631740	631740	631266	631266	630908	630908
1	40680	589133	587786	540083	539017	539230	538298	538103	538103
2	37524	537238	533953	490409	488136	489366	487356	487255	487255
3	33847	478699	471847	435629	431539	434227	430699	430630	430630
4	16986	229239	224928	206277	203774	205420	203266	203245	203245
5	16513	220259	215358	197702	194836	196792	194313	194309	194309
6	6994	88969	86911	80524	79157	80157	78904	80030	78822
7	6646	83716	81422	75630	74107	75250	73855	75132	73775
8	6207	77737	75566	70268	68740	69915	68503	69799	68428
9	5891	72993	70955	66177	64708	65888	64511	65771	64436
10	4545	59073	57820	50435	49095	50151	48877	50074	48820
11	4494	57032	55126	49218	47838	48926	47613	48852	47557
12	3995	49107	46937	42914	41447	42572	41207	42508	41168
13	3590	42649	40191	38030	36530	37705	36320	37656	36291
14	3391	39662	37217	35268	33741	34957	33533	34909	33499
15	3198	36691	34425	32589	31158	32298	30957	32248	30930
16	3102	35112	32849	31165	29743	30875	29549	30822	29524
17	2662	29499	29045	26019	24558	25729	24378	25686	24381
18	2547	30171	27536	24547	23093	24255	22900	24215	22917
19	2445	28734	26070	23143	21724	22858	21535	22821	21535
20	2257	25959	23290	20911	19509	20627	19339	20593	19352

Table shows AIC values of the mixed-effects models including all covariates.

Table A.7: *Standard deviation of the residuals for internal validation predictions. Mixed-effects / frailty models.*

Cens %	Sample Size	full linear model	drop linear model	cens exp	drop exp	cens weib	drop weib	cens ln	drop ln	cens lglg	drop lglg
0	46296	2026	2026	2025	2025	2048	2048	2077	2077	2070	2070
1	40680	2045	2016	2062	2016	2089	2041	2123	2071	2118	2081
2	37524	2075	2027	2122	2028	2149	2056	2185	2089	2180	2016
3	33847	2093	1956	2195	1958	2222	1990	2263	2023	2257	1699
4	16986	1802	1610	2263	1639	2330	1672	2384	1706	2374	1698
5	16513	1804	1608	2291	1635	2353	1671	2398	1705	2389	1428
6	6994	1611	1364	2601	1407	2563	1394	2658	1437	2634	1416
7	6646	1626	1345	2709	1390	2709	1379	2639	1424	2622	1448
8	6207	1651	1380	2712	1429	2762	1412	2698	1457	2673	1475
9	5891	1672	1406	2873	1461	2970	1438	2787	1484	2761	1484
10	4545	1708	1421	8292	1426	13681	1444	6492	1491	6904	1488
11	4494	1709	1424	8828	1450	15946	1448	7346	1495	7656	1450
12	3995	1746	1377	11195	1423	22575	1409	9608	1458	9460	
13	3590	1785	1293		1372		1337		1387		1341
14	3391	1789	1249		1335		1300		1350		
15	3198	1774	1230		1348		1288		1329		
16	3102	1770	1202		1319		1258		1300		
17	2662	1803	1189				1237		1288		
18	2547	1801	1207				1256		1308		
19	2445	1811	1189				1237		1291		
20	2257	1799	1184				1247		1301		

Table shows the standard deviation of the residuals, while predicting values on the train set.

Table A.8: *Kendall rank correlation coefficient for internal validation predictions. Mixed-effects / frailty models.*

Cens %	Sample Size	full linear model	drop linear model	cens exp	drop exp	cens weib	drop weib	cens ln	drop ln	cens lglg	drop lglg
0	46296	0.305	0.305	0.314	0.314	0.289	0.289	0.255	0.255	0.260	0.260
1	40680	0.303	0.305	0.307	0.304	0.268	0.267	0.228	0.221	0.234	0.228
2	37524	0.308	0.311	0.316	0.309	0.278	0.270	0.234	0.219	0.241	0.226
3	33847	0.323	0.328	0.325	0.325	0.284	0.280	0.237	0.228	0.244	0.236
4	16986	0.423	0.397	0.379	0.361	0.304	0.299	0.261	0.236	0.273	0.248
5	16513	0.424	0.393	0.370	0.358	0.281	0.293	0.237	0.224	0.248	0.236
6	6994	0.536	0.393	0.457	0.350	0.398	0.367	0.321	0.304	0.335	0.323
7	6646	0.534	0.394	0.451	0.346	0.351	0.364	0.315	0.285	0.333	0.307
8	6207	0.532	0.387	0.469	0.339	0.368	0.356	0.335	0.282	0.355	0.304
9	5891	0.528	0.387	0.457	0.336	0.351	0.362	0.310	0.286	0.339	0.310
10	4545	0.564	0.394	0.273	0.378	0.202	0.388	0.208	0.299	0.210	0.317
11	4494	0.565	0.396	0.280	0.379	0.209	0.388	0.218	0.302	0.221	0.320
12	3995	0.560	0.403	0.285	0.363	0.217	0.383	0.226	0.289	0.230	0.305
13	3590	0.549	0.427		0.351		0.380		0.259		
14	3391	0.542	0.448		0.368		0.378		0.275		
15	3198	0.556	0.449		0.350		0.348		0.282		
16	3102	0.565	0.453		0.353		0.358		0.288		
17	2662	0.572	0.459				0.398		0.299		
18	2547	0.579	0.463				0.403		0.308		
19	2445	0.579	0.467				0.417		0.334		
20	2257	0.600	0.488				0.417		0.340		

Table shows estimation of the Kendall rank correlation coefficient. The test compares the ranking of the predicted values compared to the real values (on the train set).

Table A.9: *Standard deviation of the residuals for external validation predictions. Mixed-effects / frailty models.*

Cens %	Sample Size	full linear model	drop linear model	cens exp	drop exp	cens weib	drop weib	cens ln	drop ln	cens lglg	drop lglg
0	46296	1418	1418	1447	1447	1406	1406	1406	1406	1406	1406
1	40680	1422	1422	1536	1469	1417	1414	1416	1412	1416	1412
2	37524	1433	1438	1554	1514	1443	1422	1426	1415	1428	1417
3	33847	1459	1446	1792	1561	1487	1428	1441	1420	1448	1423
4	16986	1589	1488	2254	1660	1727	1481	1545	1449	1606	1460
5	16513	1588	1490	2293	1664	1820	1476	1567	1444	1644	1454
6	6994	2146	1499	32724	20361	3407	1753	1873	1457	2018	1455
7	6646	2195	1485	19468	10665	3624	1587	1987	1450	2174	1448
8	6207	2101	1481	7678	3280	3608	1479	1996	1440	2182	1442
9	5891	2080	1486	10468	10084	4576	1491	2174	1439	2299	1441
10	4545	2463	1489	9649	1505	15074	1466	7904	1471	6869	1476
11	4494	2892	1491	9092	1578	18774	1465	9755	1468	8284	1474
12	3995	2539	1489	14362	1578	39338	1470	21802	1462	14181	1467
13	3590	2278	1480		1557		1468		1458		
14	3391	2152	1483		1629		1458		1465		1468
15	3198	2136	1476		1670		1496		1470		
16	3102	2123	1472								
17	2662	2021	1471								
18	2547	2248	1473								
19	2445	2191	1475								
20	2257	2266	1482								

Table shows the standard deviation of the residuals, while predicting values on the test set.

Table A.10: *Kendall rank correlation coefficient for external validation predictions. Fixed-effect models.*

Cens %	Sample Size	full linear model	drop linear model	cens exp	drop exp	cens weib	drop weib	cens ln	drop ln	cens lglg	drop lglg
0	46296	0.197	0.197	0.022	0.022	0.042	0.042	0.048	0.048	0.048	0.048
1	40680	0.191	0.186	0.016	-0.001	0.037	0.016	0.040	0.017	0.040	0.016
2	37524	0.175	0.166	-0.015	0.003	0.002	0.021	0.011	0.021	0.010	0.020
3	33847	0.159	0.165	0.004	-0.001	0.018	0.014	0.020	0.025	0.019	0.023
4	16986	0.113	0.122	0.009	-0.006	0.018	0.013	0.025	0.021	0.023	0.017
5	16513	0.110	0.117	0.011	-0.004	0.018	0.010	0.023	0.015	0.018	0.013
6	6994	0.119	0.077	0.001	-0.004	0.004	-0.003	0.005	0.005	-0.003	0.004
7	6646	0.130	0.080	0.001	-0.005	0.007	-0.002	0.007	0.001	0.003	-0.002
8	6207	0.130	0.081	-0.002	-0.006	0.005	0.002	0.002	0.011	-0.001	0.008
9	5891	0.134	0.079	0.000	-0.005	0.007	-0.010	0.005	-0.002	0.001	-0.008
10	4545	0.138	0.084	0.003	-0.011	0.014	-0.011	0.011	-0.005	0.012	-0.006
11	4494	0.146	0.079	0.005	-0.015	0.015	0.006	0.012	0.010	0.012	0.004
12	3995	0.143	0.082	0.006	-0.017	0.009	0.000	0.004	0.003	0.007	-0.001
13	3590	0.133	0.077	-0.006	-0.006	-0.004	0.005	-0.003	0.003	-0.002	-0.003
14	3391	0.132	0.073		-0.009		-0.001		0.002		-0.002
15	3198	0.124	0.079		-0.008		-0.001		-0.005		-0.010
16	3102	0.123	0.084								
17	2662	0.118	0.088								
18	2547	0.130	0.089								
19	2445	0.129	0.084								
20	2257	0.108	0.075								

Table shows estimation of the Kendall rank correlation coefficient. The test compares the ranking of the predicted values compared to the real values (on the test set).

