THESIS FOR THE DEGREE OF LICENTIATE OF ENGINEERING

Online experimentation in automotive software engineering

Yuchu Liu

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2022

Online experimentation in automotive software engineering

Yuchu Liu

ISSN 1652-876X
This thesis has been prepared using LaTeX.

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Göteborg, Sweden
Phone: +46 (0)31 772 1000
www.chalmers.se

Printed by Chalmers Reproservice
Göteborg, Sweden, May, 2022

To friends, the family we choose.

Abstract

Context: Online experimentation has long been the gold standard for evaluating software towards the actual needs and preferences of customers. In the Software-as-a-Service domain, various online experimentation techniques are applied and proven successful. As software is becoming the main differentiator for automotive products, the automotive sector has started to express an interest in adopting online experimentation to strengthen their software development process.
Objective: In this research, we aim to systematically address the challenges in adopting online experimentation in the automotive domain.
Method: We apply a multidisciplinary approach to this research. To understand the state-of-practise in online experimentation in the industry, we conduct case studies with three manufacturers. We introduce our experimental design and evaluation methods to real vehicles driven by customers at scale. Moreover, we run experiments to quantitatively evaluate experiment design and causal inference models.
Results: Four main research outcomes are presented in this thesis. First, we propose an architecture for continuous online experimentation given the limitations experienced in the automotive domain. Second, after identifying an inherent limitation of sample sizes in the automotive domain, we apply and evaluate an experimentation design method. The method allows us to utilise pre-experimental data for generating balanced groups even when sample sizes are limited. Third, we present an alternative approach to randomised experiments and demonstrate the application of Bayesian causal inference in online software evaluation. With the models, we enable software online evaluation without the need for a fully randomised experiment. Finally, we relate the formal assumption in the Bayesian causal models to the implications in practise, and we demonstrate the inference models with cases from the automotive domain.
Outlook: In our future work, we plan to explore causal structural and graphical models applied in software engineering, and demonstrate the application of causal discovery in machine learning-based autonomous drive software.

Keywords: Automotive software, Bayesian statistics, Causal inference, Embedded software, Online experimentation.

List of Publications

This thesis is based on the following publications:

[A] **Yuchu Liu**, Jan Bosch, Helena Holmström Olsson, Jonn Lantz, “An architecture for enabling A/B experiments in automotive embedded software”. Published in 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC), July. 992-997.

[B] **Yuchu Liu**, David Issa Mattos, Jan Bosch, Helena Holmström Olsson, Jonn Lantz, “Size matters? Or not: A/B testing with limited samples in automotive embedded software”. Published in 2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Sep. 300-307.

[C] **Yuchu Liu**, David Issa Mattos, Jan Bosch, Helena Holmström Olsson, Jonn Lantz, “Bayesian propensity score matching in automotive embedded software engineering”. Published in 2021 28th Asia-Pacific Software Engineering Conference (APSEC 2021), Dec. 233-242.

[D] **Yuchu Liu**, David Issa Mattos, Jan Bosch, Helena Holmström Olsson, Jonn Lantz, “Bayesian causal inference in automotive software engineering and online evaluation”. In submission to a Software Engineering journal.

Publication not included in this thesis:

[E] David Issa Mattos and **Yuchu Liu**, "On the Use of Causal Graphical Models for Designing Experiments in the Automotive Domain", To appear at 2022 25th Evaluation and Assessment in Software Engineering (EASE 2022), June.

Individual Contributions

I am the main author of all publication included in this thesis. I am responsible for the planning, execution, analysis, and reporting of all research work included in this thesis. The list below specify my contribution follow the CRediT (Contributor Roles Taxonomy) system.

Conceptualisation: formulation of the research goals and aims in Paper A, C, and D.

Methodology: developing and formulating the research methodology in Paper A, B, C, and D.

Software: programming the statistical models presented in Paper B, C, and D, no software was used for the data analysis in Paper A.

Formal analysis: formal analysis and data processing in Paper A, B, C, and D.

Investigation: conducting a investigation process, specifically performing the experiments, interviews and data collection in Paper A, B, C, and D.

Data Curating: managing and warehousing data collection in Paper A, B, C, and D.

Writing: I am the main author and produced the complete draft of Paper A, B, C, and D.

Project administration: I am responsible for the coordination with case study companies and project management in our studies from Paper A, B, and C.

Acknowledgments

First, I would like to extend my gratitude to my supervisors, Professor Jan Bosch, Professor Helena Holmström Olsson, and Dr. Jonn Lantz. I am grateful for their patient guidance during the research and expert knowledge on the topic. I would also like to extend my gratitude to Dr. David Issa Mattos, for his insightful contribution in our collaboration. Moreover, I would like to pay a tribute to my late examiner Professor Ivica Crnkovic for his support during my research.
Second, I would like to take this opportunity to thank our industry collaborators, Volvo Cars, AB Volvo, CEVT, and Zenseact, for sharing their resources and experience on the topic.
Third, I would like to thank some of my fellow colleagues who are also pursuing their doctorates from Volvo Cars and Chalmers University of Technology. Although the collaboration opportunities were limited, they have been incredibly supportive and reminded me that I am not alone in this journey.
Last but not least, I am extremely grateful for all my friends, especially Andrei Ursachi, Bogdan Constantin Pomohaci, Brenda Eliza Harms, Carlos Viñas White, and Martynas Šapoka. Without you, I will not be who I am today.

Contents

Abstract i

List of Papers iii

Individual Contribution v

Acknowledgements vii

1 Introduction 1

2 Background 5
2.1 Online experimentation 5
2.2 Challenges in automotive 8
Hardware dependency 8
Operation environments 9
Non-functional requirements 9
Functional requirements 10
2.3 Potential outcome and causal inference 11
Causality 11
Stable unit treatment value assumption 13
Consistency 13
Exchangeability 13

Positivity 15
2.4 Bayesian inference 15

3 Research objective and method **19**
3.1 Objective 19
RQ1: How to run large-scale online experiments in the automotive domain in a fast and reliable fashion? 20
RQ2: How can the inherent limitation of sample sizes in the automotive domain be addressed? 21
RQ3: Can causality be concluded in the absence of randomisation in a software online evaluation? 21
3.2 Method 22
Literature review 23
Case study 24
Empirical Experiment 25
3.3 Threats to validity 27
Internal validity 27
External validity 27

4 Summary of included papers **29**
4.1 Paper A 29
4.2 Paper B 30
4.3 Paper C 30
4.4 Paper D 31

5 An architecture for enabling A/B experiments in automotive embedded software **33**
5.1 Introduction 33
5.2 Background and constraints 34
Background 34
Constraints 35
5.3 Research method 37
Literature review 37
Case study 39
5.4 Existing architectures 40
5.5 Architecture and case studies 42
System requirements 43

System characteristics . . . 44
Case study I . . . 45
Case study II . . . 47
5.6 Discussion . . . 48
5.7 Conclusion . . . 49

6 Size matters? Or not: A/B testing with limited sample in automotive embedded software 51
6.1 Introduction . . . 51
6.2 Background . . . 53
6.3 Research method . . . 54
Data collection . . . 55
Validity considerations . . . 56
6.4 The Balance Match Weighted design . . . 57
The Balance Match Weighted design . . . 58
Feature selection . . . 58
Propensity score distance . . . 59
Greedy full matching . . . 60
The repetition parameter M . . . 61
6.5 Case study . . . 62
Case study fleet . . . 62
Selected features . . . 63
Matched A/B groups . . . 65
Experiment outcome . . . 66
Recommended procedure . . . 67
6.6 Discussion . . . 69
Existing data and unobserved variables . . . 69
Multiple driver households and car sharing . . . 70
6.7 Conclusion . . . 70

7 Bayesian propensity score matching in automotive embedded software engineering 73
7.1 Introduction . . . 73
7.2 Background and related work . . . 75
7.3 Input data . . . 77
Data structure . . . 79

7.4 Bayesian propensity score matching . . . 79
Probabilistic graphical model . . . 81
Matching . . . 83
7.5 Results . . . 85
Bayesian propensity score . . . 85
Matched A/B groups . . . 86
Treatment effect . . . 88
Bayesian propensity score matching for observational test . . . 89
7.6 Discussion . . . 92
Threats to validity . . . 92
Limitations . . . 93
7.7 Conclusion . . . 94

8 Bayesian causal inference in automotive software engineering and online evaluation 97
8.1 Introduction . . . 97
8.2 Background . . . 99
Randomised experimentation . . . 99
The potential outcomes framework . . . 100
Bayesian statistics and inference . . . 102
8.3 The BOAT framework . . . 104
BOAT . . . 104
Research Method . . . 107
8.4 Bayesian propensity score matching . . . 111
Theory . . . 113
Study I: Limited access to users . . . 116
Results . . . 117
8.5 Bayesian difference-in-differences . . . 120
Theory . . . 121
Study II: Seasonality effect . . . 123
Results . . . 127
8.6 Bayesian regression discontinuity . . . 129
Theory . . . 130
Study III: Covariate dependent treatment assignment . . . 133
Results . . . 134
8.7 Discussion . . . 137
Causal assumptions and domain knowledge . . . 138

Extension to BOAT 138
8.8 Conclusion 140

9 Concluding remarks and future work **143**
9.1 Conclusion 143
RQ1: How to run large-scale online experiments in the automotive domain in a fast and reliable fashion? 144
RQ2: How can the inherent limitation of sample sizes in the automotive domain be addressed? 144
RQ3: Can causality be concluded in the absence of randomisation in a software online evaluation? 145
9.2 Future work 145

References **149**

List of Figures

1.1 Challenges of online experimentation adoption in automotive and proposed approaches and solutions. 3

2.1 An illustration of a two-level experiment. 6

2.2 An illustration of relationships between treatment, outcome, and confounder in a controlled experiment and an observational study. 12

2.3 A simplified graph showing the relationships of treatment ($t \in T$), target variable ($y \in Y$), and confounding factors ($x \in \mathbf{X}$). . 14

2.4 An illustration of Bayesian inference of prior, evidence, and posterior distributions. 16

3.1 A timeline showing the research questions, research methods, and the corresponding activities. 22

5.1 Existing A/B experiment framework categorised by environment and variant generation methods. 41

5.2 Process of the cloud-based A/B test architecture, illustrating the general work flow of conducting an A/B experiment with parametrised functions. 42

6.1 Relationships of input features ($\mathbf{X}$), treatment (τ) and target variable (Y) in the propensity score matching model. 57
6.2 Minimum total distance (Δk) calculated from the propensity scores reduces as we increase the repetitions (M). 61
6.3 Scatter plots of feature 0 through 5 and their correlation to the target variable, min-max scaled. 64
6.4 Kernel density estimation of the target variable, min-max scaled, of A and B groups when matched at random (left), and matched using the Balance Match Weighted design (right). 66

7.1 Probably graphical model of a Bayesian logistic regression, with observed input features ($\mathbf{x_n}$), treatment indicator (y_n), and latent variables as regression model coefficients (α, β). 82
7.2 Posterior distributions of the Bayesian logistic regression coefficient $\beta = \{\beta_1, ..., \beta_{14}\}$, and intercept α. 84
7.3 Kernel density distribution of the propensity scores of the control (p_c) and treatment (p_t) groups calculated on the mean of posterior distributions, and twenty-five values randomly sampled from the posterior distributions representing uncertainties. 86
7.4 Kernel density distributions of the Bayesian propensity scores for the control (p_c) and treatment (p_t) group, when, no matching was done, matched with a caliper at 0.05, and matched with 1-1 nearest neighbour. 87
7.5 Online software evaluation with limited sample sizes, by utilising Bayesian propensity score matching. 89

8.1 A simplified directed acyclic graph showing the relationships of treatment (t), target variable (y), and covariates ($\mathbf{X}$). 100
8.2 A decision flowchart on which Bayesian causal model from the BOAT framework to apply when designing an online software evaluation. (BRDD: Bayesian regression discontinuity, BPSM: Bayesian propensity score matching, BDID: Bayesian difference-in-differences) 106
8.3 An illustration explaining the Propensity Score Matching model. Note, figure does not represent real data. 112

8.4 Kernel density distribution of the propensity scores of the control (p_c) and treatment (p_t) groups calculated on the mean of posterior distributions, and twenty-five values randomly sampled from the posterior distributions representing uncertainties. 118
8.5 An illustration explaining the Difference-in-Differences model and how the average treatment effect (ATE) is estimated. . . . 120
8.6 A simplified directed acyclic graph showing the relationships of treatment (t), target variable (y), covariates ($\mathbf{X}$), and time dependent latent variables summarised as τ. 122
8.7 Target variable y measured before the treatment (τ_{-1}), when the treatment is applied (τ), and after the treatment (τ_1), for the control and the treatment groups, y is min-max scaled. . . 128
8.8 An illustration explaining the Regression Discontinuity Design model, and how the average treatment effect is estimated. . . . 129
8.9 A simplified directed acyclic graph showing the relationships of treatment (t), target variable (y), assignment variable (X), the cut-off point (c), and other confounding factors ($\mathbf{Z}$). 130
8.10 Target variable measured before and after the cut-off point ($c = 60$), with respect to the assignment variable. 136
8.11 Posterior distribution of the regression coefficients β, and the regression intercept α. 137
8.12 Posterior distribution of the regression coefficients β, β is ordered as Table. 8.5, and the regression intercept α. 142

9.1 Challenges of online experimentation adoption in automotive, proposed approaches and solutions, and future outlook of this doctorate research. 146

List of Tables

5.1 Papers selected which describing architecture of A/B experiments, for web and embedded software). 38
5.2 System components adopted by case study companies. 47

6.1 Mean and variance of each of the five input features **X**, min-max scaled, in the matched control and treatment groups. . . . 65

7.1 Descriptive statistics of the target variable and covariates, and a description of how the variables are computed. Each variable is aggregated to the vehicle level and max-min scaled. 77
7.2 Propensity score in control and treatment groups, before and after matching is applied. 87

8.1 Guidelines of design science research method [95], and practices applied following the guidelines in this research. 108
8.2 Propensity scores in control and treatment groups, before and after a matching is performed. 119
8.3 Descriptive statistics of the target variable and covariates as inputs to Bayesian difference-in-differences model, and a description of how the variables are computed. Each variable is aggregated to the vehicle level and min-max scaled. 125

8.4 Average energy consumption (Wh/km) for the control and the treatment group at each time step. 127

CHAPTER 1

Introduction

With the digitalisation of modern vehicles, software is becoming the main focus of the automotive industry [1]–[3]. To remain competitive, gaining a good understanding of real-world feedback such as vehicle usage and customer preference becomes of crucial importance. Companies are collecting and utilising more and more customer data to build a comprehensive understanding of customer preferences and to improve their decision making when designing the software. The automotive domain is no exception.

Traditionally, automotive software development is driven by specifications and requirements [4]. Due to the nature of such a development method, the software specifications and requirements are frequently composed well before the end product is introduced to the customers [5]. As a result, much of the requirements are based on limited understanding of how the software functionalities will be used; more often than not, there is little to no existing data to support such assumptions. In recent years, learning lessons from the success of data-driven software development in Software-as-a-Service (SaaS) companies [6]–[8], the automotive domain is collecting more and more vehicle and customer data to understand the use of their vehicles in the real world; therefore, inform development organisations of more direct customer preference [9]

and vehicle performance in a wider range of real-world conditions.

To adapt to change more rapidly and continuously deliver new software features, more and more automotive companies are adopting fast and flexible contemporary development approaches, such as Agile, in their software organisations [3], [9]–[11]. At its core, the Agile methodology requires constant feedback from stakeholders as input to feature prioritisation and properly define user stories. Although qualitative techniques such as surveys, customer workshops, and interviews are valid approaches, these methods are expensive, time-consuming, and as the reach is limited, companies often struggle to collect representative samples. Moreover, while providing customer insights, none of these qualitative approaches can help us to understand the performance of the vehicles themselves. Therefore, quantitative approaches powered by large datasets are more desirable for fast and iterative software evaluation.

There are many well-known challenges of big data application in the automotive domain, such as data collection, data transfer efficiency and handling [3], [12], and data distribution within large development organisations [13], [14]. In addition to all of mentioned above, unstructured and pure observational data do not offer counterfactual answers to interventions similar to a software update - nowhere in a purely observed empirical distribution informs you if a software change could improve the vehicle performance or the user experience [15]. Climbing the ladder of causation requires a shift from observation to intervention, e.g., experimentation. Thus, a systematic and structured approach to software evaluation, online experimentation, has long been the gold standard in the SaaS domain [16]–[18].

There are many forms of online experimentation. It is essentially a scientific approach to controlling variables and concluding counterfactual outcomes [19], one of which is the most well-known A/B testing. A/B testing is a two-level experiment, the control (typically an existing version of a software feature, and the treatment (a modified version of the same software feature). A large amount of users are randomly split into two groups and are exposed to the control or treatment version of the software at random, a process that ensures the exchangeability of the two groups. A causal relationship between treatment and effect can be established when the exchangeability is satisfied, since it explicitly states that the treatment effect is a result of the treatment and not the existing differences between the groups, observed or not [20]. By the

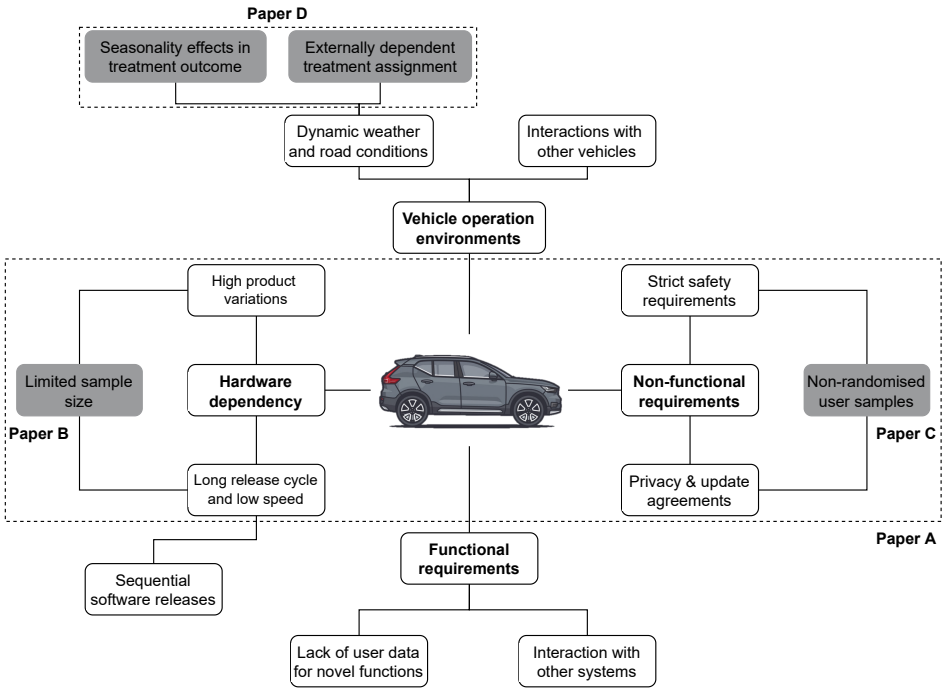

Figure 1.1: Challenges of online experimentation adoption in automotive and proposed approaches and solutions.

law of large numbers, the likelihood of balanced control and treatment groups through randomisation becomes higher when the sample size becomes larger; then an unbiased treatment effect can be inferred [16], [21].