Table A.11: *Standard deviation of the residuals for internal validation predictions on filtered dataset. Fixed-effect models.*

Cens %	Sample Size	linear model	linear model	cens exp	drop exp	cens weib	drop weib	cens ln	drop ln	cens lglg	drop lglg
0	45682	1239	1239	1241	1241	1246	1246	1260	1260		
1	39997	1198	1194	1214	1196	1218	1200	1230	1212	1228	1210
2	36429	1201	1190	1238	1193	1243	1197	1256	1209	1254	1207
3	17350	1103	1075	1151	1086	1159	1088	1164	1095	1162	1094
4	16466	1108	1053	1173	1063	1187	1065	1187	1072	1183	1071
5	6848	1043	1003	8821	1006	52678843	1013	477104	1018	1002824	1016
6	6461	1051	995	3666	998	6332778	1001	448751	1008	1067571	1007
7	6019	1062	1003	3107	1006	2481013	1009	296487	1017	516372	1016
8	5647	1075	1012	1661	1015	2231651	1018	63882	1026	74993	1025
9	4394	1104	997	1308	1002	1991	1004	1538	1014	1414	1012
10	3874	1108	932	1358	939	1520	941	1428	949	1395	948
11	3462	1103	884	2187	893	957527	891	167223	898	305640	898
12	3237	1071	858	1572	865	219761	866	64331	875	92739	874
13	2992	1066	839	1488	841	29110	846	21298	855	22452	854
14	2650	1061	818	1473	820	26482	826	2019	837	2069	836
15	2425	1054	797	4757	800	5413296	803	280261	809	318945	808
16	2198	1049	793	7057	794	9479944	799	1132558	805	2415363	804
17	1783	1031	725	10925	726	15873387	731	1088972	738	2244147	737
18	1688	1027	738	9531	740	16989704	745	1077234	753	2155302	752
19	1352	1044	782	36453	783	157688009	789	7999907	798	34900064	796
20	1186	1059	768	29022	768	386756744	775	11172981	782	7615209	781

Table shows the standard deviation of the residuals, while predicting values on the train set. The dataset has been filtered by removing extreme values of the response variable – approximately 1.1% of the initial dataset.

Table A.12: *Kendall rank correlation coefficient for internal validation predictions on filtered dataset. Fixed-effect models.*

Cens %	Sample Size	full linear model	drop linear model	cens exp	drop exp	cens weib	drop weib	cens ln	drop ln	cens lglg	drop lglg
0	45682	0.284	0.284	0.281	0.281	0.267	0.267	0.238	0.238	0.	0.
1	39997	0.272	0.267	0.275	0.262	0.261	0.248	0.231	0.216	0.238	0.223
2	36429	0.265	0.263	0.270	0.259	0.254	0.243	0.222	0.210	0.230	0.217
3	17350	0.255	0.256	0.255	0.233	0.227	0.212	0.189	0.174	0.198	0.182
4	16466	0.259	0.250	0.249	0.229	0.218	0.207	0.181	0.169	0.190	0.176
5	6848	0.212	0.203	0.127	0.192	0.120	0.149	0.119	0.132	0.121	0.144
6	6461	0.214	0.193	0.131	0.179	0.110	0.160	0.109	0.119	0.114	0.131
7	6019	0.212	0.193	0.126	0.182	0.095	0.169	0.097	0.122	0.103	0.134
8	5647	0.211	0.199	0.149	0.185	0.081	0.177	0.078	0.125	0.084	0.138
9	4394	0.224	0.209	0.226	0.191	0.120	0.183	0.102	0.107	0.119	0.126
10	3874	0.227	0.220	0.231	0.198	0.171	0.187	0.117	0.123	0.137	0.129
11	3462	0.235	0.219	0.078	0.188	0.010	0.182	0.011	0.129	0.010	0.135
12	3237	0.232	0.234	0.156	0.210	0.011	0.201	0.011	0.134	0.011	0.140
13	2992	0.241	0.241	0.219	0.232	0.017	0.211	0.013	0.142	0.012	0.151
14	2650	0.260	0.260	0.286	0.249	0.018	0.225	0.091	0.150	0.080	0.158
15	2425	0.263	0.230	0.073	0.218	0.039	0.199	0.039	0.153	0.039	0.162
16	2198	0.265	0.217	0.079	0.210	0.056	0.189	0.049	0.135	0.045	0.145
17	1783	0.266	0.217	0.079	0.211	0.064	0.195	0.060	0.145	0.053	0.156
18	1688	0.268	0.224	0.083	0.217	0.063	0.201	0.061	0.156	0.055	0.166
19	1352	0.265	0.223	0.102	0.220	0.083	0.199	0.061	0.153	0.072	0.167
20	1186	0.278	0.206	0.172	0.209	0.163	0.184	0.173	0.124	0.177	0.129

Table shows estimation of the Kendall rank correlation coefficient. The test compares the ranking of the predicted values compared to the real values (on the train set). The dataset has been filtered by removing extreme values of the response variable – approximately 1.1% of the initial dataset.

Table A.13: *Accuracy of fast/slow classification of the models.*

Cens %	Sample Size	ACC exp	weib	ln	lglg
0	46296	0.553	0.586	0.646	0.618
1	40680	0.532	0.564	0.600	0.603
2	37524	0.526	0.561	0.601	0.604
3	33847	0.522	0.559	0.604	0.607
4	16986	0.487	0.548	0.620	0.623
5	16513	0.488	0.549	0.621	0.623
6	6994	0.485	0.551	0.615	0.616
7	6646	0.486	0.551	0.616	0.616
8	6207	0.493	0.556	0.612	0.611
9	5891	0.499	0.558	0.609	0.609
10	4545	0.499	0.552	0.612	0.613
11	4494	0.498	0.549	0.610	0.613
12	3995	0.495	0.552	0.616	0.616
13	3590	0.492	0.552	0.620	0.620
14	3391	0.495	0.554	0.620	0.620
15	3198	0.492	0.550	0.622	0.617
16	3102	0.495	0.551	0.624	0.619
17	2662	0.489	0.552	0.633	0.632
18	2547	0.490	0.551	0.631	0.623
19	2445	0.490	0.542	0.631	0.629
20	2257	0.490	0.523	0.63	0.623

Accuracy of the models while classifying the duration as fast or slow. Time threshold is set as the median value of the duration of the entire dataset. The models used to calculate these values are corresponding to the results presented in Table A.2.

Appendix B

Analysis scripts

B.1 Main analysis script

```
##########################################
## Survival analysis on time-to-deliver ##
##########################################

## clear cache
rm(list = ls())

## libraries necessary
library(survival)
library(xtable)
library(flexsurv)
library(lattice)
library(actuar)
library(rms)
library(plyr)
library(lme4)

setwd("/Users/sokratis/Documents/xx")

## updated dataset containing records from 4 releases
## aggreagated at the commit level
_data <-
  read.csv("./data/four_releases_aggregated.anonymized.csv",
```

```
              na.strings = "")

source("./scripts/data_preprocess.R")
source("./scripts/survreg_curves_helper.R")
source("./scripts/survival_analysis_helper.R")
source("./scripts/validation_helper.R")

_data <- data_prepro(_data)
# _data <- _data[_data$y < 8760, ]

## Create the dataset 123 for the 3 first releases - train set
_data123 <-
  _data[_data$major_release_number %in%
    c("v.x", "v.x+1", "v.x+2"),]
_data123$major_release_number <-
  droplevels(_data123$major_release_number)
_data123$developer_country <-
  droplevels(_data123$developer_country)
_data123$symptom <- droplevels(_data123$symptom)
#_data123$developer <- droplevels(_data123$developer)

## Create the dataset 4 for the last release - as the test set
_data4 <- _data[_data$major_release_number %in% "v.x+3",]
_data4$major_release_number <-
  droplevels(_data4$major_release_number)
_data4$developer_country <- droplevels(_data4$developer_country)
_data4$symptom <- droplevels(_data4$symptom)

# all variables for fixed-effect models
covs_all <-
  c(
    "is_defect",
    "pre_post_ga",
    "distinct_component_count",
    "distinct_function_count",
    "developer_country",
    "symptom"
  )

```

```
_data4_full <- _data4
_data4_fr <- _data4[, c(covs_all, "developer")]
_data4 <- _data4[, covs_all]

###################################
## full analysis function call ##
###################################
full_analysis <- surv_an(data_surv = _data123)

###############
## CENSORING ##
###############

# censoring points in a sequence
cens_seq <- seq(0.01, 0.2, by = 0.01)

cens_analysis <- list()
drop_analysis <- list()

j <- ratio_cens <- size <- 0
for (i in unique(_data123$deliver_end_date)) {
  j <- j + 1
  valid <- sum(_data123$submit_date < i)
  cens <-
    sum(_data123$deliver_end_date > i &
          _data123$submit_date < i)
  ratio_cens[j] <- cens / valid
  size[j] <- valid
}

## Loop for calling the surv_an function for multiple
## censored datasets based on cens_seq
z <- 1
for (i_cens in cens_seq) {
  tmp_data <- _data123

  # get the position that has the a ratio of censoring equal to
  # what we are looking for. However, if there is a position
  # that the censoring is greater and at the same time
```

```
# the observations are more - take this position instead.
xx <-
  max(which(abs(ratio_cens - i_cens) == min(abs(
    ratio_cens - i_cens
  ))))
pos <- xx
for (k in xx:j) {
  if (size[k] > size[xx] & i_cens < ratio_cens[k]) {
    pos <- k
  }
}

data_cens <-
  c(ratio_cens[pos], size[pos],
    unique(tmp_data$deliver_end_date)[pos])

tmp_data$cens <- 0
tmp_data$cens[tmp_data$deliver_end_date > data_cens[3]] <- 1

## Keep only the observations that have
## a submit_date earlier than the censor point
_sub_data <-
  subset(tmp_data, tmp_data$submit_date < data_cens[3])

# drop symtpom levels
_sub_data$symptom <- droplevels(_sub_data$symptom)

# for developer_country values that have less than 2
# drop their records, because they can't predict afterwards
help_var <- table(_sub_data$developer_country)
_sub_data <-
  _sub_data[_sub_data$developer_country %in%
            names(help_var)[help_var > 2],]

# drop developer_country levels
_sub_data$developer_country <-
  droplevels(_sub_data$developer_country)

## Save a copy of the data before changing the y var
```

```
## to calculate residuals. The function "residuals"
## should not be used when censored information exists
backup_data <- _sub_data

## set the y of censored records to:
## "cens_point - submit_date"
_sub_data$y[_sub_data$cens == 1] <- (data_cens[3] -
             _sub_data$submit_date[_sub_data$cens == 1]) / 3600

## For drop analysis.
####################
# Remove the cesnored entries - keep the not censored
_sub_data_2 <- _sub_data[_sub_data$cens == 0,]

_sub_data_2$symptom <- droplevels(_sub_data_2$symptom)

# drop values that have less than 2 because they can't predict
help_var <- table(_sub_data_2$developer_country)
_sub_data_2 <-
  _sub_data_2[_sub_data_2$developer_country %in%
              names(help_var)[help_var > 2],]

_sub_data_2$developer_country <-
  droplevels(_sub_data_2$developer_country)

# Call the surv_an function for the censored data
cens_analysis[[z]] <-
  surv_an(data_surv = _sub_data,
          name_data = paste("cens_", i_cens, sep = ""))
cens_analysis[[z]]$rows <- nrow(_sub_data)
cens_analysis[[z]]$prop <- data_cens[1]
cens_analysis[[z]]$backup_data <- backup_data
print(paste(
  "done cens analysis of ",
  i_cens,
  " with ",
  cens_analysis[[z]]$rows,
  " records."
))
```

```

  # call the surv_an function for the filtered/dropped data
  drop_analysis[[z]] <-
    surv_an(data_surv = _sub_data_2,
            name_data = paste("drop_", i_cens, sep = ""))
  drop_analysis[[z]]$rows <- nrow(_sub_data_2)
  drop_analysis[[z]]$prop <- data_cens[1]
  drop_analysis[[z]]$backup_data <- _sub_data_2
  print(paste(
    "done drop_analysis of ",
    i_cens,
    " with ",
    drop_analysis[[z]]$rows,
    " records."
  ))

  z <- z + 1
}
##########################################################
```

B.2 Data preprocess

The content of `./scripts/data_preprocess.R` which is called in the main script, is given below.

```
#########################
## data preprocessing ##
#########################

data_prepro <- function(_data) {
  ## Convert dates to Unix times
  _data$submit_date <-
    as.numeric(as.POSIXct(_data$submit_date,
                    format = "%Y-%m-%d-%H.%M.%S"))
  _data$deliver_end_date <-
    as.numeric(as.POSIXct(
      _data$deliver_end_date,
      format = "%Y-%m-%d-%H.%M.%S",
      na.rm = T
    ))

  ## Remove some entries that due to timezone
  ## errors end up having (submit_date > deliver_end_date)
  _data <- _data[_data$deliver_end_date > _data$submit_date,]

  ## Remove 3 entries that do not have a symptom tag
  _data <- _data[(is.na(_data$symptom) != 1),]

  ## Aggregate 2 symptom tags that overlap
  _data$symptom[_data$symptom == 'Test_failed'] <-
        "test_failed"
  _data$symptom[_data$symptom == 'Build_failed'] <-
        "build_failed"
  _data$symptom <- droplevels(_data$symptom)

  ## Create the is_defect variable that we had in the previous dataset
  _data$is_defect <- 0
  # table(_data$classification)
  _data$is_defect[_data$classification %in% c(
    "Code_defect",
```

```
    "Code Defect",
    "Code Defect",
    "Test Case Defect",
    "Documentation Defect"
  )] <- 1

  _data <-  _data[order(_data$deliver_end_date),]

  ## Remove first character from all
  ## developer names (to have numeric values only)
  _data$developer <- substring(_data$developer, 2)
  _data$developer <- as.numeric(_data$developer)

  ## Create our "y" variable
  _data$y <- (_data$deliver_end_date - _data$submit_date) / 3600

  ## Create a binary variable representing fast/slow fix
  _data$fs <- 'f'
  _data$fs[_data$y > median(_data$y)] <- 's'

  #_data <- na.omit(_data)
  _data <-  _data[(is.na(_data$y) != 1),]
  _data$cens <- 0

  _data
}
```