The benefits of online experimentation experienced in the SaaS domain are also expected or reported in automotive; however, the adoption of the method is not without challenges [3], [22], [23]. As reported in literature and from practise, the challenges include limited sample sizes, long software release time, and so on. In this thesis, I summarise the challenges and the symptoms they manifest in four main aspects. These are: (a) hardware dependency - an inherent limitation of embedded software, (b) functional and (c) non-functional requirements - the former limits our understanding for experiment design and the latter limits the available experimentation sample sizes, and

finally, (d) the experimentation is carried out in heterogeneous environments in terms of weather, road and traffic conditions, and so on. The challenges and their corresponding proposed approaches are summarised in a framework in Fig. 1.1, the approaches presented in all publications included in this thesis are a combination of engineering solutions and statistical modelling ones.

Compared to a limited collection of existing publications on online automotive software experimentation [2], [3], [12], [22], [24], [25], this thesis offers novelty in the following four aspects. First, we adopt an empirical approach to this research and conduct the study in close collaboration with the automotive industry. Some of the first online experimentation practises on real-world customer vehicles in the automotive sector are documented in this thesis; as well as industry insight and state-of-practise is reported in detail. Second, we apply and evaluate statistical models for two-level experiment design and observational studies; while these statistical models have been explored in other areas of science, the publications included in this thesis are the first ones to apply them in software engineering online experimentation. Third, given the limitations in the automotive sector, an alternative approach powered by Bayesian causal inference is proposed. This approach allows us to evaluate software in an online and continuous manner when a fully randomised experiment is impossible, unethical, or undesired. Last but not least, we relate the theory of causal assumptions and Bayesian causal inference to cases experienced in the automotive domain, aiming to enable software online evaluation and infer causality without the need of a fully randomised experiment.

The rest of this thesis is organised as follows; Chapter 2 introduces the background of online experimentation, existing practises, and the theoretical background of experiment. In Chapter 3, the research objective and method is described. The summary of publications included in this thesis is presented in Chapter 4, following the four publications listed in Chapters 5 to 8. In paper A (Chapter 5), a comprehensive architectural solution is proposed aiming to enable speed of online experimentation via a hybrid edge/cloud architecture. Paper B (Chapter 6) presents an experiment design method utilising pre-experimental data, and it enables quasi-random experiments even when sample sizes are limited. An alternative approach to randomised experimentation is proposed and evaluated in paper C and D (Chapter 7 & 8). In Chapter 9, I discuss the concluding remarks and the future direction of the research.

CHAPTER 2

Background

In this chapter, the background of online experimentation in the context of embedded software is described. First, existing online experiment practises are summarised from the literature and listed. Second, the challenges of adopting online experimentation in the automotive sector are described. Third, software online experimentation is put in the context of the Rubin potential outcome framework, with the causal assumptions and the outcome model formalised. Finally, an introduction to Bayesian statistics and its application in causal inference is provided.

2.1 Online experimentation

From A/B testing to multi-armed bandits, online experimentation comes in many shapes and flavours, however, all of which work with the fundamental principle of causal inference, establishing a causal relationship between the treatment and the effects through intervention.

In essence, the most straightforward and the most practised online experimentation method [7], [8], [26], A/B testing, is a two-level experiment. When performing an A / B test, the total sample of users ($n \in N$) is randomly di-

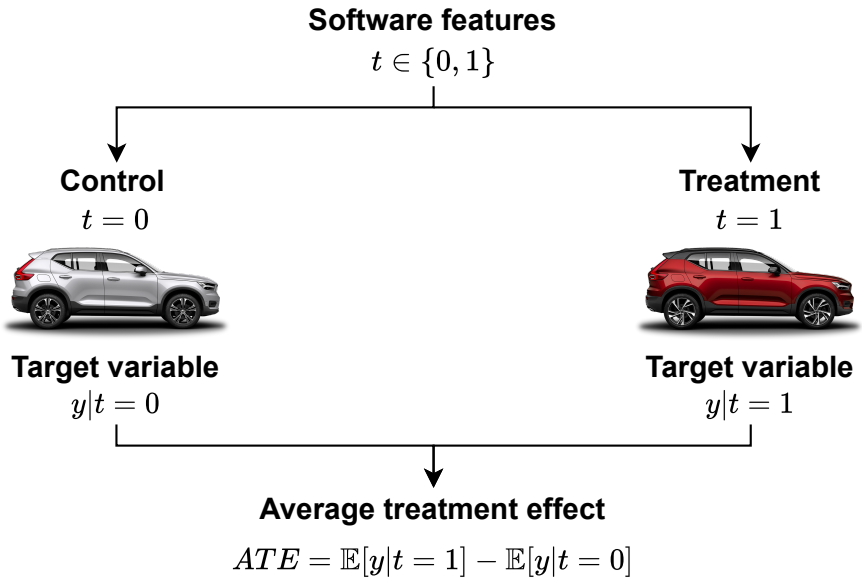

Figure 2.1: An illustration of a two-level experiment.

vided into control groups (N_c) and treatment (N_t) groups, two versions of the same software feature ($t = \{0, 1\}$) are then introduced to each corresponding group; in an example where a software update is being evaluated, it can be written as $t = \{\text{old\_version}, \text{new\_version}\}$. A simple illustration is presented in Fig. 2.1. In a properly designed A/B experiment, some performance indicator that reflects the usage and/or the customer preferences is measured. This measurement is defined as a target variable (y) in this thesis, and it embeds the customer preferences and business strategical target for a given software function, some call this the overall evaluation criteria [19]. As a trivial example, suppose we want to evaluate a software component that manages vehicle battery systems, the target variable that measures the treatment effect of the software change could be total energy consumed in the vehicle, $y = [150, 300]$, measured in wh/km. It can be bounded since we are aware of the physical limits of the vehicle. Typically, A/B testing is used to evaluate user-facing features such as user interface or user experience [18], [27]–[29]. In recent

years, A/B testing or similar online experiment practises have also been used for improving machine performance resulting from embedded software changes [22], [24]. The average treatment effect (ATE), is the difference in the mean of the target variable between the control and the treatment group.

When multiple software versions are to be evaluated, an A/B/n test can be conducted if we can safely assume that the software versions are not interacting. An A/B/n test is conducted similarly as an A/B test, with the exception that the there are multiple levels of treatment and the user samples are divided into as many groups as the levels of treatment. However, if the assumption of non-interaction between software changes does not hold, one-factor-at-a-time or factorial experiments need to be designed. The former allows you to evaluate the impact of one software feature at a time while holding all the other changes constant; the latter allows you to study the interactions between the software treatment and the target variable and the interactions between different software features themselves. In a full factorial design, to evaluate k different software features at two levels (e.g., $t = \{\text{old}, \text{new}\}$), the minimum number of experiments required is 2^k. Since the available user samples are limited, dividing them into multiple subgroups for A/B/n or factorial experiments will likely lead to high statistical variance in the measured target variable, low sensitivity, and unbalanced thus incomparable groups.

Another approach to experiment with multiple or continuously changing software versions is the multi-armed bandit problem [30], [31]. Multi-armed bandit is a sequential design of experiments, and it addresses the allocation of resources, or in this case, users samples, for a better understanding of the optimal treatments. In a bandit problem, each k arm of the software treatment, $t = \{t_0, t_1, ..., t_k\}$, provides a random response following an unknown statistical distribution. We could start with absolute no initial knowledge about how the users will response to the software treatments. The goal of most bandit algorithms is to optimise the exploit-explore trade-off, that is, when a user sample n enters the trial, should the user be shown the most effective software version to our best knowledge, or should the user be exposed to another software version to explore the expected response distribution. Multi-level treatment is not further discussed in this thesis; instead, we pay special attention to two-level treatment, since it is the most straightforward approach in online experimentation in software engineering.

Traditionally, majority of the online experiments is analysed in the fre-

quency statistical domain and with statistical tests such as t-test for sufficiency [2], [16], [19], [21], [26], [32]. The likelihood of detecting a treatment effect (or sensitivity) in frequency statistics is directly proportional to the statistical variance and sample size. In order to conclude the treatment effect is sufficiently different from untreated samples, the sensitivity of the average treatment effect needs to be high. The sensitivity of an online experiment can be improved by various techniques, most of which falls in two categories, pre-experimental design and post-experimental modelling. Techniques used for experiment design are, for example, blocking (nuisance factors are held constant within a block) and power analysis for determining sample size based on the expected effect size. Post-experimental techniques including but not limited to, post-stratification and Controlled experiment Using Pre-Experiment Data (CUPED), both of which aim to reduce variance of the target variable thus measuring more sensitive treatment effects [16], [21].

2.2 Challenges in automotive

In this section, the challenges of adopting software online experimentation in the automotive domain are listed. The challenges are derived from a limited set of literature [2], [3], [12], [22] and from the state-of-practise of the industry. The challenges are summarised in four categories, hardware dependency, vehicle operational environment such as weather and road conditions, and those challenges that are related to the software design functional and non-functional requirements that are specific to the automotive sector. The four subsections are not organised in any particular order.

Hardware dependency

The majority of the automotive software are embedded software and they have a strong hardware dependency. Historically, automotive software is delivered together with the computational hardware from suppliers, which results in the following two scenarios. First, automotive manufacturers have the tendency to optimise for unit price and each hardware unit has little to no computational power to spare. Second, the suppliers could own full copyrights to the software, which limits the ability of the manufacturers to change or update the software in the flexible and timely manner. While the later issue is addressed with the

adoption of Agile methodology [11], the former remains unsolved.

From localisation and customisation options, automotive hardware and consequently software have a large diversity in product variations. This results in a highly heterogeneous sample group. Furthermore, there are fewer users compared to the web domain, where the sample size is also reported to be an issue [16], [21]. The available users to experiment with is significantly less in the automotive domain by order of magnitude, at the same time, we often want to experiment with a subset of the vehicles and users and this result in a further reduction of sample sizes.

Operation environments

Different from a website or a mobile phone, the operation environment of a vehicle is diverse and often unpredictable which creates a dynamic range of contexts for software online experimentation. The performance of some functions is strongly influenced by changing weather and road conditions. For instance, a popular semi-autonomous feature, lane keeping assist, relies on lane marking that can be different in style and quality from location to location, thus becoming a confounding factor when evaluating a related software. While we could rely on randomisation for producing balanced control and treatment groups when the sample size is large, in practise, the experiments and group designed could be unbalanced and confounded on the temporal or location related conditions when we lack millions of users.

Furthermore, because of seasonality effects, some experiments need to be conducted longitudinally, and unobserved temporal confounders might prevail. Lastly, interaction with other vehicles could potentially cause issues in the validity of the experimentation results, because it is a direct violation of the stable unit treatment value assumption [33] - that the treatment only affects the individual sample and not anything else. The stable unit treatment value assumption is discussed and defined formally in the following section.

Non-functional requirements

There are strict non-functional requirements associated with automotive software, safety and legal requirements, among others. A software change in the functionalities governed by legal requirements might require the renewal of certain certifications that will reduce the speed of software delivery. The

safety requirements should not be violated from alternative variants for online experimentation, as the software would follow the same logic and go through the same deployment testing pipeline [3], [24]. With that being said, most automotive manufacturers would avoid the possibility of large scale disruption as even minor disturbances could directly cause profit losses for commercial vehicles such as taxis and trucks. Moreover, software changes often require explicit user consent due to privacy requirements. This combination of requirements could limit users in participating the experiment therefore leads to non-randomised user samples, selection bias, unobserved confounding effects, and the lack of generalisability in experimentation results.

Functional requirements

The last set of challenges in online experimentation adoption is related to the functional requirements of automotive software development. Not to be confused with functional requirements of experimentation platforms; in this subsection, we focus on requirements that are related to the software itself. In general, it is challenging to define a new automotive function especially when there is a lack of user data, much of the functional requirements are composed based on assumptions of how the vehicles is expected to be used. This poses a challenge for experiment design, as the treatment and target variables are often not clearly understood nor defined. In real-world experiment design problems, it is often assumed that the relationship between input and output is known. However that assumption might be false when we have no existing knowledge of the systems [34]; to maximise the values of online experimentation, a discovery of causal relation is required prior to conducting experiments.

In addition, the interaction between software systems within the same vehicle could pose a challenge in the design and analysis of experiments. There are well documented approaches in the web domain for handling conflicts in software changes, such as blue fonts shows on blue backgrounds on a website [32], [35], however, this complexity might be less well understood in the automotive domain. Suppose we aim to measure the performance of a battery preconditioning feature - prior to a trip, this system condition the battery to ideal operating temperature with a series of heat pumps. The purpose of such preconditioning is to firstly preserve the battery life through maintaining the battery system in ideal operating conditions, secondly to improve electrical

driving range during a trip as no extra energy is needed for conditioning the battery during driving, therefore, reducing auxiliary power consumption. This preconditioning software feature interacts with many other features, such as battery management, interior climate, charging, just to list a few. Setting up an aggressive heating strategy can improve battery life as it brings the battery into operating condition faster; however, it can potentially reduce energy efficiency, increase charging time, and lead to discomfort in the vehicle interior climate systems. Regarding expected performance improvement, locally optimising any feature alone without considering interactions with other software systems is never sufficient.

As a final point, from the complexity of the automotive product and the diverse operating environment, understanding the cause and effect between parameters with the same software feature and the interactions between software becomes difficult to build. It is often a process that require that rely on domain knowledge from the development organisations. The manual effort to a large extend hinders our ability to design experiments and scale the activities across organisations.

2.3 Potential outcome and causal inference

In principle, online experimentation aims to establish a causal relationship between treatment and effect and to model the expected outcome given the treatment applied. This process is often referred to as causal inference. The Rubin potential outcome framework describes such an expected outcome from experimentation [36] and observational study [20]. In this section, it is introduced in the context of online experimentation and when randomised experiments cannot be conducted. The theory of potential outcome and causal assumptions are defined formally.

Causality

Causality helps us reason with change [15]. Suppose that we observe a joint distribution of two factors $P(A, B)$, in this observation, the factors A and B are dependent or correlated. We can infer the conditional distribution of $P(A|B)$ or $P(B|A)$ through manipulation of the marginal and join probability distribution with Bayes' Theorem. However, the cause-and-effect between the

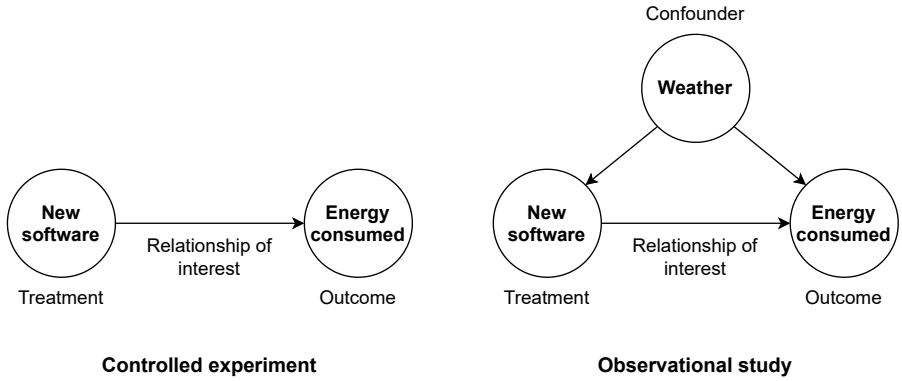

Figure 2.2: An illustration of relationships between treatment, outcome, and confounder in a controlled experiment and an observational study.

two factors A and B in our system remains unknown. In other words, in an observed marginal, join and conditional distribution, all three relationships are probable, $A \rightarrow B$, A causes B, $A \leftarrow B$, B causes A, or $A \leftarrow Z \rightarrow B$, a third confounder Z causes both A and B; therefore, a mere accumulation of observational data offers little insight into the notion of change. Similar reasoning is also presented in the literature, such as [15] and [34].

In a trivial example illustrated in Fig. 2.2, the relationship of interest is between a change in software and the total energy consumed in a vehicle. The software change is a treatment, $t \in T$, and the observed target variable is the energy consumption $y \in Y$. We would like to establish that T causes Y, $T \rightarrow Y$, and our objective is to estimate the average treatment effect (ATE) of the software change, for all samples $n \in N$, which is formally defined as the expected value differences from all levels of treatments. In a controlled experiment, where the control and treatment groups are randomly assigned, the confounding factors ($x \in \mathbf{X}$), often also called covariates, are automatically balanced from the randomisation process. The average treatment effect for a two-level experiment is the difference in the expected target variable given the treatment level $T = 1$ and the treatment level $T = 0$, formally,

$$ATE = \mathbb{E}[Y|T = 1] - \mathbb{E}[Y|T = 0] \tag{2.1}$$

Although ATE does not inform us about individual treatment effects, it offers a causal understanding of treatment with respect to the average effect of the two sample groups. Should the explicit assumptions be satisfied, the unbiased causal conclusion in a controlled experiment is straightforward provided the experiment is designed and conducted properly - all external factors are held constant and the only factor that could cause a change in the target variable, is the treatment we introduced. To establish such as relationship through observational study is more complicated due to the presence of confounding factors that are not controlled or even observed for, thus, requiring further assumptions and adjustments. A formal discussion is offered in the following two subsections. The explicit assumptions made in an experiments are listed below.

Stable unit treatment value assumption

The stable unit treatment value assumption, or SUTVA, states the potential outcome on one sample should be unaffected by the particular assignment of treatments to the other samples [33], [37].

Consistency

The consistency assumption states the potential outcome of a sample given the treatment received is the outcome that will actually be observed for that sample [38], formally,

$$T(n) = t \Rightarrow Y_t(n) = Y(n) \tag{2.2}$$

where T (n) represents the treatment to which a given sample (n) was exposed, $Y_t(n)$ is the possible result of the individual sample n being treated at level $T = t$, and $Y(n)$ is the observed outcome for this individual sample. Consistency is guaranteed in a controlled experiment by design, but in an observational study, it is not a testable assumption [39].