B.3 Survival analysis helper script

The content of `./scripts/survival_analysis_helper.R` which is called in the main script, is given below.

```r
########################################
## Complete Survival Analysis function ##
########################################

surv_an <-
  function(data_surv,
           covs = covs_all,
           name_data = "full") {
    covs <- paste(covs, collapse = " + ")
    covs <- paste("Surv(y, cens == 0) ~ ", covs)
    covs_frailty_coxph <-
      paste(c(covs, "frailty(developer, sparse = F)"),
            collapse = " + ")
    covs_frailty_survreg <-
      paste(c(covs, "frailty.gaussian(developer, sparse = F)"),
            collapse = " + ")

    #####################################
    ## Kaplan Meier & Cumulative hazard ##
    ##      with confidence interval     ##
    #####################################

    surv_ci <-
      survfit(Surv(y, cens == 0) ~ 1, conf.int = TRUE,
              data = data_surv)
    ## second survfit where we keep only the non-censored data
    surv_ci_2 <-
      survfit(Surv(y, cens == 0) ~ 1, conf.int = TRUE,
              data = data_surv[data_surv$cens == 0,])

    ## Plot a Kaplan Meier graph and a cumulative hazard graph.
    ## Noticeable is the quick fix of a big proportion of the
    ## data but some observations are dragging the graph
    ## to the right.
    pdf(paste(paste("graphs/", name_data),
```

```
        "km_ch.pdf", sep = "_"))
    par(mfrow = c(1, 2), pty = 's')
    plot(surv_ci,
         xlab = "time-to-deliver",
         ylab = "Survival",
         main = "Kaplan-Meier survival graph")
    grid (NULL, NULL, lty = "dotted", col = "lightgray")
    plot(
      surv_ci,
      fun = "cumhaz",
      xlab = "time-to-deliver",
      ylab = "Cumulative hazard rate",
      main = "Cumulative hazard graph"
    )
    grid (NULL, NULL, lty = "dotted", col = "lightgray")
    dev.off()

    pdf(paste(paste("graphs/", name_data),
        "km_ch_log.pdf", sep = "_"))
    par(mfrow = c(1, 2), pty = 's')
    plot(
      surv_ci,
      xlab = "time-to-deliver (log)",
      ylab = "Survival",
      log = 'x',
      main = "Kaplan-Meier survival graph"
    )
    grid (NULL, NULL, lty = "dotted", col = "lightgray")
    plot(
      surv_ci,
      fun = "cumhaz",
      xlab = "time-to-deliver (log)",
      log = 'x',
      ylab = "Cumulative hazard rate",
      main = "Cumulative hazard graph"
    )
    grid (NULL, NULL, lty = "dotted", col = "lightgray")
    dev.off()
    ## As it is shown from the graphs 1 and 2 which
```

```
    ## represent a Kaplan-Meier curve and a Cumulative
    ## hazard graph respectively, most of the reports
    ## are fixed very fast, while some of them take
    ## longer time to get resolved.

    ###############################################
    # Plot a basic model for two classes of
    ## a single attribute (is_defect).
    pdf(paste(name_data, "km_ch_cov.pdf", sep = "_"))
    par(mfrow = c(2, 2))
    surv_1_1 <-
      survfit(Surv(y, cens == 0) ~ is_defect,
              conf.int = FALSE,
              data = data_surv)
    plot(
      surv_1_1,
      xlab = "time-to-deliver",
      ylab = "Survival",
      main = "Kaplan-Meier survival graph",
      col = c(1, 2),
      xlim = c(0, 2000)
    )
    grid (NULL, NULL, lty = "dotted", col = "lightgray")
    legend("topright",
           c("defect", "non-defect"),
           col = c(2, 1),
           lty = 1)
    plot(
      surv_1_1,
      fun = "cumhaz",
      xlab = "time-to-deliver",
      ylab = "Cumulative hazard rate",
      main = "Cumulative hazard graph",
      col = c(1, 2),
      xlim = c(0, 2000),
      ylim = c(0, 3)
    )
    grid (NULL, NULL, lty = "dotted", col = "lightgray")
    legend("bottomright",
```

```
            c("defect", "non-defect"),
            col = c(2, 1),
            lty = 1)

    # Plot a basic model for two classes of
    ## a single attribute (pre_post_ga).
    surv_1_2 <-
      survfit(Surv(y, cens == 0) ~ pre_post_ga,
               conf.int = FALSE,
               data = data_surv)
    plot(
      surv_1_2,
      xlab = "time-to-deliver",
      ylab = "Survival",
      main = "Kaplan-Meier survival graph",
      col = c(1, 2),
      xlim = c(0, 2000)
    )
    grid (NULL, NULL, lty = "dotted", col = "lightgray")
    legend("topright",
            c("pre_ga", "post_ga"),
            col = c(2, 1),
            lty = 1)
    plot(
      surv_1_2,
      fun = "cumhaz",
      xlab = "time-to-deliver",
      ylab = "Cumulative hazard rate",
      main = "Cumulative hazard graph",
      col = c(1, 2)
    )
    grid (NULL, NULL, lty = "dotted", col = "lightgray")
    legend("bottomright",
            c("pre_ga", "post_ga"),
            col = c(2, 1),
            lty = 1)
    dev.off()

    ####################################
```

```
  ## Cox proportional hazard models ##
  ######################################

  coxph_full <- coxph(as.formula(covs), data = data_surv)
  coxph_fr_full <-
    coxph(as.formula(covs_frailty_coxph), data = data_surv)
  coxph_step <- step(coxph_full, trace = 0)

  # cox_zph_test <- cox.zph(coxph_full)$table[, 3]
  # test of proportionality assumption

  print("Cox PH done...")

  ###############################
  ## Fully parametric models ##
  ###############################
  # We utilize fully parametric models, because
  ## they model the time-to-deliver, which is
  # more appropriate than the hazard estimated
  ## by the Cox models above. However, their
  ## disadvantage is that we have to assume
  ## a distribution for our empirical data
  ## (or their residuals) for efficient estimation.
  ## The goodness of fit can be calculated by the
  ## AIC and optically from Survival graphs.

  dists <- c("Exponential", "Weibull",
             "Lognormal", "Loglogistic")

  # exponential
  #############
  exp1 <-
    try(survreg(Surv(y, cens == 0) ~ 1,
        data = data_surv, dist = "exponential"))
  exp_full <-
    survreg(as.formula(covs),
            data = data_surv, dist = "exponential")
  exp_fr_full <-
    survreg(
```

```
      as.formula(covs_frailty_survreg),
      data = data_surv,
      dist = "exponential",
      x = T
    )
  exp_step <- step(exp_full, trace = 0)

  S_exp <- function(x) {
    1 - pexp(x, 1 / exp(exp1$coefficients[1]))
  }
  print("Exponential done...")

  # weibull
  #########
  weib1 <-
    flexsurvreg(Surv(y, cens == 0) ~ 1,
                data = data_surv, dist = "weibull")
  weib_full <-
    survreg(as.formula(covs),
            data = data_surv, dist = "weibull")
  weib_fr_full <-
    survreg(
      as.formula(covs_frailty_survreg),
      data = data_surv,
      dist = "weibull",
      x = T
    )
  weib_step <- step(weib_full, trace = 0)

  S_weib <- function(x) {
    1 - pweibull(x, exp(weib1$coefficients[1]),
                 exp(weib1$coefficients[2]))
  }
  print("Weibull done...")

  # lognormal
  ###########
  ln1 <-
    flexsurvreg(Surv(y, cens == 0) ~ 1,
```

```
                data = data_surv, dist = "lognormal")
  ln_full <-
    survreg(as.formula(covs),
            data = data_surv, dist = "lognormal")
  ln_fr_full <-
    survreg(
      as.formula(covs_frailty_survreg),
      data = data_surv,
      dist = "lognormal",
      x = T
    )
  ln_step <- step(ln_full, trace = 0)

  S_ln <- function(x) {
    1 - plnorm(x, ln1$coefficients[1],
               exp(ln1$coefficients[2]))
  }
  print("Lognormal done...")

  # loglogistic
  #############
  lglg1 <-
    flexsurvreg(Surv(y, cens == 0) ~ 1,
                data = data_surv, dist = "llogis")
  lglg_full <-
    survreg(as.formula(covs),
            data = data_surv, dist = "loglogistic")
  lglg_fr_full <-
    survreg(
      as.formula(covs_frailty_survreg),
      data = data_surv,
      dist = "loglogistic",
      x = T
    )
  lglg_step <- step(lglg_full, trace = 0)

  S_ll <- function(x) {
    1 - pllogis(x, exp(lglg1$coefficients[1]),
                1 / exp(lglg1$coefficients[2]))
```

```
    }
    print("Loglogistic done...")

    ##########################
    ## then plot all together
    pdf(paste(paste("graphs/", name_data),
        "surv_all.pdf", sep = "_"))
    plot(
      surv_ci,
      conf = "none",
      xlab = "time-to-deliver",
      ylab = "Survival",
      lty = 1
    )
    grid (NULL, NULL, lty = "dotted", col = "lightgray")
    lines(0:max(data_surv$y),
          S_exp(0:max(data_surv$y)),
          col = 2,
          lty = 6)
    lines(0:max(data_surv$y),
          S_weib(0:max(data_surv$y)),
          col = 3,
          lty = 2)
    lines(0:max(data_surv$y),
          S_ln(0:max(data_surv$y)),
          col = 4,
          lty = 4)
    lines(0:max(data_surv$y),
          S_ll(0:max(data_surv$y)),
          col = 5,
          lty = 5)
    legend(
      x = "topright",
      legend = c("Kaplan-Meier", dists),
      lwd = 2,
      bty = "n",
      col = c("black", 2, 3, 4, 5),
      lty = c(1, 6, 2, 4, 5)
    )
```

```
    dev.off()

    pdf(paste(paste("graphs/", name_data),
        "surv_all_ch.pdf", sep = "_"))
    plot(
      surv_ci,
      fun = "cumhaz",
      conf = "none",
      xlab = "time-to-deliver",
      ylab = "Cumulative Hazard",
      ylim = c(0, 20),
      lty = 1
    )
    grid (NULL, NULL, lty = "dotted", col = "lightgray")
    lines(0:max(data_surv$y),
          -log(S_exp(0:max(data_surv$y))),
          col = 2,
          lty = 6)
    lines(0:max(data_surv$y),
          -log(S_weib(0:max(data_surv$y))),
          col = 3,
          lty = 2)
    lines(0:max(data_surv$y),
          -log(S_ln(0:max(data_surv$y))),
          col = 4,
          lty = 4)
    lines(0:max(data_surv$y),
          -log(S_ll(0:max(data_surv$y))),
          col = 5,
          lty = 5)
    legend(
      x = "bottomright",
      legend = c("Cumulative hazard", dists),
      lwd = 2,
      bty = "n",
      col = c("black", 2, 3, 4, 5),
      lty = c(1, 6, 2, 4, 5)
    )
    dev.off()
```

```
    aic_1 <- c(try(AIC(exp1))
               , AIC(weib1), AIC(ln1), AIC(lglg1))
    aic_full <-
      c(try(AIC(exp_full))
        , AIC(weib_full), AIC(ln_full), AIC(lglg_full))
    aic_fr_full <-
      c(AIC(exp_fr_full) ,
        AIC(weib_fr_full),
        AIC(ln_fr_full),
        AIC(lglg_fr_full))

    names(aic_1) <- dists
    names(aic_full) <- dists
    names(aic_step) <- dists

    ## Return a list with all the results.
    list(
      surv_ci = surv_ci,
      surv_ci_2 = surv_ci_2,
      coxph_full = coxph_full,
      coxph_step = coxph_step,
      coxph_fr_full = coxph_fr_full,
      exp1 = exp1,
      exp_full = exp_full,
      exp_fr_full = exp_fr_full,
      weib1 = weib1,
      weib_full = weib_full,
      weib_fr_full = weib_fr_full,
      ln1 = ln1,
      ln_full = ln_full,
      ln_fr_full = ln_fr_full,
      lglg1 = lglg1,
      lglg_full = lglg_full,
      lglg_fr_full = lglg_fr_full,
      aic_1 = aic_1,
      aic_full = aic_full
    )
  }
```

```
## end of surv_an function ##
#############################
```

B.4 Survival analysis helper script

The content of `./scripts/validation_helper.R` which is called in the main script, is given below.

```
##########################
## Validation function ##
##########################

validate_survival <-
  function(val_fit,
           fit_data,
           val_data,
           val_data_fr,
           full_data = _data) {

    _data4_full <-
      full_data[full_data$major_release_number == "v.x+3" &
                  full_data$developer_country != "y13",]

    ##############
    ## Visual 1 ##
    ##############

    fit_val_4 <- survfit(Surv(y, cens == 0) ~ 1, data = _data4_full)
    fit_val_123 <- survfit(Surv(y, cens == 0) ~ 1, data = fit_data)

    # Plot a Kaplan-Meier for the secondary dataset
    # and a Kaplan-Meier for the primary data
    pdf("visual_surv_curves.pdf")
    plot(
      fit_val_4,
      xlab = "time-to-deliver",
      ylab = "Survival",
      col = 6,
      main = "Kaplan-Meier graphs"
    )
    lines(fit_val_123, col = 5)
    legend(
      x = "topright",
```

```
    legend = c("Primary data", "Secondary data"),
    lwd = 2,
    bty = "n",
    col = c(5, 6)
  )
  dev.off()

  ## We see some differences on the empirical data.
  ## In the new (validation/test) data the survival
  ## curve is more steep in the beginning and
  ## extends further to the right.

  ##############
  ## Visual 2 ##
  ##############
  weib1 <-
    flexsurvreg(Surv(y, cens == 0) ~ 1,
                data = fit_data, dist = "weibull")

  ## Plot a Kaplan-Meier for the secondary data (4th release)
  ## and compare with the best fitted model from the
  ## train data (first 3 releases)
  pdf("visual_fit.pdf")
  plot(
    weib1,
    xlab = "time-to-deliver",
    ylab = "Survival",
    main = "Kaplan-Meier vs Fitted model",
    col = 5
  )
  lines(fit_val_4, col = 6)
  legend(
    x = "topright",
    legend = c(
      "Weibull model (primary data)",
      "Kaplan-Meier (secondary data)"
    ),
    lwd = 2,
    bty = "n",
```

```
      col = c(5, 6)
    )
    dev.off()

    ## We see that the fit is not as good as expected
    ## which is also conceived by visually inspecting
    ## the previous plot (2 x Kaplan-Meier curves).
    ## However, this is just the empirical data, that
    ## are expected to have some changes over time.
    ## We will be assessing the best fitted model
    ## from the train set, on the new test set later on.
}

##########################################################################
## Save coef_1 and coef_2 of the empty models to see how the values
## change with proportion of censoring
##########################################################################

# censored
cens_coef1 <- full_analysis$weib1$coefficients[1]
cens_coef2 <- full_analysis$weib1$coefficients[2]

for (i in 1:length(cens_analysis)) {
  cens_coef1 <- c(cens_coef1, cens_analysis[[i]]$exp1$coefficients[1])
  cens_coef2 <-
    c(cens_coef2, cens_analysis[[i]]$exp1$coefficients[2])
}

x <- 0:20
par(mar = c(5, 4, 4, 5) + 0.1)
plot(
  x,
  cens_coef1,
  type = "l",
  col = "red",
  xlab = "Censoring Proportion (%)",
  ylab = "coefficient 1",
  main = "exponential",
  sub = "cens_analysis"
```

```
)
par(new = TRUE)
plot(
  x,
  cens_coef2,
  type = "l",
  col = "blue",
  xaxt = "n",
  yaxt = "n",
  xlab = "",
  ylab = ""
)
axis(4)
mtext("coefficient 2", side = 4, line = 3)
legend(
  "top",
  col = c("red", "blue"),
  lty = 1,
  legend = c("coef 1", "coef 2")
)
legend("top",
       col = "red",
       lty = 1,
       legend = "coef 1")

#dropped
drop_coef1 <- full_analysis$exp1$coefficients[1]
drop_coef2 <- full_analysis$exp1$coefficients[2]

for (i in 1:length(drop_analysis)) {
  drop_coef1 <-
    c(drop_coef1, drop_analysis[[i]]$weib1$coefficients[[1]])
  drop_coef2 <-
    c(drop_coef2, drop_analysis[[i]]$weib1$coefficients[[2]])
}

x <- 0:20
par(mar = c(5, 4, 4, 5) + 0.1)
plot(
```

```
  x,
  drop_coef1,
  type = "l",
  col = "red",
  xlab = "Censoring Proportion (%)",
  ylab = "coefficient 1",
  main = "exponential",
  sub = "drop analysis"
)
par(new = TRUE)
plot(
  x,
  drop_coef2,
  type = "l",
  col = "blue",
  xaxt = "n",
  yaxt = "n",
  xlab = "",
  ylab = ""
)
axis(4)
mtext("coefficient 2", side = 4, line = 3)
legend(
  "top",
  col = c("red", "blue"),
  lty = 1,
  legend = c("coef 1", "coef 2")
)
legend("top",
       col = "red",
       lty = 1,
       legend = "coef 1")
```