Exchangeability

To draw a factual (the observed outcome under treatment and control) and counterfactual (reasoning of what would have happened if a treatment is never applied) conclusion, randomised experiment is key. The randomisation pro-

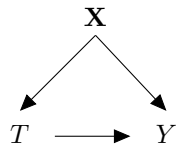

Figure 2.3: A simplified graph showing the relationships of treatment ($t \in T$), target variable ($y \in Y$), and confounding factors ($x \in \mathbf{X}$).

cess allows one to randomly sample from the total population, and ensures exchangeability of the control and treatment groups. Exchangeability explicitly states that the treatment outcome is independent of treatment assignment, thus, the treatment effect is caused by the treatment itself and not by some unobserved preexisting factors within the sample groups. In the dentition below, $\perp$ denotes independence, and $|$ means that given the condition, $[Y|T = 0]\perp t$ reads as the outcome Y given treatment $T = 0$ is independent from the treatment assignment T, formally,

$$[Y|T = 0], [Y|T = 1]\perp T \tag{2.3}$$

Exchangeability assumption is automatically satisfied through randomisation. Therefore, in a controlled experiment, correlation is causation. In a non-randomised study such as an observational study, the treatment assignment might be confounded on covariates thus violating the exchangeability assumption. Further assumptions are required for causal inference, i.e., conditional exchangeability.

Conditional exchangeability

In an observational study or a quasi-random experiment, there are confounding factors $x \in \mathbf{X}$ that influence the treatment and/or the target variable. These confounding factors are often called covariates, or contexts. To draw unbiased causal conclusions, the confounding factors need to be identifiable and observed, so that the conditional exchangeability assumption holds. The conditional exchangeability is a strong assumption that is not testable with observational data alone. Formally,

$$[Y|T = 0], [Y|T = 1]\perp T|\mathbf{X} \tag{2.4}$$

There are a few identification strategies for confounding factors X, such as propensity score [20], difference-in-differences [40], regression discontinuity [41], and instrumental variables [42], all of which made their own explicit or implicit assumptions in order to draw causal conclusions. As [38] states, causality cannot be inferred from observational data alone, behind all causal conclusion there exist causal assumptions.

Positivity

Positivity is an important assumption for the observational study, stating that all samples in both control and treatment groups are equally likely to be exposed to the treatment given the confounders; therefore, it requires that there be treated and untreated samples in every combination of the values of the observed confounding factors [43]. This assumption is testable in observational data through means such as visualisation of the confounding factors.

$$0 < P(T = t|\mathbf{X} = x|) < 1 \tag{2.5}$$

for all $x \in \mathbf{X}$ and $t \in T$.

2.4 Bayesian inference

As briefly mentioned in previous sections, Bayes' theorem is a set of statistical rules describing the relationship between marginal, joint, and conditional distribution. In this section, the concept will be introduced in detail and formally. Moreover, particular attention is paid to using the language of Bayesian statistics to describe causal inference.

Bayesian statistics focus on the inference of events based on prior knowledge that are related to the event, and the degree of belief is updated as more evidence is obtained. It describes the probability distribution of the event A given B, given our previous observation on related events A and B. When expressed mathematically,

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.6}$$

where,

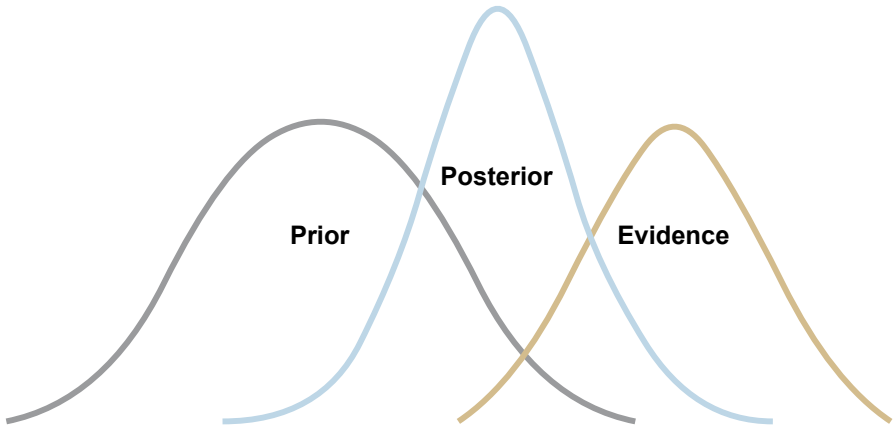

Figure 2.4: An illustration of Bayesian inference of prior, evidence, and posterior distributions.

$P(A)$ is the prior, the probability of a hypothesis before any evidence is observed and presented.

$P(B)$ is the likelihood, the probability of observing the evidence.

$P(B|A)$: is the probably of observing the evidence given the prior.

$P(A|B)$: is the posterior probability given the evidence, the observation, and the prior.

The first component of the Bayesian model is the prior probability distribution, $P(A)$, often referred to as domain knowledge, is an important aspect of Bayesian inference as it expresses the belief of an event before any observation is made and captures available knowledge. A prior can be determined from previous observation, experimentation, or can be elicited from subjective assessment of an experienced individual, threats to validity aside. A prior is described by an entire probabilistic distribution, and an informative prior expresses high confidence with low statistical variance [44]. The second component is the likelihood function $P(B)$, which is computed from the observed data. The combination of the prior and likelihood, the posterior probability distribution can be inferred. The posterior distribution, $P(A|B)$, informs the

expected outcome and, at the same time, provides information on the uncertainty of the said outcome. Their relation can be expressed graphically, as shown in Fig. 2.4. Note that the figure is an illustration and is not plotted from any data.

The language of Bayesian statistics is well suited in describing causal inference problems from observational study. Note that, usually the relationship in a classic Bayesian network does not imply causality, in this section, we use Bayesian language to describe a causal graph. Let us recall an observational relationship illustrated in Fig. 2.3 between treatment T, response variable Y, and, for simplicity, a single confounding factor X, suppose we conduct an experiment and all quantities in this graphical network are observed. Each node is represented by a probabilistic distribution that is either independent (with no edges pointing towards it) or dependent on other nodes, the relationship of which is specified with the directions of the edges. In this particular example, the probability distribution of the confounding factors X is simply $P(X)$, and $P(T|X)$ for the treatment. The probability distribution of the expected outcome is dependent on both the treatment T, and the confounders X, and it is expressed as $P(Y|T, X)$. The joint probability of event X, Y, and T is implied in the graph structure, and can be expressed as,

$$P(X, Y, T) = P(X)P(T|X)P(Y|T, X) \tag{2.7}$$

Suppose we now introduce a treatment $T = t$, the treatment effect can be inferred as the following. The dependency $X \to T$ no longer exists as we intervene in T, therefore, the term $P(T|X)$ is removed.

$$P(X, Y|T = t) = P(X)P(Y|X, T = t) \tag{2.8}$$

Bayesian causal inference has been applied in many area of science such as medicine[37], [45], transport engineering [46], and economic [47]. Compared to the equivalent models in the frequency domain, Bayesian causal models are reportedly less sensitive to sample sizes.

CHAPTER 3

Research objective and method

In this chapter, the research objective is presented together with the research methods applied in this research. The objective is broken down into three goals that this thesis is aiming to address. The research methods are chosen for achieving the goals provided there is a limited collection of literature documenting online experimentation in the automotive domain, we conducted case studies with our industry collaborators, as well as designed online software experimentation at scale for evaluating causal inference models such as Bayesian propensity score matching. In addition, threats to validity are discussed separately in terms of internal and external validity.

3.1 Objective

Following the success stories of online experimentation in software engineering, the automotive domain expressed interests in adopting the method to build more competitive software products. The objective of this doctorate research is to enable online experimentation in software engineering in the automotive domain given the challenges and limitations experienced in the domain. These challenges prevent the manufacturer from conducting online and randomised

experimentation altogether or they could invalidate the conclusion of experimentation, as violations of causal assumption could occur. To achieve this research objective, the following research questions are proposed.

RQ1 How to run large-scale online experiments in the automotive domain in a fast and reliable fashion?

RQ2 How can the inherent limitation of sample sizes in the automotive domain be addressed?

RQ3 Can causality be concluded in the absence of randomisation in a software online evaluation?

RQ1: How to run large-scale online experiments in the automotive domain in a fast and reliable fashion?

The first research question is formulated with the aim of defining the necessary components and actions of conducting online experimentation on a scale in the automotive sector, given the understanding of the challenges of online experimentation adoption in automotive, the state-of-the-art in online experimentation approaches, and the current state-of-practise in the sector.

Online experimentation has been extensively applied in the SaaS domain, such as the large-scale A / B tests reported by [32], [35]. Moreover, the existing approaches pay special attention to scalability [7], [8], [29], [48], speed of deployment [28], [29], [49], and the reliability of the results [7], [8], [19], [28]. However, none of the existing approaches addresses the shortcoming from the nature of embedded software, as the experimentation models themselves do not automatically address issues such as deployment speed and data collection. The need of having a functional infrastructure, pipeline, and even data-driven decision making mentality in place is highlighted by majority of the literature documenting online experimentation in the SaaS even in the embedded domain [3], [23], [50].

Moreover, this research question is formulated with emphasis on the core component of experimentation, randomisation. For an experiment to be reliable, it requires that the control and treatment groups are randomly allocated to ensure the causal assumption of exchangeability is satisfied [15], [20], [37]. With this research question, our aim is to understand whether randomisation

can be achieved in practise and on a large scale without compromising safety and privacy in the automotive setting.

RQ2: How can the inherent limitation of sample sizes in the automotive domain be addressed?

Compared to the SaaS domain, where the lack of samples is sometimes reportedly problematic [16], [21], the automotive domain has significantly fewer user bases available for experimentation, by orders of magnitude. Netflix reported around 214 million active users in 2021, while Volvo Cars reported a sales volume of just shy of 700,000 cars. Moreover, due to the high number of product variants in combination with a wide range of operation environments, without running experiments on a large scale, it is safe to assume that the sample size will be an inherent limitation for the adoption of online experimentation.

First, when inferring the effect of a treatment in a randomised experiment, a small sample size could cause a less sensitive measured treatment effect [16]. This means that the difference between the treated and untreated samples is too small to conclude a significant change statistically speaking, therefore, invalidating the experiment result. Second, when assigning control and treatment groups at random from a limited sample population, there is a high probability that the covariates of the groups are not balanced, creating bias and introducing a confounding effect, therefore hindering the conclusion of causality [17], [51]–[53]. Third, the datasets from automotive domain are low in sample size, but high in data feature dimension. For evaluation of a software function, hundreds of parameters could be collected in relation to the evaluation. Gaining insights into which parameters are essential at expressing software performance improvement, is challenging in a sample size efficient manner. The research question RQ2 is drafted to cover all three aspects as a result of the sample size limitation.

RQ3: Can causality be concluded in the absence of randomisation in a software online evaluation?

Causal insight is achieved through randomisation and intervention. RQ3 is formulated to address situations in which a randomisation is not an option due to limitations manifested from the non-functional requirements of automotive software, i.e., strict safety requirements, and the lack of user privacy and

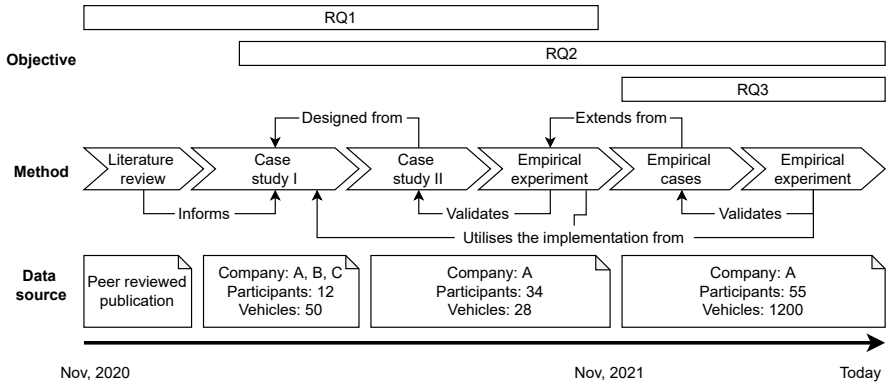

Figure 3.1: A timeline showing the research questions, research methods, and the corresponding activities.

software update agreements. This dictates that we are only able to narrowly access a subset of users instead of the total population of vehicles. As a result, instead of a fully randomised experiment, automotive software online evaluation is likely to be done in a quasi-random or even an observational manner without randomisation at all.

Behind every causal conclusion lies an assumption, and each causal inference model has built-in specific assumptions [17], [20], [40], [41], [54]. We aim to understand the assumptions in relation to scenarios experienced in the automotive domain, such as seasonality effects from vehicle operational conditions. Furthermore, this type of analysis can allow us to infer the treatment effect *post facto*, which can be a valuable approach in a domain where there are a large amount of observed data but the ability to experiment is limited.

3.2 Method

In general, there is a lack of publications documenting online experimentation practises in the automotive sector. In this thesis, we combine case studies with industry partners and empirical experiments conducted on a small- to medium-sized scale to validate our hypotheses. In addition, to study the state-of-the-art in online experimentation practises in the SaaS domain, for Paper A, a literature review on experimentation architecture is conducted compliment-

ing with a case study conducted with three automotive manufacturers. The literature review aims to inform the case study with the state-of-art published within and outside of the automotive domain. After identifying the necessary components in enabling a continuous online experimentation through a case study, case study II is designed aiming to address an inherent limitation in the domain, sample size. Combining a case study with an empirical experiment, a control/treatment group design method is applied and evaluated in practise, utilising the software architecture implemented in place from case study I. The same software architecture enabled the follow-up empirical experiment on a much larger scale, which is designed to address and answer RQ3. The research questions, methods addressing the questions, and their corresponding activities are summarised in Fig. 3.1.

Literature review

To gain a good understanding of the existing online experimentation practises in the embedded and SaaS domains, we conduct a literature review. A literature review is an important process in accessing and analysing relevant information within a field of science from a large collection of publications. It is considered a key practise in evidence based software engineering [55], as it allows one to understand the state-of-art reported in the domain and with that, identify the potential knowledge gap and improvements to be made. Since 2004, There is an increase of number of publications applying literature view as a research method in software engineering [56]. In our research, a literature review is conducted primarily for building an understanding of the existing online experimentation framework and/or software architecture, and their applicability in the automotive domain.

In this study, data is collected through IEEE Xplore, ScienceDirect, and Google Scholar with the following query, ("A/B testing" OR "A/B experiment" OR "online experiment" OR "bucket testing" OR "continuous experiment") AND ("software architecture") AND ("embedded software" or "automotive software"). However, keyword combination with automotive software return no results, therefore, it was removed at a later stage. This query returns a total of 104 results excluding duplicates and the search is filtered over a time frame of 2010 to 2020. Each search result is examined by at least one of the authors by reviewing the keywords, abstracts, and body of the text. Publications that focus on describing software architecture or online experimentation

framework are included.

Case study

Software case study is a well suited method for studding adoption of new development methods in the natural context of a software company [57]. Case study method is suited for investigating a novel development approach, such as online experimentation, in context within the industry that is otherwise difficult to investigate in an isolated fashion. In Paper A, our study has an exploratory nature, that is to build understanding of the state-of-practise, propose architectural solutions, and analyse and evaluate the solution close to the end working environment that is the automotive domain. We understand and are aware of the limitations of the case study as a research method in software engineering, such as that it does not allow one to draw causal conclusions between the change and the response; however, in our application, it is mainly a means of exploring in the face of limited publications on the subject matter. A version of our software architecture is deployed to a fleet of 50 vehicles.

In Paper B, a case study is carried out following the guidelines [58] and [59] in combination with a quantitative analysis. Guided by careful research design and extensive planning with our industry collaborator, we planed, deployed, executed, and analysed an online experiment at scale with an automotive company *in situ* for a total of six month, during which we work closely and directly with the software development team. In addition, we conducted an experiment on a vehicle fleet of 28 cars. The quantitative data are continuously collected and used for evaluation of the experiment design method. All participants of both case studies are members of the software development teams, including software engineers, data scientists, product owners, team managers, and technical experts.

Case study companies

In this thesis, we present case studies results from three separate automotive company, all of which are multinational companies based in Sweden. Company A and C are passenger vehicle manufacturers, and company B a commercial vehicle manufacturer and operator. All three companies participated in the study under non-disclosure agreements; thus, a brief description is provided.

Company A is an original equipment manufacturer for passenger vehicles and a multinational company with most of its operations in Europe, Asia, and North America. In recent years, company A is transitioning from an equipment manufacturer to a service provider aiming to provide mobility as a service and similar products. This company also has the ambition of becoming a software-orientated business to remain competitive in the rapidly changing electric vehicle landscape.

Company B is an original equipment manufacturer of commercial vehicles with its business covering truck, bus, construction, heavy duty equipment, and maritime engines. With fleet management as part of their service, the vehicles produced by company B are connected and rely on innovative software solutions. Company B is a multi-national company with a head office in Sweden. Company C is a design and development consultancy for personal passenger vehicles; they develop and design passenger vehicles, software solutions, and mobility products for different vehicle brands.

Case study participants

In Paper A, in order to gain a good insight on the current development of online experiment adoption, we conducted interviews with twelve employees from company A, and five each from company B and C. Their roles include software engineer, software architects, product owner, data engineer, and data scientist, working with some aspects of online experimentation. All interviews are conducted by least one of the authors and the response of the participants are documented as meeting notes.

In Paper B, we actively worked with the development team of company A *in situ* for a period of six month; this development team consists of 34 members, including software developers, data scientists, and product owners. At least one of the authors organised and participated in the weekly project meeting on the research topic throughout the period. All the research activities with all case study companies are organised and managed by the main author of the publications.

Empirical Experiment

The research approach for Paper B, C and D is based on empirical experiments; the experiments are conducted with an industry collaborator (company

A) on a fleet of 1,200 vehicles. This is a quantitative approach which allows the highest possible level of repeatability and generalisability. We deploy software treatments on a small- to medium-scale in a controlled manner, collect user data for a continuous four-month period, and use the empirical data collected as input to the causal models.

Compared to a simulated dataset, the use of empirical datasets allows us to evaluate causal inference models in a real-world setting. We can further evaluate the validity of the model by adapting to stochastic noise and the applicability of the model when predictability is low. Moreover, in running small- to medium-scaled online experiments, by introducing theoretical models to industrial practises, hope that our results could inspire other researchers and manufacturers to adopt online experimentation as a prominent approach in software engineering.

Data collection

The total data collection period takes place between October 2020 and December 2021 for a total of four datasets from 1,200 unique vehicles in total. Using on-board sensors and existing telecommunications modules on the vehicles, measurements are done in a continuous fashion at 10 Hertz and sent to a centralised server for remote access. After post-processing, the data is collected from 493,187 trips. The vehicles participating in the empirical experiment and receiving the software treatments are driven by employees of the case company A. They are company cars leased to the employees with special agreements to allow such a software change. The vehicles' users are informed of the software change, however, not the details of the specific changes. All data collection is done anonymously, and no metadata or data can be reconstructed to identify drivers or vehicles.

The data feature generation from raw measurements is done together with the development teams, and this process takes into consideration the physical properties and limits of vehicles, such as the average trip speed shall never exceed the rated top speed of the vehicle. First, all data features are generated from at least two or more measurements to triangulate the validity of the sensor signals. Second, measurements generated from vehicles that are brand new (with mileage under 100 kilometres) are discarded. Third, the data features are aggregated on the user level, as we are interested in the treatment effect on individual vehicles.

3.3 Threats to validity

In this section, threats to validity are discussed first in terms of internal validity, and external validity. For each threat to validity, the mitigation strategy is presented in the corresponding subsection.

Internal validity

As a type of internal validity, content validity refers to the degree to which the elements in a study are representative of the domain that the study seeks to measure. The most common threat to content validity is incomplete content, that is when the study does not cover all aspects of the subjects that it aims to measure. Our mitigation strategy is to iterate the study design amongst the researches involved, to ensure that the relevant aspects to the research questions are covered.

Criterion validity measures how well a test score predicts real-life outcomes, for example if we measure the line-of-code produced by a software engineer, are we able to predict this engineer's future work efficiency. In this thesis, no similar predictor is hypothesised, therefore, we do not consider criterion validity and its related threats.

Construct validity addresses the concerns of whether the measures accurately access what they are supposed to, in other words, whether the selected measurements articulate the concepts of the research questions. In a qualitative research, a potential threat to construct validity could be that the interviewer and the interviewees understand the terminology differently [57]. To mitigate such a threat in our studies, we exclusively interview people with experience in online experimentation, and we distribute definitions to terminologies that are critical to our interview questions prior to the interviews.

External validity

External validity refers to which extend can the conclusions be generalised to other setting within the domain. Since the research has been conducted with a limited numbers of companies, and with limited datasets, generalisability is not a given. However, the ecological aspect of the external validity should be well addressed, as all of our research is conducted empirically and their findings are derived from real-world software engineering scenarios.

Population validity - whether the findings are generalisable to a wilder context of population, in this case, to companies in the automotive domain which we did not conduct study with. This aspect is addressed in the following ways. First, in qualitative research, results are only included when the same perspective is mentioned by people from two or more organisations. Second, we cross validate our results, from both qualitative and quantitative research, with literature from within the domain and other area of science, to make sure no spurious results are found. Last, the automotive sector is a standardisation focused domain, majority of the companies deploy similar development processes and experiencing similar issues [60], thus, the conclusions derived from our empirical experiments should be generalisable to a large extend to other automotive companies.

CHAPTER 4

Summary of included papers

This chapter provides a summary of the included papers.

4.1 Paper A

Yuchu Liu, Jan Bosch, Helena Holmström Olsson, Jonn Lantz
An architecture for enabling A/B experiments in automotive embedded software
 DOI: 10.1109/COMPSAC51774.2021.00134 .

A/B experimentation is a known technique for data-driven product development and has demonstrated its value in web-facing businesses. With the digitalisation of the automotive industry, the focus in the industry is shifting towards software. For automotive embedded software to continuously improve, A/B experimentation is considered an important technique. However, the adoption of such a technique is not without challenge. In this paper, we present an architecture to enable A/B testing in automotive embedded software. The design addresses challenges that are unique to the automotive industry in a systematic fashion. Going from hypothesis to practice, our ar-

chitecture was also applied in practice for running online experiments on a considerable scale. Furthermore, a case study approach was used to compare our proposal with state-of-practice in the automotive industry. We found our architecture design to be relevant and applicable in the efforts of adopting continuous A/B experiments in automotive embedded software.

4.2 Paper B

Yuchu Liu, David Issa Mattos, Jan Bosch, Helena Holmström Olsson, Jonn Lantz
Size matters? Or not: A/B testing with limited samples in automotive embedded software
©IEEE DOI: 10.1109/SEAA53835.2021.00046 .

A/B testing is gaining attention in the automotive sector as a promising tool to measure causal effects from software changes. Different from the web-facing businesses, where A/B testing has been well-established, the automotive domain often suffers from limited eligible users to participate in online experiments. To address this shortcoming, we present a method for designing balanced control and treatment groups so that sound conclusions can be drawn from experiments with considerably small sample sizes. While the Balance Match Weighted method has been used in other domains such as medicine, this is the first paper to apply and evaluate it in the context of software development. Furthermore, we describe the Balance Match Weighted method in detail and we conduct a case study together with an automotive manufacturer to apply the group design method in a fleet of vehicles. Finally, we present our case study in the automotive software engineering domain, as well as a discussion on the benefits and limitations of the A/B group design method.

4.3 Paper C

Yuchu Liu, David Issa Mattos, Jan Bosch, Helena Holmström Olsson, Jonn Lantz
Bayesian propensity score matching in automotive embedded software engineering
©IEEE DOI: 10.1109/APSEC53868.2021.00031 .

Randomised field experiments, such as A/B testing, have long been the gold standard for evaluating the value that new software brings to customers. However, running randomised field experiments is not always desired, possible or even ethical in the development of automotive embedded software. In the face of such restrictions, we propose the use of the Bayesian propensity score matching technique for causal inference of observational studies in the automotive domain. In this paper, we present a method based on the Bayesian propensity score matching framework, applied in the unique setting of automotive software engineering. This method is used to generate balanced control and treatment groups from an observational online evaluation and estimate causal treatment effects from the software changes, even with limited samples in the treatment group. We exemplify the method with a proof-of-concept in the automotive domain. In the example, we have a larger control ($N_c = 1100$) fleet of cars using the current software and a small treatment fleet ($N_t = 38$), in which we introduce a new software variant. We demonstrate a scenario that shipping of a new software to all users is restricted, as a result, a fully randomised experiment could not be conducted. Therefore, we utilised the Bayesian propensity score matching method with 14 observed covariates as inputs. The results show more balanced groups, suitable for estimating causal treatment effects from the collected observational data. We describe the method in detail and share our configuration. Furthermore, we discuss how can such a method be used for online evaluation of new software.

This paper was awarded the Best Paper in Software Engineering in Practice at APSEC 2021.

4.4 Paper D

Yuchu Liu, David Issa Mattos, Jan Bosch, Helena Holmström Olsson, Jonn Lantz
Bayesian causal inference in automotive software engineering and online evaluation
.

Randomised field experiments, such as A/B testing, have long been the gold standard for evaluating software changes. In the automotive domain, running randomised field experiments is not always desired, possible, or even ethical. In the face of such limitations, we develop a framework BOAT (**B**ayesian

causal modelling for **O**vserv**A**tional **T**esting), utilising observational studies in combination with Bayesian causal inference, in order to understand real-world impacts from complex automotive software updates and help software development organisations arrive at causal conclusions. In this study, we present three causal inference models in the Bayesian framework and their corresponding cases to address three commonly experienced challenges of software evaluation in the automotive domain. We develop the BOAT framework with our industry collaborator, and demonstrate the potential of causal inference by conducting empirical studies on a large fleet of vehicles. Moreover, we relate the causal assumption theories to their implications in practise, aiming to provide a comprehensive guide on how to apply the causal models in automotive software engineering. We apply Bayesian propensity score matching for producing balanced control and treatment groups when we do not have access to the entire user base, Bayesian regression discontinuity design for identifying covariate dependent treatment assignments and the local treatment effect, and Bayesian difference-in-differences for causal inference of treatment effect overtime and implicitly control unobserved confounding factors. Each one of the demonstrative case has its grounds in practise, and is a scenario experienced when randomisation is not feasible. With the BOAT framework, we enable online software evaluation in the automotive domain without the need of a fully randomised experiment.

CHAPTER 5

An architecture for enabling A/B experiments in automotive embedded software

The layout has been revised.

5.1 Introduction

A/B experimentation or A/B testing is a method for evaluating software changes in a quantifiable manner. Continuous A/B testing is an important method in understanding and delivering measurable customer value. Many web-facing companies have demonstrated success from A/B experiments, such as Booking.com [18], Google [35] and Microsoft [28], [32], [48], just to list a few. With the digitalisation of the automotive industry, software is becoming a main differentiator of products [2]. A/B testing is an effective tool to evaluate software and support organisations in making data-driven decisions [61]. However, the adoption of continuous A/B experiments in automotive embedded software is not without challenges.

Embedded software has hardware constraints. Such constraints could manifest as limitations to computational power [24], long release cycles [2] and often

dependency on suppliers [3]. Data collection and handling is also believed to be challenging in the automotive specific applications [3], [12]. Although a fair number of publications point out the challenges in A/B experiment adoption [2], [3], [12], [24], we identified a gap in the literature concerning architectural solutions to enable A/B experiments. Furthermore, there is little to no reports on concluded or ongoing online A/B experiments in the automotive domain.

In this paper, we present an architecture that enables A/B experiments in the automotive domain and aim to address the challenges that are unique to this industry. We present a literature review of A/B experiment architecture in embedded and web-facing environments. Moreover, we conducted a case study of the architecture applied at scale and to report the state-of-practise of A/B testing in automotive. Compared to the existing literature, the contribution of this paper is two-fold. First, we present an architecture that enables A/B testing automotive software. We reviewed the literature and did not find a similar architecture for A/B experiments. Secondly, we apply this architecture in practise, in fleets of considerable scale. We present the case study and state-of-practise of two other automotive companies.

The rest of this paper is organised as following. In Section. 5.2, we introduce the unique constraints in automotive industry for A/B testing. In Section. 5.2, we present our research method. We summarise the existing A/B experiment frameworks and architecture in Section. 5.2. In Section. 5.5, we present our architecture design along with the case studies. Discussions and conclusion are presented in Section. 5.6 and 5.7.

5.2 Background and constraints

In this section, we introduce the background on A/B testing and list the constraints of adopting the method in automotive embedded software.

Background

A/B testing is a type of continuous experimentation where users or systems are split into subgroups and issued with different variants of the same software. By studying the response from each cohorts, A/B experiments can guide product development in an effective manner [18], [32], [35]. Typically, eligible users are split into two groups, the A version (control) and the B version

(treatment). For both user groups, their interactions with the functions are recorded and evaluated based on a set of carefully designed metrics reflecting business and/or customer values [62].

Almost all well-established A/B testing frameworks are for web-facing businesses. Such frameworks or models cannot be applied directly in an embedded environment as they do not address specific challenges. These challenges come from many aspects, they can be technical, business, and organisational as demonstrated by Mattos *et al.* [2]. As embedded software often has dependency on hardware, fast software release becomes difficult to accomplish [3], [12], [24]. Although challenging to adopt, many advantages of continuous experiments that were proven in the web-facing businesses are also expected in the automotive industry [12].

Constraints

In addition to the challenges summarised by relevant literature [2], [3], [12], we list the specific constraints in automotive which motivate our architecture design. Automotive embedded software is distributed to hundreds of Electronics Control Modules (ECUs). These software are traditionally developed using the "V-model" where the OEMs deliver specifications and suppliers deliver implementations [4]. This model has exhibited its limitations.

Release cycles and speed

Combining the strict standards with the growing complexity, the automotive software release process is rigid. First, the development and release of automotive embedded software is usually strongly dependent on suppliers. Secondly, automotive companies have traditionally designed software release cycles based on their hardware release process [50]. This process cannot handle rapid changes, as all integration and tests are planned at fixed periods. Moreover, the most commonly adopted automotive software architecture AUTOSAR [1] lacks flexibility in partial updates [3]. If the new software is not backwards compatible, all ECUs in the vehicle need to be updated. Last but not least, updating software which are governed by legislation might require renewal of certifications, which will add delays to the software release process.

[1] autosar.org

Sample size and management

Controlling boundary conditions is impossible for online experiments, as vehicles can be driven to everywhere and at anytime. Therefore, to conclude sufficient treatment effects, A/B experiments need be conducted on large and randomly selected sample groups. This large group of users needs to be managed as online experiments require a flexible configuration of A/B or A/B/n groups. However, the sample groups are difficult to manipulate when the software needs to be updated through physical contact with the cars. Same challenge could be experienced when an A/B test is concluded, and the software needs to be inverted to the original version.

Managing sample groups longitudinally can be burdensome. Performance of some automotive functions depends on temporal factors and has seasonality effects, thus experiments need to be conducted longitudinally. Therefore, the ability to orchestrate the A/B groups over time is beneficial.

Data infrastructure

To conclude a casual effect of the treatment, data collection for A/B experiments requires certain level of accuracy. Storing such data locally in each vehicle is not feasible, as it becomes difficult to access and it will require a large memory on-board. The success of an A/B experiment is largely relied on appropriate assumptions when designing an experiment and fast feedback when conducting one. Sharing data within a large organisation can be problematic [14]. In order to maximise the data, all development teams need to have easy access to relevant data. As a result, companies suffer from misrepresentation of customer values.

Safety requirements and fallback

Automotive software has high safety requirements. In an A/B test, all alternative versions can never obstruct such requirements which might affect road safety and/or legal compliance. The functional requirements need to be safeguarded while ensuring a continuous release of alternative versions seems impossible today. Another practise to decrease hazards on the road is to have built-in fallback for safety critical functions. For instance, one could install both the A and B alternatives on-board. Then the A alternative can be used as a fallback when it is thoroughly tested and validated.

5.3 Research method

In this paper, we combine a literature review with case studies. We studied several existing A/B experiment frameworks inside and outside of the industry through literature reviews, to compare our approach to existing frameworks. Furthermore, to validate the architecture designed, we conducted case studies based on a series of ongoing efforts in A/B experiments from three separate automotive manufacturers.

We explore the following research question:

RQ1 What are the existing software architecture for online experimentation?

RQ2 What are the requirement for enabling continuous experimentation with automotive embedded software?

Literature review

This literature review is done to understand existing A/B experiment frameworks within and outside of the automotive domain. To identity and explore work that is relevant for the research question, we follow the methodology described by Kitchenham [63].

Data collection

We included the following terms in our search query: ("A/B testing" OR "A/B experiment" OR "online experiment" OR "bucket testing" OR "continuous experiment") AND ("software architecture") AND ("embedded software" or "automotive software"). Alternative terms are included as there is no standard terminology. Keyword combination with "automotive software" yield no meaningful results, thus we expanded the search query to also include embedded software. The databases included in our search process are IEEE Xplore, ScienceDirect, and Google Scholar, returning a total of 104 results excluding duplicates. To ensure the results are relevant today, we limit the publications to the recent ten years.

Table 5.1: Papers selected which describing architecture of A/B experiments, for web and embedded software).

No.	Title of publication	Authors	Year
P1	Overlapping Experiment Infrastructure: More, Better, Faster Experimentation [35]	Tang *et al.*	2010
P2	Architecture for Large-Scale Innovation Experiment Systems [23]	Eklund & Bosch	2012
P3	Eternal Embedded Software: Towards Innovation Experiment Systems [50]	Bosch & Eklund	2012
P4	Beyond data: from user information to business value through personalized recommendations and consumer science [49]	Amatriain	2013
P5	Online controlled experiments at large scale [32]	Kohavi *et al.*	2013
P6	Design criteria to architect continuous experimentation for self-driving vehicles [24]	Giaimo & Berger	2017
P7	The RIGHT model for Continuous Experimentation [29]	Fagerholm *et al.*	2017
P8	Experimentation growth: Evolving trustworthy A/B testing capabilities in online software companies [18]	Fabijian *et al.*	2018
P9	The Anatomy of a Large-Scale Experimentation Platform [48]	Gupta *et al.*	2018
P10	Scalable Data Reporting Platform for A/B Tests [64]	Vasthimal *et al.*	2019
P11	Experimentation in the Operating System: The Windows Experimentation Platform [28]	Li *et al.*	2019

Inclusion criteria

Each paper resulted from the search process was reviewed by at least one of the authors. We examine the keywords, abstracts, and the body of the paper to identify A/B experiment frameworks and the applicable sector for said frameworks. We selected publications which focus on A/B experiment architecture and/or framework from embedded applications. We did not include publications discussing the benefits or challenges or feasibility of A/B testing. This inclusion criteria resulted in a total of three papers. Since the technique is well established in web-facing applications, we included work on A/B test-

ing framework in the web domain. A total of 11 publications included in this review are listed in 5.1.

Case study

Following guidelines from Runeson and Höst [57], we conducted two sets of case studies with three separate automotive companies. In study I, we examine the proposed architecture in practise on a cloud-based A/B experimentation in a vehicle fleet at scale. We study the architecture for A/B testing in a fleet from one of the three companies. The software for case study I was developed in-house in company A. As online experiments are not commonly applied in the industry, to the best of our knowledge, there is a lack of quantitative data to study from. To understand the state-of-practise, we conducted semi-structured interviews with two more OEMs as case study II.

Case study attendees

The three companies included in the case studies are large OEMs. In each company, we conducted interviews and workshops with at least five different employees from each company, working with varying aspects of software development. Their roles include software engineer, software architect, product owner, data engineer and data scientist.

Data collection

One of the authors was actively involved in the experimentation design from ground up and supported the entire process. We document the process through meeting notes and design specifications in the project. The questions from case study II were specifically designed to understand the current state-or-practise of A/B experiments in an automotive setting. We also aim to understand the potential of cloud-based A/B testings in each company. During the interviews, we presented our architecture design to the attendees along with questions regarding current practises adopted in their companies. All the interviews were conducted by at least one of the authors. The responses were documented as meeting notes, which were distributed to the interview participants.

We recognise the limitation of our case study approach, as the results of our case studies were obtained from three companies. The outcome can be specific

to these companies and without further investigation, we cannot generalise the conclusion to the automotive industry.

5.4 Existing architectures

In this section, we present the results from our literature review. We included 11 publications [18], [23], [24], [28], [29], [32], [35], [48]–[50], [64] that focus on describing A/B experiment architectures, in both embedded software and online applications. From our literature review, we have discovered that there is a general gap in the literature on architectures or frameworks designed specifically for automotive software. Based on the topic, we summarise the papers into four overlapping categories. They are grouped firstly by their environment, i.e., embedded or web-facing. Paper [28], [29] are applicable for both groups. We include OS and embedded applications in the same category, as they share many common challenges for instance, the devices can be offline [28]. Second, we identified in these papers how a software variant is shipped to the users. Namely, if a complete software change is required, or variants can be introduced through parameter changes. The categories are presented in Figure 5.1. As can be seen, variant introduction through parameter change is not a widely explored method within embedded software.