B.5 External validation

The following script is called independently and provides metrics for external validation.

```
##########################
## External validation ##
##########################

### for fixed-effect
### full_analysis

## Calculate the residuals for the
## frailty models that drop some levels themselves.
## They probably drop the levels that don't
## have enough values.
## full_analysis

pred_exp <- predict(full_analysis$exp_full,
                    newdata = _data4)
pred_weib <- predict(full_analysis$weib_full,
                     newdata = _data4)
pred_ln <- predict(full_analysis$ln_full,
                   newdata = _data4)
pred_lglg <- predict(full_analysis$lglg_full,
                     newdata = _data4)

res_exp <- pred_exp - _data4_full$y
res_weib <- pred_weib - _data4_full$y
res_ln <- pred_ln - _data4_full$y
res_lglg <- pred_lglg - _data4_full$y

rcc_exp <-
  cor.test(pred_exp, _data4_full$y,
           type = 'kendal')$estimate
rcc_weib <-
  cor.test(pred_weib, _data4_full$y,
           type = 'kendal')$estimate
rcc_ln <-
  cor.test(pred_ln, _data4_full$y,
           type = 'kendal')$estimate
```

```
rcc_lglg <-
  cor.test(pred_lglg, _data4_full$y,
           type = 'kendal')$estimate

acc_exp <- (sum(pred_exp < 140 &
    _data4_full$y < 140) + sum(pred_exp >= 140 &
    _data4_full$y >= 140)
    ) / length(_data4_full$y)
acc_weib <- (sum(pred_weib < 140 &
    _data4_full$y < 140) + sum(pred_weib >= 140 &
    _data4_full$y >= 140)
    ) / length(_data4_full$y)
acc_ln <- (sum(pred_ln < 140 &
    _data4_full$y < 140) + sum(pred_ln >= 140 &
    _data4_full$y >= 140)
    ) / length(_data4_full$y)
acc_lglg <- (sum(pred_lglg < 140 &
    _data4_full$y < 140) + sum(pred_lglg >= 140 &
    _data4_full$y >= 140)
    ) / length(_data4_full$y)

cat(
  paste(
    "full_analysis / fixed / external",
    "\n",
    "\t",
    full_analysis$exp_full$dist,
    "\tSD: ",
    round(sd(res_exp)),
    "RCC: ",
    round(rcc_exp, 3),
    "ACC: ",
    round(acc_exp, 3),
    "\n",
    "\t",
    full_analysis$weib_full$dist,
    "\tSD: ",
    round(sd(res_weib)),
    "RCC: ",
```

```
      round(rcc_weib, 3),
      "ACC: ",
      round(acc_weib, 3),
      "\n",
      "\t",
      full_analysis$ln_full$dist,
      "\tSD: ",
      round(sd(res_ln)),
      "RCC: ",
      round(rcc_ln, 3),
      "ACC: ",
      round(acc_ln, 3),
      "\n",
      "\t",
      full_analysis$lglg_full$dist,
      "\tSD: ",
      round(sd(res_lglg)),
      "RCC: ",
      round(rcc_lglg, 3),
      "ACC: ",
      round(acc_lglg, 3)
  )
)

####################
### for mixed-effects
### full_analysis

## Split the data in 2 parts
## 1 excludes the developers that the sparse
## matrix is ignoring
## 1 with rest of them (for this one we will
## ignore the developers coefficient)

exclude <- c(dev1, dev4, dev6, ...)

_data4_fr_part1 <-
  _data4_fr[!(_data4_fr$developer %in% exclude),]
_data4_full_part1 <-
```

```
  _data4_full[!(_data4_full$developer %in% exclude),]
_data4_fr_part2 <-
  _data4_fr[_data4_fr$developer %in% exclude,]
_data4_full_part2 <-
  _data4_full[_data4_full$developer %in% exclude,]

# For part 1, predict will work fine.
# test: pred_part_1_exp <-
#         predict(full_analysis$exp_fr_full, newdata = _data4_fr)
pred_part_1_exp <-
  predict(full_analysis$exp_fr_full, newdata = _data4_fr_part1)
pred_part_1_weib <-
  predict(full_analysis$weib_fr_full, newdata = _data4_fr_part1)
pred_part_1_ln <-
  predict(full_analysis$ln_fr_full, newdata = _data4_fr_part1)
pred_part_1_lglg <-
  predict(full_analysis$lglg_fr_full, newdata = _data4_fr_part1)

# For part 2 we have to ignore the
# coefficients for the developers
x_colnames <- colnames(full_analysis$exp_fr_full$x)
x_colnames_2 <-
    x_colnames[substring(x_colnames, 1, 5) != "frail"]

tmp_survreg <-
  survreg(
    Surv(y, rep(1, nrow(
      _data4_full_part2
    ))) ~ is_defect + pre_post_ga +
          distinct_component_count +
          distinct_function_count +
          developer_country + symptom,
    data = _data4_full_part2,
    x = T
  )

# the dataset is smaller now,
# so just keep the columns neccessary
x_colnames_2 <- colnames(tmp_survreg$x)
```

```
x_colnames_2 <-
  names(full_analysis$exp_fr_full$coefficients) %in%
          x_colnames_2

pred_part_2_exp <-
  tmp_survreg$x %*%
     full_analysis$exp_fr_full$coefficients[x_colnames_2]
pred_part_2_weib <-
  tmp_survreg$x %*%
     full_analysis$weib_fr_full$coefficients[x_colnames_2]
pred_part_2_ln <-
  tmp_survreg$x %*%
     full_analysis$ln_fr_full$coefficients[x_colnames_2]
pred_part_2_lglg <-
  tmp_survreg$x %*%
     full_analysis$lglg_fr_full$coefficients[x_colnames_2]

# keep the predicted values as well as the real values
# and then get the residuals
pred_y_exp <-
  cbind(c(pred_part_1_exp, exp(pred_part_2_exp)),
        c(_data4_full_part1$y, _data4_full_part2$y))
pred_y_weib <-
  cbind(c(pred_part_1_weib, exp(pred_part_2_weib)),
        c(_data4_full_part1$y, _data4_full_part2$y))
pred_y_ln <-
  cbind(c(pred_part_1_ln, exp(pred_part_2_ln)),
        c(_data4_full_part1$y, _data4_full_part2$y))
pred_y_lglg <-
  cbind(c(pred_part_1_lglg, exp(pred_part_2_lglg)),
        c(_data4_full_part1$y, _data4_full_part2$y))

res_exp <- pred_y_exp[, 1] - pred_y_exp[, 2]
res_weib <- pred_y_weib[, 1] - pred_y_weib[, 2]
res_ln <- pred_y_ln[, 1] - pred_y_ln[, 2]
res_lglg <- pred_y_lglg[, 1] - pred_y_lglg[, 2]

acc_exp <-
  (sum(pred_y_exp[, 1] < 140 &
```

```
        pred_y_exp[, 2] < 140) + sum(pred_y_exp[, 1] >= 140 &
        pred_y_exp[, 2] >= 140)) / length(pred_y_exp[, 1])
acc_weib <-
  (sum(pred_y_weib[, 1] < 140 &
        pred_y_weib[, 2] < 140) + sum(pred_y_weib[, 1] >= 140 &
        pred_y_weib[, 2] >= 140)
  ) / length(pred_y_weib[, 1])
acc_ln <-
  (sum(pred_y_ln[, 1] < 140 &
        pred_y_ln[, 2] < 140) + sum(pred_y_ln[, 1] >= 140 &
        pred_y_ln[, 2] >= 140)) / length(pred_y_ln[, 1])
acc_lglg <-
  (sum(pred_y_lglg[, 1] < 140 &
        pred_y_lglg[, 2] < 140) + sum(pred_y_lglg[, 1] >= 140 &
        pred_y_lglg[, 2] >= 140)) / length(pred_y_lglg[, 1])

rcc_exp <-
  round(cor.test(pred_y_exp[, 1], pred_y_exp[, 2],
        type = 'kendal')$estimate, 3)
rcc_weib <-
  round(cor.test(pred_y_weib[, 1], pred_y_weib[, 2],
        type = 'kendal')$estimate, 3)
rcc_ln <-
  round(cor.test(pred_y_ln[, 1], pred_y_ln[, 2],
        type = 'kendal')$estimate, 3)
rcc_lglg <-
  round(cor.test(pred_y_lglg[, 1], pred_y_lglg[, 2],
        type = 'kendal')$estimate, 3)

cat(
  paste(
    "full_analysis / mixed / external",
    "\n",
    "\t",
    full_analysis$exp_fr_full$dist,
    "\tSD: ",
    round(sd(res_exp)),
    "RCC: ",
    rcc_exp,
```

```
    "ACC: ",
    round(acc_exp, 3),
    "\n",
    "\t",
    full_analysis$weib_fr_full$dist,
    "\tSD: ",
    round(sd(res_weib)),
    "RCC: ",
    rcc_weib,
    "ACC: ",
    round(acc_weib, 3),
    "\n",
    "\t",
    full_analysis$ln_fr_full$dist,
    "\tSD: ",
    round(sd(res_ln)),
    "RCC: ",
    rcc_ln,
    "ACC: ",
    round(acc_ln, 3),
    "\n",
    "\t",
    full_analysis$lglg_fr_full$dist,
    "\tSD: ",
    round(sd(res_lglg)),
    "RCC: ",
    rcc_lglg,
    "ACC: ",
    round(acc_lglg, 3)
  )
)

## Plot Accuracy fluctuations based on threshold shofting

threshold = 1:8500
acc = rep(0, 8500)

for (i in threshold) {
  acc[i] <-
```

```
    (sum(pred_y_ln[, 1] < i &
    pred_y_ln[, 2] < i) + sum(pred_y_ln[, 1] >= i &
    pred_y_ln[, 2] >= i)) / length(pred_y_ln[, 1])
}

plot(
  threshold,
  acc,
  type = 'l',
  xlim = c(24, 8750),
  ylab = "ACC",
  xlab = "threshold",
  main = "Accuracy of fast/slow
          classification on different thresholds"
)
#axis(1, at = seq(0, 8500, by = 1400))
grid (NULL, NULL, lty = "dotted", col = "lightgray")
abline(h = 0, v = 140, col = "blue")
abline(h = 0, v = 1460, col = "green")
legend(
  "bottomright",
  c("median", "current slow/fast threshold"),
  col = c("blue", "green"),
  lty = 1
)
```