Although the design process is vastly different, there are a numbers of shared components for embedded and web experiment architecture. This includes experiment configuration, data collection, experiment analysis and metrics evaluation. Therefore, some experiment models can be employed in various environments including web, operation systems and embedded [29], [48]. Tang *et al.* [35] and Kohavi *et al.* [32] both report a multi layered experiment configuration system that can handle multiple A/B experiments. Users will be assigned to A or B variant in a consistent manner [48], [64]. In the web environment, this is achieved by assigning unique IDs when users visit the web pages.

Data infrastructure is a major component in any experiment framework. All researchers include data infrastructure as part of their experiment frameworks, particularly focused on trustworthiness [18], [29], [32], [35], [48], [64]. Such data collection is also required in embedded environments, however, is more difficult due to hardware limitations [50]. An experiment architecture [23] for automotive software used an on-board data storage before uploading the data

		A/B experiment environment	
		Web	Embedded/OS
Variant generation	Software change	P4, P7, P8, P9, P10, P11	P2, P3, P6, P7, P9, P11
	Parameter change	P1, P5	

Figure 5.1: Existing A/B experiment framework categorised by environment and variant generation methods.

through the vehicle's telemetry.

Another key element for A/B experiments is rapid software release. We found that all architectures for rapid experiments in an embedded environment rely on continuous deployment. The "RIGHT Model" discuss that if a function is novel, continuous deployment might not be necessary [29]. However, most frameworks in embedded environments [23], [24], [50] require a well-established continuous deployment process to achieve rapid experimentation loops. Software variant release through Over-the-air(OTA) can increase delivery speed in automotive applications [23]. In the web environment, rapid experimentation can be achieved more flexibly through an array of mechanisms. For example, an offline and online experiment systems in Netflix, as demonstrated by Amatriain [49]. Existing data can be used to train the models before they are introduced to an online experiment, which allows faster and cheaper evaluation of software. Another technique in increasing experiment speed is using parameter updates as mentioned by Tang *et al.* [35]. The A/B variants in target functions are parameterised and configured through data files. These parameters are changed more frequently than code, which enables fast experiments provided the parameters exist.

Furthermore, to fully utilise the benefits of A/B testing, all papers highlighted the importance of the organisational and cultural mindset of making data-driven decisions. Kohavi *et al.* [32] summarise prerequisites which an organisation needs to adapt, highlighting the importance of data-driven decision

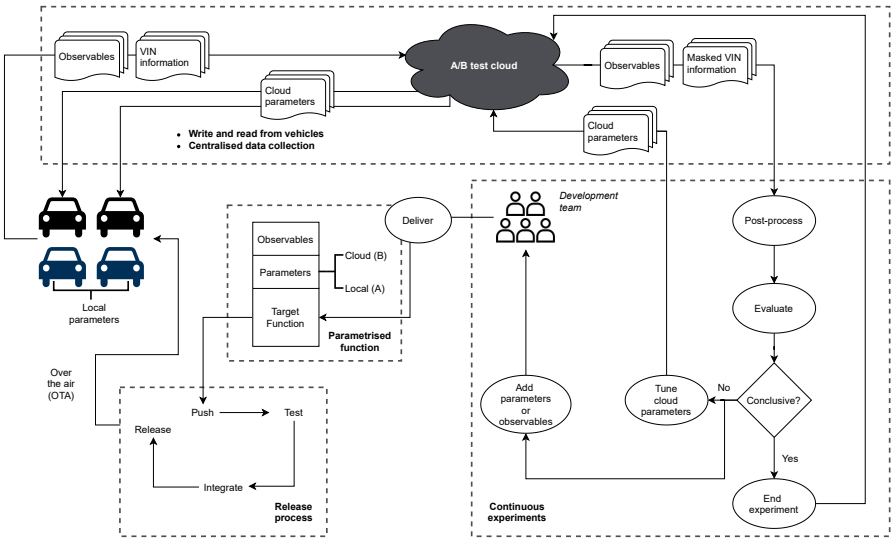

Figure 5.2: Process of the cloud-based A/B test architecture, illustrating the general work flow of conducting an A/B experiment with parametrised functions.

making mindsets. In the "Experiment Growth Model" introduced by Fabijan *et al.* [18], all components of their A/B experimentation model become more mature as the entire organisation evolves through different stages.

5.5 Architecture and case studies

In this section, we present a software architecture that could enable A/B testing in the automotive domain. A hybrid architecture is presented. The essence of the architecture is to imitate an online environment for an otherwise offline application. In doing so, automotive A/B testing can benefit from the flexibility of online experiments We present the components of our architecture in Fig.5.2.

System requirements

Our architecture enables continuous automotive A/B experiments and follows a set of requirements. The architecture needs to respect the boundary conditions of automotive domain and follow the principles of continuous experiments. We list these requirements in two groups, technical and non-technical. The technical requirements are:

- The A/B experiments rely on fast releases, continuous integration and continuous deployment processes. The architecture should satisfy the property of a continuous deployment alike environment while not conflicting with the release processes that are in place.
- The experiments can be managed in an easy manner. This includes managing eligible users in control and treatment groups, tracking their status over time, configuration of experiments if/when there are several different treatments to be tested and ability to terminate an experiment once a conclusion is reached.
- To conduct successful experiments, data quality is key. The signals to be measured and their sample frequency should be specified accordingly to the function of interest. We call these signals observables. Data should be accessible to the development and analysis teams within the company through standardised pipelines and compatible with standardised analysis tools.
- Alternative software variants running in the vehicles should have a built-in safety fallback although it should always preform to standards. Furthermore, the architecture should follow automotive security standards, so that no unauthorised modification can be done to the software remotely or otherwise.
- The architecture should be scalable in terms of number of experiments that can be conducted in parallel, and number of eligible target users. This means, ideally any function can be tested on any group of users.

The non-technical requirements are:

- The A/B experiment architecture should promote decision autonomy on the team level. Instead of a top-down approach, the development

teams should be encouraged to explore functional changes freely and take data-driven decisions from the outcome of their A/B experiments.

- There should be a high level of transparency and collaboration in between teams, in particular when functions have dependency on each other. By providing a common platform for configuring and analysis of experiments, this architecture can potentially improve collaboration between teams.

System characteristics

We present a hybrid architecture (Fig.5.2) combining on-board and cloud functionalities. The system is composed of six main components. These are parameterised functions, a release process which most companies have in place. There is a cloud host that writes parameters to the vehicles and collects data from the vehicles. Finally, a centralised data storage and pipeline for distributing the measured data.

The system workflow can be described as follows. First, a function which characteristics can be defined by a list of parameters is delivered. There are two sets of parameters embedded in the function, the local set, which is the default and the cloud set that can receive incoming values externally. The benefit of parameterisation in A/B testing was also highlighted by [35]. A set of observables which measure function performance is also predetermined. The function and its parameters are delivered to a release process which will integrate with other functions and release the software to vehicles. This release and installation of software can be done through workshop visits or OTA.

Once the software is introduced to the vehicles, users are identified through Vehicle Identification Numbers (VIN), which is unique and comprised of vehicle meta-data. This ensures that although the software is introduced to all cars, no experiments will be conducted unless the users are deemed eligible in advance. Upon key-on of a vehicle, a vehicle will send its VIN to the A/B test cloud. Since the A and B groups are configured in the cloud, the test cloud will match the VIN and then return a status indicator to the vehicle. Ineligible cars will have no match in the cloud and receive no response. For all eligible vehicles, they can be partitioned into A and B groups through remote configuration. The control group will use the functions' local parameters and the treatment group will receive cloud parameters. Since the parameter names

are predefined, the vehicle cannot accept any other values, thus increase security. Furthermore, as the cloud parameters are blank values in the vehicles, cloud parameter change can be done remotely. This design enables function behaviour change through parameters provided the parameters exist. Development teams can continuously A/B test and adjust the existing parameters without complete software change and independently from the company-wide release cadence. A complete software update is required only when new parameters need to be added.

Data collection is done through the cloud and it measures a set of predefined observables. The observables are measured and temporarily stored on board, then sent to the cloud at time intervals while driving. This data are collected in a centralised data lake, cleaned, then distributed to development teams. During a trip, time series data is collected for dynamic observables. For stationary observables, only one or a few snapshots are measured. After analysis of the A/B tests, further actions can be taken such as adjusting cloud parameters, re-partitioning A/B groups, or concluding the experiments. When the experiments are concluded, the connection to the cloud will be interrupted and vehicles will invert back to the local parameters automatically. Moreover, the local variant always serves as a safety fallback in critical situations.

Case study I

The first case study was performed in company A on an energy management (EM) function that was developed internally. The function Energy Management has a local and a cloud set of parameters which determined the local and the cloud energy management strategy, respectively. The local strategy is already available in vehicles of the same variants which company A sells in a given market. By default, the vehicle will always run the local strategy unless a connection to the cloud is established and the vehicle is deemed to be an eligible user.

While the development team delivers the function through company A's existing software release process, a group of fifty eligible users were selected. These users were selected at random from corporate customers who voluntarily signed up for the experiments and their user agreement covers the data collection. Each user was assigned to a vehicle for a total 18 month period, during which the vehicles were used as regular family cars. In the fleet, there are three types of unique vehicle model and powertrain combination. Internet

connection through 4G were provided to the users as the function requires active internet connection to the cloud. Therefore, when software including EM function changes was introduced to all vehicles in the fleet, the A/B experiment was only activated for eligible users. Furthermore, as part of the experiment, the users were given the option to switch off the cloud strategy manually form the vehicles.

To randomise the network and spill-over effects, the experimenters allocated A and B groups dynamically on a schedule. Upon activating connections to the fleet, the local strategy was ran on all vehicles for a short period to establish a baseline response. Then the team automated the partitioning of A/B groups through a rolling schedule while keeping the sample size of two groups balanced, i.e. exactly half of the fleet runs on the A variant and another half on B. During the experiment, there were in total 58 observables being measured simultaneously to evaluate the EM function from both A and B groups, including but not limited to net energy consumption, driving dynamics and travel demand patterns. The experimenters also monitored how frequently the users actively switch off the B variant themselves.

A number of automated mechanisms were put in place in the cloud to ensure data quality. Some observables were measured and stored as high frequency time series data while some were measured and stored as aggregate values or snapshots. The experimenters were able to define the data requirements according to their experiment hypothesis. The experimenters have access to the data collected almost in real-time. The data collected was post-processed in an automated manner in a database, while the teams can also choose to export the raw data. The file size of data collected per week averages at 1.7 gigabytes when exported in CSV format. All the analysis and evaluation were done to support decisions of further modifications to the cloud parameters when needed. B, the cloud variant of the function can be changed from the cloud without any modifications to the vehicles or the software within the vehicles.

The EM function software has dependency on six or more ECUs that are mostly supplier parts. Traditionally, changing the EM software means rebuilding of all these ECUs completely through suppliers and downloading the software to the vehicles physically. The usual lead time for such changes is anywhere from three month to one year.

With the cloud-based setup, the configuration of experiments can be done

with a simple flip of a switch and the parameter changes can be done within the functional teams. This system caters an environment where continuous experiments are independently from release processes that could be lengthy at times. In average, the total distance travelled by all eligible users is over 18.000 kilometres per week and over 80% of the vehicles are being driven daily. Comparing to any traditional test vehicle fleets, they are generating valuable measurements at a much larger scale more quickly. These measurements are also obtained from a wider range of driving conditions and more directly reflecting customer usage patterns of the vehicles.

Case study II

The second case study is conducted to understand the state-of-practise in company B and C, this is compared with the first implementation in company A. The summary is listed in Table. 5.2. The components in our architecture is listed, and the current adopted practises by each company are highlighted with check marks.

Table 5.2: System components adopted by case study companies.

System components	**A**	**B**	**C**
Parametrised function development	✓		
Integration to the existing release process	✓	✓	✓
Write cloud parameters to vehicles	✓		
Request observables from vehicles	✓	✓	✓
Centralised and cloud-based data collection	✓	✓	✓
Over-the-air software update	✓	✓	

Even though neither company has experienced the complete cloud-based A/B experiment system, but there are many commonalities in the components adopted. Through our interviews, it was apparent that both company B and C have adopted some levels of the practise, most specifically the off-board data collection capabilities. Company B has invested intensively for an online data collection system for their fleet vehicles, where they can monitor the entire fleet in real-time. A set of predetermined observables are measured, their

data collected and distributed to corresponding function development teams through a centralised database. Each functional team within the company can also request for more observables to be measured from the fleet, through a service request to the data team. A similar approach was reported from company C. A centralised cloud-based database was built to remotely acquire high quality data in a fast manner. The teams have the freedom to determine the sampling frequency accordingly to their measurement requirements. The data collected was used for various product development purposes such as remote diagnostics and predictive maintenance.

5.6 Discussion

In this paper, we presented a hybrid architecture combining in-vehicle and cloud elements that solves many problems in adopting continuous A/B experiments in automotive software. Comparing to existing A/B experiment architecture for embedded software, our architecture offers the flexibility of being independent from a continuous deployment process. By allowing parameter changes, the function can be experimented continuously without software changes.

However, we foresee some potential weakness in the design and they are discussed here. Firstly, the threshold of functional behaviour change through parameters is low comparing to a complete software change. The system enables A/B experiments for fine tuning of functions but not complete concept changes. In other words, the A/B testing can be done on relatively mature functions through our architecture.

Secondly, many parameters changes are not independent from each other in an automotive setting. When multiple experiments are running simultaneously, the configuration of experiments becomes critical as suggested by [35] and [32] from their experience in the online businesses. Similar to wed-facing applications, we need to consider contradicting parameters. At the same time, some parameters configurations can be hierarchical when one function is deemed to be a sub-function of something else. For instance, battery cooling temperature effects energy consumption management and climate comfort for the driver. This type of parameters adjustment needs to be coordinated when both functions are being A/B tested. Performance of hierarchical functions cannot be determined individually. As a result, centralised, well-established

and understood performance metrics need to be put in place before parallel/multiple experiments can be conducted simultaneously.

Thirdly, each team can independently determine parameters and observables. Ideally, the teams shall coordinate their experiment designs when parameters or observables are shared in between different functions. Such coordination requires organisational support [18]. As many automotive companies are going through agile transformation [3], the data-driven development mindsets and support structure are gradually improving. The speed of the transformation will influence how quickly an A/B experiment system can be put in place.

Fourthly, receiving cloud-based parameters requires an active internet connection that can be disrupted during a number of driving conditions, and usually has a delay in response. Meanwhile the functions can be safeguarded through using local parameters as fallback, a function which requires millisecond response time cannot rely on cloud parameters. Common examples of functions which are time critical include lane keeping assist, automatic breaking and active cruise control. A possible setup for time critical functions is to embed the A and B versions of parameters in the software itself, and use the cloud to trigger the switch in between them. As a trade-off, one will lose the freedom of tuning parameters independently from release processes.

Last but not least, some types of software are strictly governed by legal frameworks. Most types of legal requirements cover two dimensions. First, functional changes can not demonstrate deteriorated performance in legal compliance tests. Second, vehicles cannot behave worse on the road in comparison to the lab tests. We did not specifically address this issue in our software architecture, since legal compliance testing is built into software release processes in the industry. Moreover, the potential performance change from cloud parameters can be demonstrated in advance if we limited the upper and lower limits of parameters changes for all legally governed functions.

5.7 Conclusion

In recent years, some research effort was put in the adoption of A/B experiments in the automotive domain [3], [12], [24]. In this paper, we raised a research question on how to enable continuous experiments in an automotive, and presented an architecture that demonstrated such capabilities. Through

a literature review, we found that embedded experiment architectures share many components with web-facing ones, however, lack the capability of rapid changes. The architecture design is a hybrid A/B testing model that address many challenges in the industry. Comparing to existing frameworks, our hybrid architecture enable rapid software changes without compromising the high safety and security standards. Similar framework for automotive software A/B testing is not previously discussed in literature. We shared case studies of cloud-based A/B experiments at scale, which shows high potential of the parameterised hybrid architecture. The components of our architecture were compared with the state-of-practise of two other large automotive manufacturers. We found that the case study companies have applied many components, thus paving the way to an A/B experiment capable architecture.

CHAPTER 6

Size matters? Or not: A/B testing with limited sample in automotive embedded software

The layout has been revised.

6.1 Introduction

A/B testing, or A/B experimentation, is an online experiment technique for evaluating causal effects from software changes [16], [21]. In recent years, there is an increasing interest in adopting A/B testing in the automotive software businesses as it is considered an important tool for product development [3], [12].

As an online experimentation method, A/B testing relies on large sample sizes that are not always available in the automotive business. With hundreds of millions of users, challenges in increasing sample size were experienced in the web domain as reported by [16] and [21]. In the automotive domain, the available users to experiment with are notably more limited by orders of magnitude comparing to the web domain. Since the most popular vehicles are sold in the ballpark of one hundred thousand units annually, with an average

model sold in the range of tens of thousand units, and almost all vehicle models have local versions in their perspective sales markets. Moreover, we would want to experiment with a fraction of the total population of vehicles. For instance, our experiments often only involve a particular model of a sedan with a specific electrical machine in Northern Europe, which further reduces the available sample sizes. Thus, obtaining larger samples in automotive A/B testing is often unrealistic.

Two problems that occur from the use of experimentation with limited and small samples is the presence of random imbalance, i.e., the experimental groups are not compared prior to the experiment, and metric sensitiveness due to limited experimental power. Methods such as the CUPED (Controlled-experiment Using Pre-Experiment Data) and stratification [16], [21] are used in the web domain to increase the detection of changes within low sensitive metrics, i.e. metrics with high variance, however, they still require large sample sizes. Moreover, these methods are commonly used only with a single covariate and cannot be used interchangeably with numerical and discrete covariates. Re-randomisation and seed selection are often a potential solution to random imbalance, but it can increase the time to conduct an experiment and they are not guaranteed to provide balanced groups if there are changes in the design of different experiments. Research literature on A/B testing, experimentation in the software domain and in the automotive software development does not provide clear guidance on how to conduct experiments with low sample size and potentially imbalanced groups. Inspired by recent developments in the area of medicine and clinical trials, in this paper, we present a case study in automotive software utilising the Balance Match Weighted method [52] to create an experimental design that minimises group variance by balancing the control and the treatment groups with similar observed features (or, covariates). The design is guaranteed to provide maximum balance among the covariates and the analysis takes into account the covariates to reduce metric sensitiveness. Moreover, this design allows to include both numerical and categorical covariates. In this paper, features and covariates refer to the independent variables in a statistical model and the two terms are used interchangeably.

The contribution of this paper is three-fold. First, we present the Balance Match Weighted method to design experiments in detail. While this design has been used in other areas of science, this is the first paper to apply it to

experimental design in software development and in A/B testing. Second, we provide a case study, in the automotive domain, of the Balance Match Weighted design. With this design, we are able to draw valid conclusions from the experiment with significantly lower sample sizes compared to the randomised field experiments usually conducted in the web domain. Third, we discuss the advantages and limitations of the Balance Match Weighted in the design of experiments in the automotive domain.

The rest of this paper is arranged as follows. In Section 6.2, we present background and related work. Our research method is reported in Section 6.3. We describe the Balance Match Weighted design in detail in Section 6.4. The results from our empirical validation case study are presented in Section 6.5. The discussion and conclusion are shown in Section 6.6 Section and 6.7 respectively.

6.2 Background

The two-group design, also called A/B testing, is an experimental design method [26], [65]. In this design, users are randomly assigned to different variants of the product, the control variant (the current system) and the treatment variant (the system with a modification). The users are randomly assigned to different variants, and, after a period, the instrumented metrics for each variation are statistically compared. One of the assumptions of this design is that if that the users of each variant group are equally comparable, i.e., the only systematic difference between them is the introduced software variant. If this assumption holds, the research and development organisation can establish a causal relationship between the software modification and the differences observed in the metrics. Kohavi *et al.* [26] provide an in-depth discussion of common experimental design techniques used in online experiments.

Web-facing companies rely on randomisation and on a large number of users to ensure that the groups are comparable. However, due to the presence of random imbalance, even with large numbers randomisation is not guaranteed to produce comparable groups [48]. For instance, Bing used to re-randomise one out of four experiments due to random imbalance. Besides re-randomisation, Microsoft also utilises historical data to perform multiple A/A tests in order to find the best seeds to find the best-balanced groups [48].

A large number of diverse users in each experimental group lead to an in-

crease in the variance of the metrics. This increase in the variance leads to less sensitive metrics [61]. Research on A/B testing has provided different statistical methods to reduce variance on experiments such as stratified sampling and the CUPED method (Controlled experiment Using Pre-Experiment Data) [16], [21]. The CUPED method is similar to using control covariates in a regression. This method utilises pre-experiment data to identify covariates that can reduce the variance in the estimation and compensates for it in the average treatment effect.

Our proposed approach using the Balanced Matched Weight method addresses both the balance of the groups in small samples as well as reduces the variance in the metrics. It requires pre-experiment data to identify the features (or covariates) to balance the groups and utilises these features in a regression framework to reduce the variance of the metrics similarly to the CUPED method.

6.3 Research method

The objective of this study is to explore and validate A/B group design with the Balance Match Weighted method, in order to effectively A/B test with limited samples in the automotive domain. We employ a case study method to empirically explore the A/B group design method in an automotive company, following guidelines from [58] and [59].

This study is part of a larger research collaboration between several automotive companies aiming to introduce A/B testing at scale. As a first step to adopt A/B testing, the study company has deployed fleets of vehicles driven by internal users as testbeds for developing software architectural solutions, data analytic tools, and so on. Furthermore, as already identified in previous research [2], [3], [22] one of the limitations of A/B testing in automotive companies is the smaller sample size. In this context, the goal of this research is to identify techniques and experimental design methods aimed at inference with small samples.

Within medicine, experimental design minimisation techniques are widely used for small samples experiments [51], [52], and for experimental designs where there is prior information regarding the user characteristics and large variances between the users [66], the Balance Match Weighted yields good variance reduction by balancing the groups compared to full randomised ex-

periments.

This paper investigates the use of the Balance Match Weighted for experimental design in the automotive domain. This is captured by the following research questions:

RQ1: How can we apply Balance Match Weighted design for the partition of A/B groups in the automotive domain?

RQ2: What are the advantages and limitations of the Balance Match Weighted design in the automotive domain?

To address these research questions, we conduct a case study with an automotive company. The case study company is an automotive manufacturer. Their business includes the design, development, and manufacturing of passenger vehicles. We chose to utilise a case study method for the following reasons. First, A/B testing is not yet an adopted practise in the embedded system domain to the best of our knowledge, thus there is a lack of literature and empirical data. Second, a well-designed case study allows us to empirically validate the method in an automotive context, and it allows us to analyse the advantages of the design in its intended applications.

Data collection

In our research, we take advantage of the resources in the case study company and utilise two main sources of data collection. First, we actively worked with a software development team *in situ* for a period of six months. This development team consists of 34 members, their roles include software engineer, product owner, data scientist, etc. The team focuses on software solutions for vehicle energy management and optimisation. At least one of the authors participated in every project meeting, workshop, and discussions during the entire period, and provided inputs in relation to A/B testing. We collected meeting notes and design documentation.

The second source of our data collection is quantitative. From October 2020 to March 2021, we collect measurements from a fleet of 28 cars leased to employees of the case study company. The vehicles are commissioned for acquiring immediate user feedback of novel functionalities and they are driven as regular private vehicles. We instrument the vehicles with software that actively measures 51 signals through vehicle on-board sensors. The raw measurements are sent off-board and are permanently stored, and the research group is granted full access to the database.

Validity considerations

In this subsection, we present the threats to validity in our case study and how the threats are mitigated.

Internal validity

The propensity score model in the design method makes a strong ignobility assumption, which assumes that the effect on the target variable from the unobserved variables is minimum. Since our case study is designed around existing software, the observed variables are defined prior to the study. This implies that some co-founding variables might not be observed. No special action is taken to mitigate this risk as the ignobility assumption should be considered as an inherent limitation of the design method.

The quantitative trip data collection was done during a twenty-week period. We raised concerns on if an usage pattern can be established during a relativity short period. We mitigate this risk in two ways. First, after analysing the data, we have discovered that on the aggregated level, data from over 13,000 valid trips were collected and they are collected from a total of 205,000 kilometres driven distance. On average, each vehicle has made more than 250 trips during the period. Second, we have observed that the usage pattern of each individual vehicle does not differ drastically from week to week. Therefore, we consider the number of trip samples sufficient and we assume the seasonality effects in this fleet are low.

External validity

In this case study, we have applied the experiment design method to one software developed by one automotive company. We recognise the limitations of the approach and our findings might not be applicable to the entire automotive domain. However, we believe the design method can be adapted to run A/B experiments on similar software developed by other automotive manufacturers, as we demonstrated the design method using quantitative usage data that is arguably independent of the vehicle manufacturer.

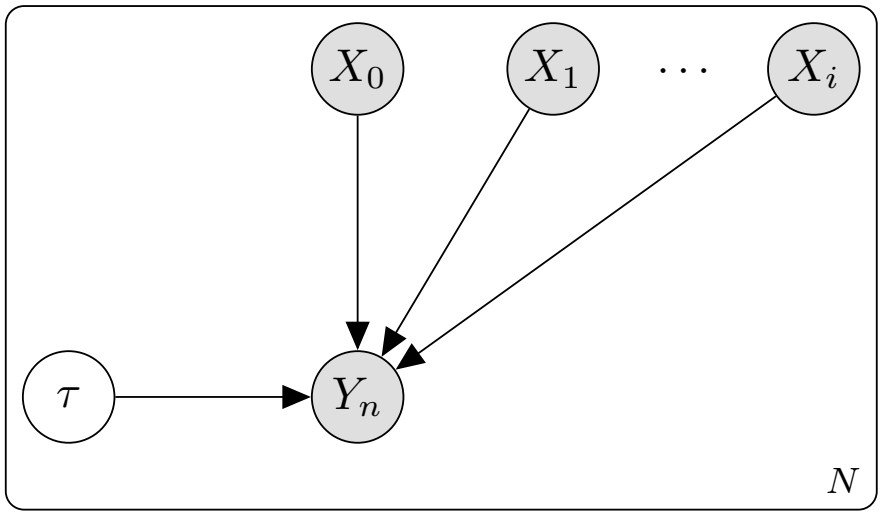

Figure 6.1: Relationships of input features ($\mathbf{X}$), treatment (τ) and target variable (Y) in the propensity score matching model.

6.4 The Balance Match Weighted design

This section provides a detailed description of the theoretical background of the Balance Match Weighted design method.

The Balance Match Weighted design is an extension to the propensity score matching method [20], proposed by Xu and Kalbfleisch [52]. In the original literature, the Balance Match Weighted design was used to select balanced groups for controlled experiments for medical research. Similar to some medical applications, A/B experimentation in software testing is not a traditionally controlled experiment, i.e., we cannot manipulate the boundary conditions of when and how the software is used. As a result, the measured treatment effects could be caused by other variables rather than the treatment itself. These variables could also result in a large variation in the measured treatment response, thus making treatment effects undetectable.

Prior to the experiment, when treatment has not been applied and the outcome is not known, Balance Match Weighted can be used in pre-experiment data to select the participants for each experimental group [52], [54], [66].

After the experiments have been conducted, the covariates used in the Balance Match Weighted are used to reduce the variance in estimating the average treatment effect [51].

The Balance Match Weighted design

Consider an A/B experiment, where the sample size N is small. If the groups are partitioned at random, it is likely we produce unbalanced groups. As a result, the response measured in the target variable Y could have high variance and we risk inconclusive experiments.

Algorithm 1 The Balance Match Weighted method

Inputs: M repetitions, N total number of users participating in the experiment

1: Identify the relevant features $\mathbf{X}$.
2: Randomise $N/2$ subjects in control ($\tau = 0$) and another $N/2$ subjects in treatment ($\tau = 1$) group.
3: **while** $m < M$ **do**
4: From the identified features $\mathbf{X}$, compute estimated propensity score distance δk_n.
5: Perform greedy full matching based on the propensity score distance by minimising $\Delta k_m = \sum_{n=0}^{N/2} \delta k_n$.
6: Record the triplet $\{\Delta k_m, n|\tau = 1, n|\tau = 0\}$
7: Select the control and treatment where Δk_m is minimum.

The Balance Match Weighed design was formulated by Xu and Kalbfleisch, the purpose is to reduce the imbalance and to increase the precision of the estimated treatment effect [52]. Comparing to the literature, we make a slight modification to the matching process to satisfy our constraints. It is an iterative process, described as the algorithm in Algorithm 1. In the following subsections, we discuss each step of the algorithm.

Feature selection

We use a network diagram to illustrate relationships in features, the target variable, and the treatment, as shown in Fig. 6.1. The shaded nodes are observed variables, the transparent node indicates if the sample is in the control or treatment group, and arrows indicate dependency.

Consider a set of i features, $\mathbf{X} = \{X_0, X_1, ..., X_i\}$, which are observed prior to the experiments. The observation is done for each individual subject sample n in the entire sample set $N \in \{0, 1, ..., n\}$, and believed to be predictors to the target variable Y. The changes in Y are dependent on $\mathbf{X}$. The target variable Y is also what the treatment τ aims to influence. We use τ as an indicator on whether the treatment is applied, $\tau \in \{0, 1\}$. $\mathbf{X}$ is independent from τ. We consider cases when control and treatment groups are even. Treatment will be applied to $N/2$ samples, with another $N/2$ in the control group.

In a successful A/B experiment, the treatment effect is sufficient to detect and therefore the expectation of the target variable Y is, $\mathbb{E}(Y|\mathbf{X}, \tau = 1) - \mathbb{E}(Y|\mathbf{X}, \tau = 0) \neq 0$.

An important assumption made in the model is ignobility [20], [51]. That is, we assume unobserved features do not affect the target variable Y. To satisfy this assumption, an optimal model includes all known features which correlate to the target variable only and not to the treatment [67], [68]. As shown in Fig. 6.1, there is no dependency in between features $\mathbf{X}$ and treatment τ. When the sample size N is small, including a large number of i features, might not be feasible [51]. In this case, a recommendation made by Rubin [66] suggests to first include a small set of features known to be related to the target variable, perform the matching and experiments, then include more features if bias is high in the outcome. One should not include the target variable Y in the propensity score model.

Propensity score distance

After selecting features that are highly informative of the target variable, the next step is to calculate the propensity score. To compute the propensity score ρ, we fit the input features $\mathbf{X}$ to a logistic regression, with indicating variable $\tau = 0$ for the control group and $\tau = 1$ for the treatment group.

We obtained the propensity score from the outcome of the logistic regression. The propensity score is a probability value that falls between 0 and 1. The individual propensity score distance for each subject n is defined as the absolute difference of propensity score in the control $(\mathbf{X}|\tau = 0)$ and treatment $(\mathbf{X}|\tau = 1)$ group,

$$\delta k_n = |\rho_{n,\tau=0} - \rho_{n,\tau=1}| \tag{6.1}$$

where,

$$\rho_{\tau=1} = P(Z = 1|\mathbf{X}_n) = \frac{e^{\beta_0+\beta\mathbf{X}}}{1 + e^{\beta_0+\beta\mathbf{X}}} \tag{6.2}$$

and β are fitted coefficients for the linear logistic regression model, β_0 is the fitted intercept. The total propensity score distance for all subjects in the control and treatment group is defined as:

$$\Delta k = \sum_{n=0}^{N/2} \delta k_n \tag{6.3}$$

Prior to an A/B experiment, the treatment indicator τ is unknown. Therefore, in the scenario of calculating propensity scores to design experiment groups before the treatment is applied, we randomise the control and the treatment groups as step 2 of the Balance Match Weighted method.

Greedy full matching

After computing the propensity scores, one should perform a matching of control and treatment groups. There are some commonly applied matching methods, including caliper matching, 1:1 nearest neighbour [54] matching, and full matching [52], [69].

In the existing literature proposing the Balance Match Weighted design, matching is achieved through the optimal full match [69]. Optimal full match makes replacement, meaning that one subject in the control group can be matched to multiple subjects in the treatment group. Furthermore, optimal full match allows discarding of subjects from the sample group, which is considered as a hard constraint in our case study for the following two reasons. First, our experiment subjects, the vehicles are costly to run without being included in the experiments. Second, since the matching is done prior to the experiment with an unknown treatment effect, we do not yet know the target variable but an expected outcome, discarding subjects at this stage is considered premature. Thus we suggest a greedy full matching should be performed to match all subjects. In practise, after the treatment is applied, one can discard subjects based on propensity scores computed from the actual control and treatment groups.

We formulate the matching of propensity scores as an optimisation problem,

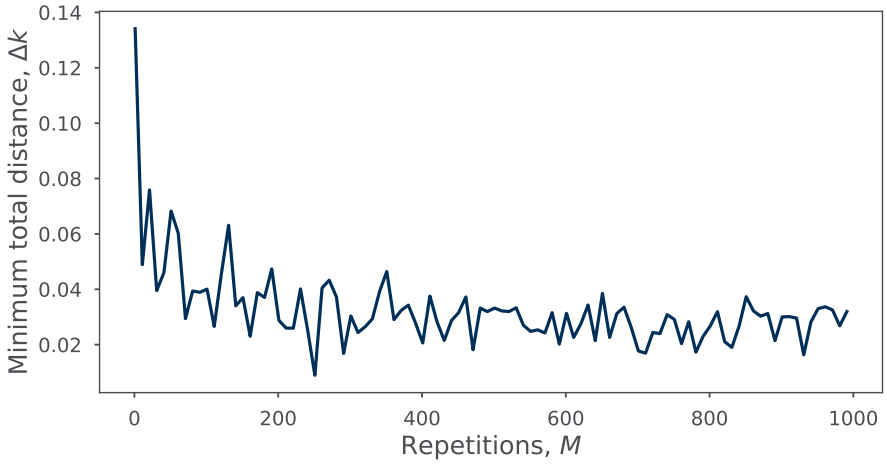

Figure 6.2: Minimum total distance (Δk) calculated from the propensity scores reduces as we increase the repetitions (M).

with the objective to minimise the global propensity score distance (Δk) in between control and treatment groups. We perform the greedy matching without replacement as we assume all subjects are independent. This means that each subject in the control group can have only one corresponding subject in the treatment group.

The repetition parameter M

In the literature, Xu and Kalbfleisch [52] suggest the larger M is, the better the results this design will obtain. We decide on the number of repetitions by running the design with increasing repetitions of resolution 10, that is $M \in \{1, 10, ..., 1000\}$, and analysing the trend of minimum total distance Δk as M increases. We illustrate an example in Fig. 6.2, for $N = 28$ with six features, the improvement for Δk becomes negligible after $M \approx 500$ repetitions. An elbow effect is reached when the improvement of Δk becomes minimum and the improvement no longer justifies the computational cost.

6.5 Case study

We carried out our case study from October 2020 to March 2021, with a fleet of 28 passenger vehicles. These vehicles belong to a large project of collaborative development with company car users from the case study company. All vehicles from the fleet are driven by corporate users as their primary family cars. In our case study, we select vehicles of the same model with the same electrified propulsion, and the selected vehicle model is commercially available. The vehicles are in the possession of users since December 2019, and all users reside in Västra Götaland County, Sweden.

We aim to test the vehicle energy management (VEM) software, that can help reduce energy consumption through the prediction of vehicle routing. From the confidential consideration of the software, we will not further discuss the functionality itself in this paper. We introduce a variant A of the software to the fleet for a continuous period to collect pre-experiment data. Utilising this data, we design the partitioning of control and treatment groups with the Balance Match Weighted method. The B software variant is then shipped to the treatment group accordingly to the group partition. The results of our group design with pre-experimental data, and the experiment design and outcome are presented in this section.

Case study fleet

This case study is done in two distinct periods. First, we have a twenty-week interval that is the observation period prior to applying treatment. Data generated from this period are used to partition the A/B groups. After that, there is an experiment period for two weeks during which the treatment is applied. Data collected during the experiment period are for analysing the group design and the actual treatment effects.

Aiming to simulate the real usage of cars in the case study, we do not dictate how the vehicles are driven. All the measurements and testing are done single-blinded, i.e., we do not interact with the users at all and they are not informed of the details of the A/B experiment. Measurements are done through the on-board sensors of the vehicle. We trust the measurements to a very large extent as the same sensors are used for calibration and diagnostic of all other functions in any commercially available vehicle. The measurements are done continuously during all trips, and transmitted to a cloud storage

through the telematics system of the vehicles via 4G. The data generated are in time series at 10 Hertz, marked by an anonymous version of the vehicle identification number (VIN) for each vehicle and a unique ID for each trip. We measure 51 signals from each vehicle, including but not limited to velocity, engine usage, climate system usage, GPS position, and so on. In the data collection, we discard measured trips that are less than one minute in duration, or one hundred meters in distance. In post-processing, we generate a number of observed features from the raw measurements.