B.6 Linear model metrics

The following script is called independently and provides metrics for the linear models, that we used as a baseline criterion measure.

```
##################################
## Linear models test script ##
##################################

cens_seq <- seq(0, 0.2, by = 0.01)

cens_ <- list()
drop_ <- list()

j <- ratio_cens <- size <- 0
for (i in unique(_data123$deliver_end_date)) {
  j <- j + 1
  valid <- sum(_data123$submit_date < i)
  cens <-
    sum(_data123$deliver_end_date > i & _data123$submit_date < i)
  ratio_cens[j] <- cens / valid
  size[j] <- valid
}

## Loop for calling the surv_an function for multiple censored datasets
## based on cens_seq
z <- 1
covs_here <-
  "y ~ is_defect + pre_post_ga + distinct_component_count +
   distinct_function_count + developer_country + symptom"

for (i_cens in cens_seq) {
  if (i_cens == 0) {
    cat(covs_here, '\n')
  }
  tmp_data <- _data123

  xx <-
    max(which(abs(ratio_cens - i_cens) == min(abs(
      ratio_cens - i_cens
```

```
    ))))
  pos <- xx
  for (k in xx:j) {
    if (size[k] > size[xx] & i_cens < ratio_cens[k]) {
      pos <- k
    }
  }

  data_cens <-
    c(ratio_cens[pos], size[pos], unique(tmp_data$deliver_end_date)[pos])

  tmp_data$cens <- 0
  tmp_data$cens[tmp_data$deliver_end_date > data_cens[3]] <- 1

  ## set "y" of censored records to: "cens_point - submit_date"
  tmp_data$y[tmp_data$cens == 1] <-
    (data_cens[3] - tmp_data$submit_date[tmp_data$cens == 1]) / 3600

  ##keep only the observations that have submit_date < censor point
  _sub_data <- subset(tmp_data, tmp_data$submit_date < data_cens[3])
  # drop levels # not sure if necessary now
  _sub_data$symptom <- droplevels(_sub_data$symptom)

  # drop values that have less than 2 because they can't predict afterwards
  help_var <- table(_sub_data$developer_country)
  _sub_data <-
    _sub_data[_sub_data$developer_country %in%
                names(help_var)[help_var > 2],]
  _sub_data$developer_country <- droplevels(_sub_data$developer_country)

  _sub_data_2 <- _sub_data[_sub_data$cens == 0,]
  _sub_data_2$symptom <- droplevels(_sub_data_2$symptom)

  # drop values that have less than 2 because they can't predict afterwards
  help_var <- table(_sub_data_2$developer_country)
  _sub_data_2 <-
    _sub_data_2[_sub_data_2$developer_country %in%
                  names(help_var)[help_var > 2],]
  _sub_data_2$developer_country <-
```

```
    droplevels(_sub_data_2$developer_country)

  ## "full linear model" / cens / fixed-effect / internal
  ##########################################################
  llm_fit <- lm(as.formula(covs_here), data = _sub_data)
  llm_y_hat <- predict(llm_fit)
  res_llm_fit <- llm_y_hat - _sub_data$y
  res1 <- sd(res_llm_fit)
  kendal_cor_1 <-
    round(cor.test(llm_y_hat, _sub_data$y, type = 'kendal')$estimate, 3)

  print(paste(
    "cens_fixed_lm",
    i_cens,
    "(",
    round(data_cens[1], 3),
    ")",
    nrow(_sub_data),
    round(res1),
    kendal_cor_1
  ))

  # ## "full linear model" / cens / fixed-effect / external
  # ##########################################################
  llm_y_hat_ext <- predict(llm_fit, newdata = _data4)
  res_llm_fit_ext <- llm_y_hat_ext - _data4_full$y
  res1_ext <- sd(res_llm_fit_ext)
  kendal_cor_1_ext <-
    round(cor.test(llm_y_hat_ext, _data4_full$y,
          type = 'kendal')$estimate, 3)

  print(
    paste(
      "cens_fixed_lm_external",
      i_cens,
      "(",
      round(data_cens[1], 3),
      ")",
      nrow(_sub_data),
```

```
    " - ",
    round(res1_ext),
    kendal_cor_1_ext
  )
)

## "drop linear model" / drop / fixed-effect / internal
###########################################################
llm_fit <- lm(as.formula(covs_here), data = _sub_data_2)
llm_y_hat <- predict(llm_fit)
res_llm_fit <- llm_y_hat - _sub_data_2$y
res2 <- sd(res_llm_fit)

kendal_cor_2 <-
  round(cor.test(llm_y_hat, _sub_data_2$y, type = 'kendal')$estimate, 3)

print(paste(
  "drop fixed lm",
  i_cens,
  "(",
  round(data_cens[1], 3),
  ")",
  nrow(_sub_data),
  round(res2),
  kendal_cor_2
))

## "drop linear model" / drop / fixed-effect / external
###########################################################
llm_y_hat_ext <- predict(llm_fit, newdata = _data4)
res_llm_fit_ext <- llm_y_hat_ext - _data4_full$y
res1_ext <- sd(res_llm_fit_ext)

kendal_cor_1_ext <-
  round(cor.test(llm_y_hat_ext, _data4_full$y,
        type = 'kendal')$estimate, 3)

print(
  paste(
```

```
      "drop fixed lm external",
      i_cens,
      "(",
      round(data_cens[1], 3),
      ")",
      nrow(_sub_data),
      " - ",
      round(res1_ext),
      kendal_cor_1_ext
    )
  )

  ## "full linear model" / cens / mixed-effect (1|developer) / internal
  ##########################################################################
  llmer_fit <-
    lmer(as.formula(paste(covs_here, " + (1|developer)")),
         data = _sub_data)
  llmer_y_hat <- predict(llmer_fit)
  res_llmer_fit <- llmer_y_hat - _sub_data$y
  res3 <- sd(res_llmer_fit)

  kendal_cor_3 <-
    round(cor.test(llmer_y_hat, _sub_data$y, type = 'kendal')$estimate, 3)

  print(paste(
    "cens mixed lmer",
    i_cens,
    "(",
    data_cens[1],
    ")",
    nrow(_sub_data),
    round(res3),
    kendal_cor_3
  ))

  ## "full linear model" / cens / mixed-effect (1|developer) / external
  ##########################################################################
  llmer_y_hat_ext <-
    predict(llmer_fit,
```

```
              newdata = _data4_fr,
              allow.new.levels = TRUE)
res_llmer_fit_ext <- llmer_y_hat_ext - _data4_full$y
res1_ext <- sd(res_llmer_fit_ext)

kendal_cor_1_ext <-
  round(cor.test(llmer_y_hat_ext, _data4_full$y,
        type = 'kendal')$estimate, 3)

print(
  paste(
    "cens mixed lmer external",
    i_cens,
    "(",
    round(data_cens[1], 3),
    ")",
    nrow(_sub_data),
    " - ",
    round(res1_ext),
    kendal_cor_1_ext
  )
)

## "drop linear model" / drop / mixed-effect (1|developer) / internal
#########################################################################
llmer_fit <-
  lmer(as.formula(paste(covs_here, " + (1|developer)")),
       data = _sub_data_2)
llmer_y_hat <- predict(llmer_fit)
res_llmer_fit <- llmer_y_hat - _sub_data_2$y
res4 <- sd(res_llmer_fit)

kendal_cor_4 <-
  round(cor.test(llmer_y_hat, _sub_data_2$y, type = 'kendal')$estimate,
        3)

print(paste(
  "drop mixed lmer ",
  i_cens,
```

```
      "(",
      data_cens[1],
      ")",
      nrow(_sub_data_2),
      round(res4),
      kendal_cor_4
    ))

  ## "drop linear model" / drop / mixed-effect (1|developer) / external
  ##########################################################################
  llmer_y_hat_ext <-
    predict(llmer_fit,
            newdata = _data4_fr,
            allow.new.levels = TRUE)
  res_llmer_fit_ext <- llmer_y_hat_ext - _data4_full$y
  res1_ext <- sd(res_llmer_fit_ext)

  kendal_cor_1_ext <-
    round(cor.test(llmer_y_hat_ext, _data4_full$y,
          type = 'kendal')$estimate, 3)

  print(
    paste(
      "drop mixed lmer external",
      i_cens,
      "(",
      round(data_cens[1], 3),
      ")",
      nrow(_sub_data),
      " - ",
      round(res1_ext),
      kendal_cor_1_ext
    )
  )

  z <- z + 1
}
```

References

[1] Walid AbdelMoez, Mohamed Kholief, and Fayrouz M. Elsalmy. Improving bug fix-time prediction model by filtering out outliers. In *2013 The International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE)*, pages 359–364. IEEE, May 2013.

[2] Hirotugu Akaike. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723, 1974.

[3] Prasanth Anbalagan and Mladen Vouk. On predicting the time taken to correct bug reports in open source projects. In *2009 IEEE International Conference on Software Maintenance*, pages 523–526. IEEE, Sep 2009.

[4] Pamela Bhattacharya and Iulian Neamtiu. Bug-fix time prediction models. In *Proceeding of the 8th working conference on Mining software repositories - MSR '11*, page 207, New York, New York, USA, May 2011. ACM Press.

[5] Stamatia Bibi, Grigorios Tsoumakas, Ioannis Stamelos, and Ioannis P Vlahavas. Software defect prediction using regression via classification. In *AICCSA*, pages 330–336, 2006.