The VEM software tested in this paper was developed internally and shipped by the function development team. The software was tested and validated through the standard processes in the case study company. To enable full flexibility of an online A/B experiment, we adopt a hybrid architecture for this software. The architecture determines that the software has two sets of parameters, local (A) and cloud (B). The local set of parameters are defaults for all vehicles with the same configuration. While the cloud set of parameters are blank onboard but can receive external values from a cloud. Since the VEM function is fully parameterised, this setup allows us to configure the function behaviour remotely. During the observation period, all 28 vehicles are set on the A variant of parameters. After the data is collected and analysed from the observation period, we partition the A and B groups and switch to the cloud parameters for group B.

Selected features

We approach our feature selection both quantitatively and qualitatively. As the vehicles were generating data from over 51 signals at 10 Hertz, on the weekly average, the dataset size is around one gigabyte when exported in the CSV format.

In the quantitative selection process, we first aggregate all raw signals measured from the time series and compute the target variable Y and all potential features from a few of these signals. We do this on both the trip level and car level. We generate descriptive statistics to explore the correlation of target variables and all potential features. In the end, we select six of the variables to be included in our features. We examine the change of target variable over time, as well as its correlation to the features, see Fig. 6.3. These variables are expected to be informative to the target variable. The features are strongly correlated with the target variable and such correlation is consistent over time

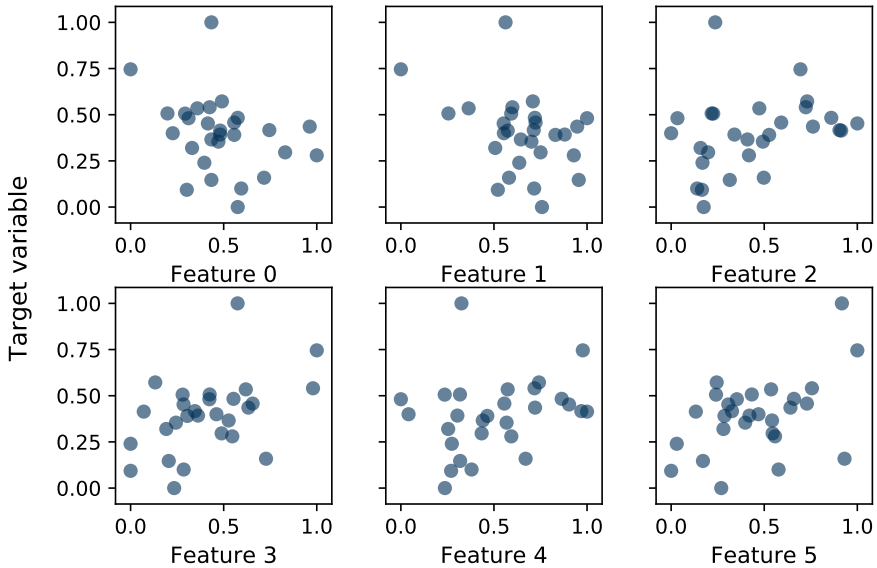

Figure 6.3: Scatter plots of feature 0 through 5 and their correlation to the target variable, min-max scaled.

for the same vehicle. Feature 0 and 1 have a negative correlation with the target variable, at -0.32 and -0.37 respectively. Feature 2 through 5 are positively correlated with the target variable. The minimum correlation is at 0.26 in between feature 4 and the target variable, and the maximum correlation is 0.47 in between feature 5 and the target variable.

To ensure all known covariates which affect the target variable are included in the input features, we validate our features selected quantitatively with expert workshops. We present our feature selection method and outcomes to a group of experts who actively developed the VEM software. Our proposal derived from data aligns with expert knowledge. With that being said, we are testing a novel function which implies that we do not have more experience or data to rely on. We are aware there could be unobserved covariates that can affect the target variable. Such shortcomings should be considered as an inherent limitation of the Balance Match Weighted design method for A/B experiments.

Table 6.1: Mean and variance of each of the five input features **X**, min-max scaled, in the matched control and treatment groups.

		Feature 0	**Feature 1**	**Feature 2**
Mean	Control	0.51	0.64	0.46
	Treatment	0.46	0.67	0.43
Variance	Control	0.06	0.03	0.10
	Treatment	0.04	0.06	0.07
		Feature 3	**Feature 4**	**Feature 5**
Mean	Control	0.42	0.53	0.48
	Treatment	0.41	0.48	0.43
Variance	Control	0.06	0.08	0.06
	Treatment	0.07	0.08	0.08

Matched A/B groups

At the end of the observation period, we collected the data from all vehicles. We extract the features $\mathbf{X}$ and the target variable Y from the raw measurements. The Balance Match Weighted design is applied to the dataset following the steps prescribed in Section 6.4.

For each of the 28 subjects, the six features included in the model are aggregated on the vehicle level and stored in a 28×6 matrix. That is, each vehicle has six features that represent its usage pattern, all of which strongly correlate to the target variable. The target variable Y is not included when estimating the propensity scores. Before calculating the propensity score, the observed features are scaled with their perspective minimum and maximum values to minimise bias from extreme values in the observation. We run the design with a high number of repetitions, $M = 1000$, and obtain the A/B group partition with $N/2 = 14$ subjects in each group. We show the kernel density estimation of the target variable measured during the observation period. The distribution is shown for when the groups are partitioned are random, and when the groups are partitioned using the Balance Match Weighted design, see Fig. 6.4. Comparing to random split, the matched A/B groups have a more balanced distribution of the target variable when the groups are running the same software.

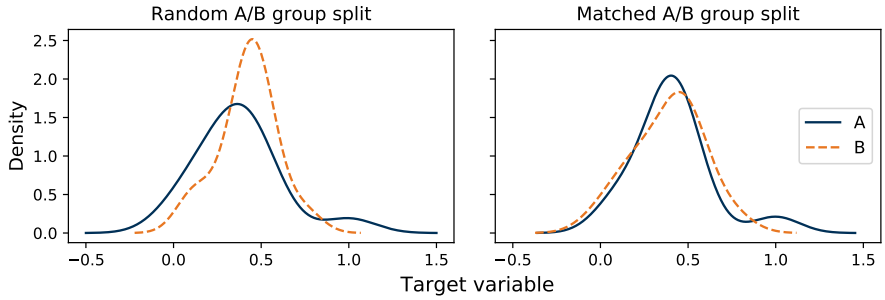

Figure 6.4: Kernel density estimation of the target variable, min-max scaled, of A and B groups when matched at random (left), and matched using the Balance Match Weighted design (right).

The goal of matching is to achieve feature balancing, namely features in the control and treatment groups are from the same empirical distribution, $p(\mathbf{X}|\tau = 0) = p(\mathbf{X}|\tau = 1)$. Following advice from literature [51], [67] , we further diagnose the validity of the matched groups by comparing their scaled mean and variance of the features. We present the results in Table.6.1, as can be seen, the means are demonstrated to be similar in the matched groups as well as the variance in the two groups. The average propensity score for the matched control and treatment group is 0.49 and 0.50, respectively, while the minimum values are 0.46 and 0.45, and the maximum values are 0.54 and 0.56, respectively. The resulting group partition exhibits high similarities in the empirical distributions with merely 14 subjects in each group. A higher level of similarity in the empirical distributions would be expected, if the number of subjects N increases [52].

Experiment outcome

We conduct the A/B experimentation by introducing the B version of the software through a set of cloud-based parameters to the matched B group. This A/B experiment serves as a demonstration of the Balance Match Weighted design method. The experiment was run for a continuous two-week period, we measure the target variable Y which is expected to reduce with using the new software.

We include an analysis of a paired test. A paired test is when we compare

the target variable Y from the same group of users, measured before and after the treatment is applied. This type of paired analysis can eliminate variation caused by the individual subject's preferences, however it is limited in quantifying external seasonality effects. We include this analysis to illustrate the benefit of an A/B test with matched groups in comparison to a pseudo-random experiment. In the matched A/B test, the mean squared error (MSE) shows an 17.9% improvement from the paired test, similarly to what is reported by [52] at $N = 30$ with eight input features.

In terms of the expected treatment effect $\mathbb{E}(Y)$, We found that, comparing to a paired test, a matched group A/B test returns 37.87% less standard deviation in the target variable. The min-max scaled average treatment effect, $(\bar{Y}|\mathbf{X}, \tau = 1) - (\bar{Y}|\mathbf{X}, \tau = 0)$, is -0.134 and -0.180 for the paired test and matched A/B test, respectively. The matched group A/B test yield lower variance in the target variable measured, at the same time returning a larger average treatment effect.

Recommended procedure

We summarise the procedure of our experiment design and list them here in a step-wise manner.

Determine eligible subjects and observe for a period

The eligibility of the subjects shall be determined based on the purpose of the A/B tests. It is ideal if the subjects themselves are directly comparable. For example, in the same A/B test, only the same vehicle model or engine types are included. Or, if deemed necessary, the categorical variables or dummy variables which determine such vehicle properties can be included in the input features. Moreover, decisions on the duration of data collection should take probable seasonality effects into consideration. The seasonality effects could either be well-known before the observation starts or discovered during the observation. In the second case, the observation period should be reasonably extended to measure such effects.

Select input features

The input feature selection shall be done both quantitatively and qualitatively. As the data from the potential experiment subjects are collected through

observations, selecting features based on pure statistical correlation might lead to spurious correlations and other problems alike. We strongly recommend a more comprehensive approach that combines expert inputs with data, to ensure that all known covariates are taken into account in the model.

Run the Balance Match Weighted design

For the case study, we have implemented the Balance Match Weighted design algorithm in Python. The code takes the features $\mathbf{X}$, repetition number M and total sample size N as input, returns the ideal partition of A and B groups, the total propensity score distance Δk, and propensity score δk_n for each pair of subjects. The code will not be shared at this stage due to our confidentiality agreement with the case study company. However, there are similar and publicly available R packages[1] for performing the matching design [52], [69].

Apply treatment and collect data

After the groups are designed and the treatment applied to the B group, the A/B experimentation shall run for a continuous period of time. In our case study, we have predetermined the length of the experiment as we observed our subjects to have a consistent travel pattern from previous data. To avoid false positive or false negative conclusions, if or when there is a high variance of the features and the target variables over time, the experimentation shall not have a predetermined duration but terminate only when a conclusion is reached.

Analyse experiment outcome

When analysing the experimentation outcome, instead of directly computing the treatment response, one should also analyse if there is a significant difference in the input features before and after applying the treatment to validate the group partitioning. This is to ensure the balanced group portioning modelled from pre-experimental data still holds, and the A/B groups are still directly comparable. If there are discrepancies between the features measured prior and during the experimentation, one may perform a propensity score

[1] github.com/markmfredrickson/optmatch

matching to only select subjects that are comparable. Instead of comparing the point estimates, we suggest to visualise and compare the total distribution of the target variable measured in the A and the B groups.

6.6 Discussion

In this section, we discuss the use, advantages, and limitations of the Balance Match Weighted design method applied in automotive software engineering. This design method allows us to conduct A/B testing on limited samples, which is considered a major challenge in adopting A/B testing in the automotive domain [3]. While our case study is an extreme example of limited sample size, through the study, we have demonstrated the intuitiveness of the group design method and the simplicity in adopting such method. Small sample A/B testing can also be beneficial when applied in an agile development process, where the sample size can be gradually increased at each development iteration if the experiments are conclusive. With that been said, we have discussed and experienced some limitations in applying the Balance Match Weighted design method in the automotive domain. They are listed in the subsections below.

Existing data and unobserved variables

Similar to the CUPED method [16], [21], performing the Balance Match Weighted design requires pre-experimental data. When the software in a novel product (e.g., a new model of vehicle) is the subject of interest, there might not be relevant existing data nor existing users. The Balance Match Weighted design can only achieve balance in the features that are observed [66]. But when a novel software is being tested, we do not always have a comprehensive picture of which features should be included in the observation. An incremental approach can be taken. To start, the development teams hypothesise the appropriate features and target variable prior, then gradually increase the number of features and sample size if the treatment effects are positive. This development activity can be planned accordingly to agile methods. As an alternative, an experiment group design method called Minimisation can be applied. Minimisation matches users as they enter the experiments [70]. Because it is reasonable to expect the number of eligible users to gradually

increase.

Multiple driver households and car sharing

During the case study, we did not have any reliable method to define if/when the vehicle is driven by different people. Technology and privacy agreements limit the identification to individual vehicles only. This means that if there is more than one driver sharing a vehicle, we will not be able to capture the variation in the measurements caused by the driver change. However, we do not believe that this affects the case study outcome greatly as the VEM function does not interact directly with the drivers, therefore the function behaviour does not strongly depend on the preferences of individual drivers.

In the automotive setting in general, we recognise the benefits of distinguishing multiple drivers sharing the same vehicle. To capture the actual preferences of the drivers, user matching, partitioning of groups, and A/B testing should be done on the driver level instead of vehicles. As driver distinction would generate more informative features, and an understanding of driver preferences is arguably necessary when user-facing software is tested.

6.7 Conclusion

A/B testing with limited samples is a challenge in the automotive sector. To address this challenge, we evaluate and report an experiment group design method, Balance Weight Matched design, that can effectively increase the experiment power with small samples. In this paper, we provide a detailed presentation of the design method and a step-by-step implementation procedure. In collaboration with an automotive company, we conduct a case study to apply, demonstrate and evaluate the design method in a fleet of 28 vehicles with two versions of an energy optimisation software. In the case study, we worked within a development team in situ. From pre-experimental data, we found that feature balance can be achieved with merely 14 subjects in each group. After introducing the software treatment to the matched B group, compared to a paired test, the matched A/B test returns 37% less standard deviations in the target variable while improving the MSE by 17%. We conclude that this design method is advantageous for conducting A/B testing in the automotive embedded software domain. As shown in our case

study, balanced groups can be produced when the sample sizes are considerably small and it improves the power of small sample experiments. Finally, we discuss some potential challenges and limitations in applying the Balance Weight Matched design in the automotive domain, including the ignorability assumption, conducting experiments with no prior data and we highlight the importance of differentiating drivers when a vehicle is shared.

In our future work, we plan to further investigate the Balance Weight Matched design method with more datasets and software, as well as develop tools for experiment analysis.

CHAPTER 7

Bayesian propensity score matching in automotive embedded software engineering

The layout has been revised.

7.1 Introduction

Over the past decade, field experiments (or, online experiments) have become a ubiquitous part of software development. Software-as-a-Service (SaaS) companies have long shared success stories of the use of experiments to assess the value software features deliver to users [16], [21], [35]. These success stories have led companies beyond the SaaS domain, specifically automotive companies, to show interest and even start running experiments [2], [3], [22], [24], [71]. Nevertheless, the automotive domain faces many unique restrictions compared to SaaS companies, such as number of software variants, architecture restrictions, safety-regulation constraints, number of vehicles available for experimentation, driver consent, and the ability to frequently update software in customer vehicles due to limitations such as user and privacy agreements among others [3], [22]. A combination of these challenges leads to many sit-

uations were the design of an online experiment is not: (1) possible, such as in limited samples; (2) desired, such as in safety-critical systems; (3) ethical, such as without explicit consent of the drivers on the scope of the software.

These restrictions to properly conduct a field experiment require that the research and development organisation to utilise a range of different causal inference techniques to assess the value delivered by the new software. In this paper, we propose the use of the Bayesian propensity score matching technique for observational causal inference in the automotive domain. We introduce the BOAT (**B**ayesian propensity score matching for **O**vserv**A**tional **T**esting) method. This method is used to generate balanced control and treatment groups from observational data and estimate causal treatment effects from software changes, even with limited samples. The BOAT method is based on the propensity score matching framework by Rubin [67]. The propensity score matching framework has been developed and wildly applied in medical science [51], [52], [54], [66], in traffic safety analysis [46], [72], in SaaS systems [17], and in automotive software for experiment design [71].

We demonstrate the BOAT method using a proof-of-concept in the automotive domain. We ship a modified software variant to a part of our case company's internal fleet, and compare that to a larger population of vehicles that are equipped with the existing variant of the software. As in the automotive sector, the access to update customer vehicles is significantly more limited than data collection. When we can collect data from more vehicles than we can ship software to, we have skewed sample sizes in the control and treatment group. Our proof-of-concept is designed to simulate such scenarios. Therefore, the new software is only download to a limited number of vehicles, these vehicles are driven by the employees as their primary personal cars. The control group ($N_c = 1100$) of cars uses the current software variant and the treatment group ($N_t = 38$) utilises the new software. We collect measurements from vehicle on-board sensors for a continuous period of five months, engineer the input features to BOAT, and perform a matching to produce control/treatment groups with balanced empirical distribution of the features. Note that, features and covariates refer to the independent variables in statistical models, and we use the two terms interchangeably.

Comparing to the existing literature, this paper provides the following contributions. First, we describe the theoretical background of Bayesian propensity score matching model. To the best of our knowledge, this is the first time

such a model is used in software development publications. Second, we discuss the feasibility of such an application in automotive software engineering. Third, we share the process of rolling out small-scale observational testing of automotive software. In combination with the BOAT model, we are able to introduce observational testing of novel software in a fast and more robust manner, and conclude the causal effects of such software change from observational studies.

The rest of the paper is organised as follows. In Section 7.2, background and related work are introduced. In Section 7.4, we describe the Bayesian propensity score matching theory in detail. We present the data structure and collection method in Section 7.3. The results are presented in Section 7.5. The discussion and conclusion are presented in Section 7.6 and Section 7.7 respectively.

7.2 Background and related work

To evaluate software changes, companies use randomised field experiment techniques such as A/B testing [16], [21], [26], [35]. In their experiments, users are randomly split into two large groups and introduced to different variants of the software. The variants usually include the existing variant (control) and a modified one (treatment). The assumption of such online experiments is that the sample size is large enough, thus the two or more groups are balanced and directly comparable, and the only difference is the software variant. When the assumption holds, the experimenters can establish a causal relation in between the software change and the treatment effect through measuring carefully designed metrics. Online experiments conducted by SaaS companies benefit from their large user base. Yet unbalanced groups could be produced in those experiments due to high diversity in users [16], [21], known confounding factors such as user preferences [17], or other unknown confounding factors [48]. These issues are addressed through techniques such as propensity score matching for online quasi-random experiments [17], stratified sampling and the CUPED method (Controlled experiment Using Pre-Experiment Data) [16], [21].