[6] Michael Borenstein, Larry V. Hedges, Julian P. T. Higgins, and Hannah R. Rothstein. *Fixed-Effect Model*, pages 63–67. John Wiley & Sons, Ltd, 2009.

[7] Michael Borenstein, Larry V. Hedges, Julian P. T. Higgins, and Hannah R. Rothstein. *Random-Effects Model*, pages 69–75. John Wiley & Sons, Ltd, 2009.

[8] A. Colin Cameron and Frank A.G. Windmeijer. R-squared measures for count data regression models with applications to health-care utilization. *Journal of Business & Economic Statistics*, 14(2):209–220, 1996.

[9] Gerardo Canfora, Michele Ceccarelli, Luigi Cerulo, and Massimiliano Di Penta. How Long Does a Bug Survive? An Empirical Study. In *2011 18th Working Conference on Reverse Engineering*, pages 191–200. IEEE, Oct 2011.

[10] Jeffrey C. Carver. Towards reporting guidelines for experimental replications: A proposal. In *1st International Workshop on Replication in Empirical Software Engineering*, pages 1–4, 2010.

[11] Enrico Colosimo, Flávio Ferreira, Maristela Oliveira, and Cleide Sousa. Empirical comparisons between kaplan-meier and nelson-aalen survival function estimators. *Journal of Statistical Computation and Simulation*, 72(4):299–308, 2002.

[12] R. Dennis Cook and Sanford Weisberg. *Residuals and Influence in Regression*. Monographs on statistics and applied probability. New York: Chapman and Hall, 1982.

[13] D. R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(2):187–220, 1972.

[14] Marco D'Ambros, Michele Lanza, and Romain Robbes. An extensive comparison of bug prediction approaches. In *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, pages 31–41. IEEE, May 2010.

[15] Norman E. Fenton and Martin Neil. A critique of software defect prediction models. *Software Engineering, IEEE Transactions on*, 25(5):675–689, 1999.

[16] Thomas R. Fleming and David P. Harrington. *Counting processes and survival analysis*, volume 169. John Wiley & Sons, 2011.

[17] Ernst G. Frankel. *Systems reliability and risk analysis*, volume 1. Springer Science & Business Media, 2013.

[18] Emanuel Giger, Martin Pinzger, and Harald Gall. Predicting the fix time of bugs. In *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*, pages 52–56. ACM, 2010.

[19] Mayy Habayeb. On the use of hidden markov model to predict the time to fix bugs. Master's thesis, Ryerson University, 2015.

[20] Rattikorn Hewett and Phongphun Kijsanayothin. On modeling software defect repair time. *Empirical Software Engineering*, 14(2):165–186, May 2008.

[21] Pieter Hooimeijer and Westley Weimer. Modeling bug report quality. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering - ASE '07*, page 34, New York, New York, USA, Nov 2007. ACM Press.

[22] David W. Hosmer Jr and Stanley Lemeshow. *Applied Survival Analysis: Time-to-Event*, volume 317. Wiley-Interscience, 1999.

[23] Philip Hougaard. Frailty models for survival data. *Lifetime data analysis*, 1(3):255–273, 1995.

[24] Nicholas Jalbert and Westley Weimer. Automated duplicate detection for bug tracking systems. In *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*, pages 52–61. IEEE, 2008.

[25] Robert I. Jennrich and Stephen M. Robinson. A newton-raphson algorithm for maximum likelihood factor analysis. *Psychometrika*, 34(1):111–123, 1969.

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

[27] Chris F. Kemerer and Mark C. Paulk. The impact of design and code reviews on software quality: An empirical study based on psp data. *IEEE Transactions on Software Engineering*, 35(4):534–550, 2009.

[28] Maurice G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[29] Taghi M. Khoshgoftaar and Edward B. Allen. Logistic regression modeling of software quality. *International Journal of Reliability, Quality and Safety Engineering*, 6(04):303–317, 1999.

[30] Sunghun Kim and E. James Whitehead. How long did it take to fix bugs? In *Proceedings of the 2006 international workshop on Mining software repositories - MSR '06*, page 173, New York, New York, USA, May 2006. ACM Press.

[31] John P. Klein and Melvin L. Moeschberger. *Survival analysis: techniques for censored and truncated data*. Springer Science & Business Media, 2005.

[32] David Kleinbaum, Lawrence Kupper, Azhar Nizam, and Eli Rosenberg. *Applied regression analysis and other multivariable methods*. Nelson Education, 2013.

[33] A. Gunes Koru, Dongsong Zhang, and Hongfang Liu. Modeling the Effect of Size on Defect Proneness for Open-Source Software. In *29th International Conference on Software Engineering (ICSE'07 Companion)*, pages 115–124. IEEE, May 2007.

[34] Tarald O. Kvålseth. Cautionary note about r 2. *The American Statistician*, 39(4):279–285, 1985.

[35] Elisa T. Lee and John Wang. *Statistical methods for survival data analysis*, volume 476. John Wiley & Sons, 2003.

[36] M.M. Lehman. Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE*, 68(9):1060–1076, 1980.

[37] Stefan Lessmann, Bart Baesens, Christophe Mues, and Swantje Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *Software Engineering, IEEE Transactions on*, 34(4):485–496, 2008.

[38] Kwan-Moon Leung, Robert M. Elashoff, and Abdelmonem A. Afifi. Censoring issues in survival analysis. *Annual review of public health*, 18(1):83–104, 1997.

[39] Lionel Marks, Ying Zou, and Ahmed E. Hassan. Studying the fix-time for bugs in large open source projects. In *Proceedings of the 7th International Conference on Predictive Models in Software Engineering - Promise '11*, pages 1–8, New York, New York, USA, Sep 2011. ACM Press.

[40] Douglas C. Montgomery, Elizabeth A. Peck, and G. Geoffrey Vining. *Introduction to linear regression analysis*. John Wiley & Sons, 2015.

[41] Nachiappan Nagappan and Thomas Ball. Use of relative code churn measures to predict system defect density. In *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on*, pages 284–292. IEEE, 2005.

[42] Daniel J. Ozer. Correlation and the coefficient of determination. *Psychological Bulletin*, 97(2):307, 1985.

[43] Lucas D. Panjer. Predicting Eclipse Bug Lifetimes. In *Fourth International Workshop on Mining Software Repositories (MSR'07:ICSE Workshops 2007)*, pages 29–29. IEEE, May 2007.

[44] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[45] Joost Schalken, Sjaak Brinkkemper, and Hans Van Vliet. Using linear regression models to analyse the effect of software process improvement. In *International Conference on Product Focused Software Process Improvement*, pages 234–248. Springer, 2006.

[46] F. W. Scholz. Maximum likelihood estimation. *Encyclopedia of Statistical Sciences*, 1985.

[47] Nicolas Serrano and Ismael Ciordia. Bugzilla, itracker, and other bug trackers. *Software, IEEE*, 22(2):11–13, 2005.

[48] Forrest Shull. Research 2.0? *IEEE Software*, 29(6):4–8, 2012.

[49] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[50] Andrej-Nikolai Spiess and Natalie Neumeyer. An evaluation of r2 as an inadequate measure for nonlinear models in pharmacological and biochemical research: a monte carlo approach. *BMC pharmacology*, 10(1):6, 2010.

[51] Cathrin Weiß, Rahul Premraj, Thomas Zimmermann, and Andreas Zeller. How Long Will It Take to Fix This Bug? In *Fourth International Workshop on Mining Software Repositories (MSR'07:ICSE Workshops 2007)*, pages 1–1. IEEE, May 2007.

[52] Robert K. Yin. *Case study research: Design and methods*. Sage publications, 5th edition, 2013.

[53] Feng Zhang, Foutse Khomh, Ying Zou, and Ahmed E. Hassan. An Empirical Study on Factors Impacting Bug Fixing Time. In *2012 19th Working Conference on Reverse Engineering*, pages 225–234. IEEE, Oct 2012.

[54] Hongyu Zhang, Liang Gong, and Steve Versteeg. Predicting bug-fixing time: An empirical study of commercial software projects. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 1042–1051, 2013.

Index

Accuracy (ACC), 37
Akaike Information Criterion (AIC), 35

Censoring, 21
Cox proportional hazard models, 28

Diagnostics, 35

Exponential distribution, 30

Fixed-effect models, 32
Frailty models, 34

Hazard function, 27

Kaplan-Meier estimator, 28
Kendall rank correlation coefficient (RCC), 37

Loglogistic distribution, 31
Lognormal distribution, 31

Maximum likelihood estimation (MLE), 21

Nelson-Aalen estimator, 28

Ordinary least squares (OLS), 19

Parametric survival models, 29

R-squared, 36
Random-effects models, 33
Research questions, 2
Residual standard deviation, 36

Skewness, 24
Survival function, 26

Time-to-deliver, 2
Time-to-fix, 1
Transformations, 25
Truncation, 38

Weibull distribution, 31