Conducting large and fully randomised online experiments can be more challenging in the automotive domain [2], [3], [22], [24]. First, the available users in the automotive domain are comparably more limited than in SaaS,

and most manufactures have a high diversity in their products (e.g., vehicle customised options), which will further reduce the available samples for large online experiments [3], [71]. Second, a vast majority of automotive software are in safety critical systems [12], [24]. Although the risks of safety compromise are minimum since the modified software variants will go through the same release process [3], but even minor disturbances at a scale can be directly translated to the profit lost for commercial vehicles such as trucks and taxis. Thus, it is undesirable to ship software for safety critical systems to a large portion of the vehicle fleet at once. Last but not least, software update and data collection requires explicit consent from the vehicle users.

As a result, comparing to large and fully randomised online experiments, automotive software online evaluation is much more feasible to be done in the format of small-scale observational studies, where the new software is only introduced to a small and selective group of vehicles. An observational study is to be conducted when a randomised experiment is not feasible and a causal relation of treatment and effect is to be established [73]. Therefore, it is critical to present causal inference methods, such as propensity score matching. With propensity score matching, one can utilise pre-experimental data to design balanced control and treatment groups, as previously demonstrated by [52], [66] in the medical sector and [71] in the automotive domain. Moreover, propensity score matching has been applied in the field of software engineering, in the efforts of analysing development efficiency [74], [75].

Bayesian propensity score matching (BPSM) is an extension of the traditional framework of propensity score matching, in which the propensity score is estimated through a Bayesian network. Using Bayesian statistics, one can conjugate the posterior distribution based on a prior, a likelihood and evidence. In other words, Bayesian statistics allows one to model based on the data and the domain knowledge [76]. Instead of providing only a point estimate of the dependent variable, Bayesian models will return the entire posterior distribution, therefore, quantifying uncertainty. BPSM, as applied in the filed of traffic research [46], has shown a higher performance than the frequency approach for small samples.

7.3 Input data

In this section, we will present the data used for the BPSM model, the collection methods, and how each data feature is engineered from the measurements. The data collection is done from the 26th of October 2020 to the 22nd of March 2021. The measurements are collected from two specific vehicle models.

Table 7.1: Descriptive statistics of the target variable and covariates, and a description of how the variables are computed. Each variable is aggregated to the vehicle level and max-min scaled.

Variables	Variable description	Group	Mean	Std.
Target variable				
Fuel consumption	total fuel injected in engine /	A	0.391	0.155
[g/km]	total distance	B	0.354	0.123
Covariates				
Share of trip start	number of trip where	A	0.356	0.177
with full battery	soc_start >80%	B	0.397	0.178
Share of trip end	number of trip where	A	0.258	0.155
with low battery	soc_end <21%	B	0.120	0.150
Number of trips	number of trips taken place	A	0.290	0.167
made on weekdays	during weekdays	B	0.225	0.153
Number of trips	number of trips taken place	A	0.269	0.173
made on weekends	during weekends	B	0.190	0.154
Average trip	total trip distance / total	A	0.301	0.132
distance [km]	number of trips	B	0.343	0.124
Maximum trip	longest trip occurred during	A	0.278	0.193
distance [km]	the observation period	B	0.240	0.186
Average trip speed	total trip distance / total	A	0.575	0.110
[km/h]	trip duration	B	0.624	0.117
Maximum trip	highest trip speed occurred	A	0.637	0.125
speed [km/h]		B	0.640	0.135
Share of distance	share of distance driven in	A	0.956	0.103
on "hybrid"	hybrid mode	B	0.987	0.033

Continued on next page

Table 7.1 – continued from previous page

Variables	Variable description	Group	Mean	Std.
Share of trips with a trailer attached	share of trip with trailer attached	A	0.034	0.081
		B	0.034	0.075
Average number of engine starts	share of occurrence of engine RPM >500	A	0.169	0.112
		B	0.176	0.176
Average ambient temperature [$°C$]	average temperature measured at car	A	0.371	0.084
		B	0.372	0.071
Minimum ambient temperature [$°C$]	minimum temperature measure at car	A	0.497	0.135
		B	0.563	0.150
Maximum ambient temperature [$°C$]	maximum temperature measure at car	A	0.388	0.101
		B	0.374	0.099

The measurement of the vehicles are done through on-board sensors for vehicle control, calibration, and diagnostics. We select low level signals for their robustness and reliability. The measurements are done during each drive cycle of the vehicle in a time series format at ten hertz frequency, marked by an arbitrary vehicle ID that cannot be decoded to identify the user nor the vehicle, and a drive cycle ID. Drive cycle refers to the events in between each vehicle key-on and key-off, and a trip for the user could consist multiple drive cycles. The measurements are sent to a central server of the case company through a telecommunications module in the vehicle, no physical access is needed to obtain the data.

To further increase the robustness of the measurements collected, we read the values from two or more sensor signals for each measurement. We intend to exclude drive cycles in which multiple signals yield drastically difference values. However, we found that the same measurement calculated from different signals only differ by a decimal point on the drive cycle level. For example, the drive cycle distance can be calculated through integrating the instantaneous vehicle velocity, or through a wheel speed sensor that measures the angular velocity of the wheels. These two signals differ by 0.0227 kilometres for the 75 percentile of drive cycles. Moreover, we measure values in base units. E.g., fuel consumption is measured in grams, because the commonly used unit litre is a secondary value dependent on the pressure and tempera-

ture. Furthermore, we do not include entries if they have any of the properties listed below. After postprocessing, we have in total 421,881 drive cycles made by 1138 vehicles of which 38 are in the treatment group.

- Drive cycle is made by vehicles with odometer distance less than 100 kilometres, i.e., brand new vehicles.
- Drive cycle average speed is greater than 200 km/h.
- Drive cycle total distance is less than 0.5 kilometres.
- Drive cycle total duration is less than one minute.

Data structure

The input data features to BPSM model is produced from the time series values collected from the vehicles. Note that due to our confidentiality agreement with the company, the input data will not be shared nor shown without scaling in this paper. First, we aggregate each measurement per drive cycle through multiple vehicle signals. Since multiple signals do not return a different outcome beyond a decimal point, we select one value to keep. After this step, we produce a dataframe that compresses of one drive cycle per vehicle per row.

Second, we calculate the features based on all trips per vehicle and produce 14 input features and one target variable to BPSM. The input features are stored in a matrix with dimension 1138×14, which corresponds to 1138 vehicles and 14 features. The features and how they are calculated are presented in Table 7.1 along with the descriptive statistics. Each feature is scaled with their perspective minimum and maximum values.

7.4 Bayesian propensity score matching

In this section, we present the theory of Bayesian propensity score matching (BPSM) in detail. A probabilistic graphical model is used to illustrate the Bayesian logistic regression generative model, and we present the prior, the evidence and the posterior in this Bayesian network. Finally, we describe different matching strategies and the ones applied in the paper.

Propensity score matching, first introduced by Rosenbaum and Rubin [20] in 1983, is a causal inference model for estimating treatment effects from observational studies. In an observational study, the measured treatment effects could be caused by confounding variables than the treatment itself, thus raise bias in the results. When the sample sizes are limited, propensity score matching can help us create balanced and comparable groups by matching the control and treatment groups, so that the covariates from both groups form similar empirical distributions. Propensity score matching can be used to design partitioning of control and treatment groups based on pre-experimental observations [52], [54], [66], [71], or used for causal treatment/no-treatment effect analysis of existing observational studies postmortem [17], [46], [72]. The most important assumption of propensity score matching is ignorability [51], which implies the unobserved covariates do not influence the target variable, thus ignorable. In other words, propensity score matching can only balance covariates that are observed but a fully randomised experiment with a large sample can balance all covariates, observed or not.

In a two-group observation study with total sample size N, the average treatment effect (ATE) is defined as the difference of the average expected value of the target variable in the control ($\mathbb{E}(z_n|y_n = 0)$) and treatment group ($\mathbb{E}(z_n|y_n = 1)$),

$$ATE = \frac{1}{N}\sum_{n=1}^{N}(\mathbb{E}(z_n|y_n = 1) - \mathbb{E}(z_n|y_n = 0)) \tag{7.1}$$

Where $y_n \in \{0, 1\}$ is a control ($y_n = 0$) or treatment ($y_n = 1$) indicator for each sample $n = \{1, 2, ..., N\}$. For each n, we observe a total of I numbers of covariates x_i, $x_i = \{x_1, x_2, ...x_I\}$, which are correlated with the target variable z_n, denotes as $\mathbf{x}_n$ for all samples N. The $\mathbf{x}_n$ is a matrix with dimension $N \times I$. Covariates $\mathbf{x}_n$ are the confounding variables that would potentially influence the target variable z_n. The potential outcome of the target variable is independent of the treatment assignment given the covariates,

$$(z_{n,c}, z_{n,t}) \perp y_n | \mathbf{x}_n \tag{7.2}$$

The average expected treatment effect becomes conditional to both treatment y_n and the covariates $\mathbf{x}_n$,

$$ATE_{PSM} = \frac{1}{N}\sum_{n=1}^{N}(\mathbb{E}(z_n|x_n, y=1) - \mathbb{E}(z_n|x_n, y=0)) \tag{7.3}$$

There are two steps in propensity score matching. The first step is the estimation of propensity score through logistic regression followed by performing matching of samples in the control and treatment groups based on their propensity scores. In BPSM, the estimation of the propensity score is done through a Bayesian logistic regression, which returns a mean propensity score for each sample and their uncertainties.

Probabilistic graphical model

A probabilistic graphical model for Bayesian network is a directed acyclic graph, in which the shaded nodes represent the observed variables such as features and treatment indicator variable. The bright nodes are latent variables. The directional edges indicate conditional dependencies in between variables, and the unconnected nodes are conditionally independent. The plate is a representation of the number of observations, i.e., samples. The first step in BPSM is to estimate the propensity score through a logistic regression. A probabilistic graphical model for Bayesian logistic regression is shown in Fig. 7.1.

Consider a non-randomised study where the users in the treatment group are not randomly assigned and there are only a limited number of users in the group. We have a total number of samples N, in which there are more or equal number of samples in the control group (N_c) than the treatment group (N_t), $N_c \geq N_t$. The regression coefficients, α and β are latent variables. That is they are not observed but inferred from other variables that are observed. The treatment indicator y_n is binary, and it follows a Bernoulli distribution,

$$y_n \sim Bernoulli(y_n|p_n) \tag{7.4}$$

where the propensity score p_n is calculated as,

$$p_n = \frac{e^{\alpha+\beta\mathbf{x}_n}}{1+e^{\alpha+\beta\mathbf{x}_n}} \tag{7.5}$$

The regression intercept α has a prior of Gaussian distributions of 0 mean

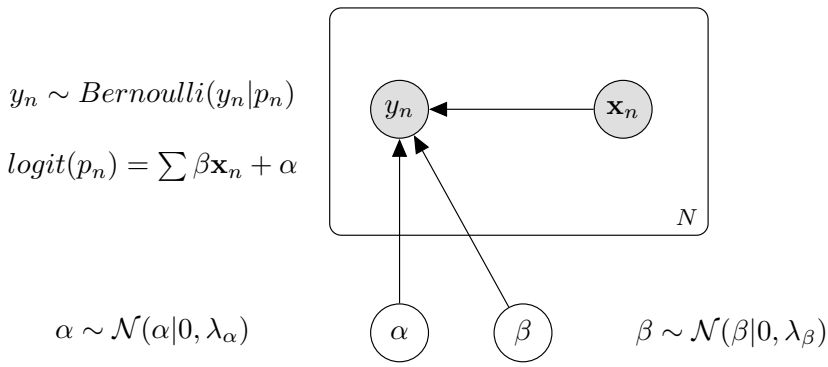

Figure 7.1: Probably graphical model of a Bayesian logistic regression, with observed input features ($\mathbf{x_n}$), treatment indicator (y_n), and latent variables as regression model coefficients (α, β).

and a variance of λ_α,

$$\alpha \sim \mathcal{N}(\alpha|0, \lambda_\alpha)$$

similarly, the regression coefficient β has a prior of Gaussian distributions of 0 mean and variance of λ_β,

$$\beta \sim \mathcal{N}(\beta|0, \lambda_\beta)$$

Bayesian networks are generative models and to generate the joint probability distribution of the regression model, the generative process is stated as the following Algorithm 2.

Algorithm 2 Bayesian logistic regression generative process

Inputs: $\mathbf{x}_n$ covariates, λ_α prior distribution of α, λ_β prior distribution of β, y_n control/treatment indicator

1: Draw $\alpha \sim \mathcal{N}(\alpha|0, \lambda_\alpha)$
2: Draw $\beta \sim \mathcal{N}(\beta|0, \lambda_\beta)$
3: **for** each vector of covariates x_i in $\{x_1, x_2, ..., x_I\}$ **do**
4: Draw $y_n \sim Bernoulli(y_n|Sigmoid(\alpha + \beta x_n))$

By Bayesian Theorem, the posterior distribution of the network is the product of the likelihood and the prior. In this case, the posterior distribution is a joint probability of y_n, α, and β marginalised over $p(y_n)$, that is,

$$\begin{aligned} &p(y_n, \alpha, \beta | \mathbf{x}_n, \lambda_\alpha, \lambda_\beta) \\ &= p(\alpha|\lambda_\alpha) \cdot p(\beta|\lambda_\beta) \cdot \prod_{n=1}^{N} p(y_n|\alpha, \beta, x_n) \end{aligned} \tag{7.6}$$

In many cases, the exact posterior distribution cannot be solved analytically, but it can be approximated with stochastic (e.g., Markov Chain Monte Carlo) or deterministic (e.g., variational inference) methods. In this paper, we approximate the posterior distribution through a stochastic method, the No-U-Turn Sampler (NUTS) in Hamiltonian Monte Carlo algorithm. Using a recursive algorithm, NUTS constructs a set of possible candidate point spans widely across the target distribution [77]. NUTS stops automatically if it retraced its steps, hence the name "No-U-Turn". We set up a NUTS sampler with a single chain, 3000 samples, and 200 warm-up samples were discarded. We include the model setup of the Bayesian logistic regression, the inference solver, and the trace plots in the online appendix.

Matching

After inferring the propensity score from the Bayesian logistic regression model, the second step of BPSM is to match the control and treatment pairs based on their propensity score distances. The objective of the matching is to form balanced control and treatment groups, that is, minimising the propensity score distance.

The propensity score distance (δp_n) is defined as the absolute difference of the propensity score in the control and treatment group,

$$\delta p_n = |p_{n,\tau=0} - p_{n,\tau=1}| \tag{7.7}$$

There are a few different methods for matching, such as calliper matching [78], 1:1, or n:1, nearest neighbour matching [54], and full matching [52], [69]. Matching can be done with or without replacement. When matching with replacement, one sample in the control group can be matched with multiple samples in the treatment group, and vice versa. Full matching method matches with replacement, such as the optimal full matching algorithm [69].

There are matching methods that do not allow sample replacement, and by

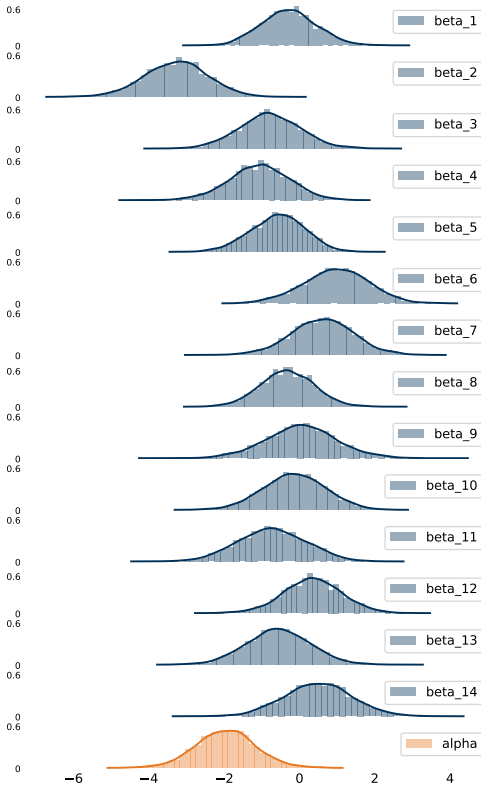

Figure 7.2: Posterior distributions of the Bayesian logistic regression coefficient $\beta = \{\beta_1, ..., \beta_{14}\}$, and intercept α.

using such matching methods, the matched control and treatment groups will have the same number of samples. Commonly applied methods include caliper matching, where the highest permitted calliper for δp_n is predetermined, and control and treatment pairs will be matched based on this calliper. Caliper matching is computationally cheap and intuitive [78], however, it could result in a reduction in number of samples in the treatment groups if the propensity score distances fall out of the predefined calliper. Sometimes it could be less ideal to reduce sample numbers in the treatment group, as they are usually considered as expensive samples especially in the embedded do-

main. Another matching method, 1:1 nearest neighbour matching, selects one controlled sample to match one treated sample with the smallest distance δp_n [54]. Using 1:1 nearest neighbour matching, no samples from the treatment group will be reduced and the sample reduction will only happen in the control group. In this paper, we apply both calliper matching and 1:1 nearest neighbour matching. As our objective is to find samples in the control group to be compared with the treated samples.

7.5 Results

In this section, the results of the Bayesian propensity score matching are presented. We show the propensity score computed from Bayesian logistic regression, along with matched groups from two different matching methods. Finally, we present the process of Bayesian propensity score matching for Observational Testing for evaluating software online.

Bayesian propensity score

Following the generative process described in 2, a Bayesian logistic regression model is implemented in Pyro [79]. The model takes covariates $\mathbf{x}_n$ and the prior distributions of α and β as inputs, and returns tensors of posterior distributions of α and β. A NUTS sampler in Hamiltonian Monte Carlo is used to infer the posterior distribution. We set up the sampler with a single chain, 3,000 samples, and 200 burn-in. Moreover, we apply a variational inference method to triangulate the results. The Brooks-Gelman-Rubin convergence criteria of $\hat{R} < 1.1$ is met, at $\hat{R} = 1.0003$. The variational inference uses a multivariate normal distribution as a guide. We define 40,000 steps for optimisation and the solver reaches a stable solution after the first 10,000 steps. Two inference methods return similar posterior distributions and point estimates. We show both methods in the online appendix attached. However, we will only focus on reporting the inference results from the NUTS sampler in this section.

Each regression coefficient β has a prior of $\beta \sim \mathcal{N}(0, \lambda_\alpha = 1)$, Gaussian distribution, and the regression intercept follows the prior distribution $\alpha \sim \mathcal{N}(0, \lambda_\beta = 1)$. Combining the priors, the posterior, $p(y_n, \alpha, \beta | \mathbf{x}_n, \lambda_\alpha, \lambda_\beta)$, is inferred from the observations $\mathbf{x}_n$ and the evidence y_n. We illustrate the

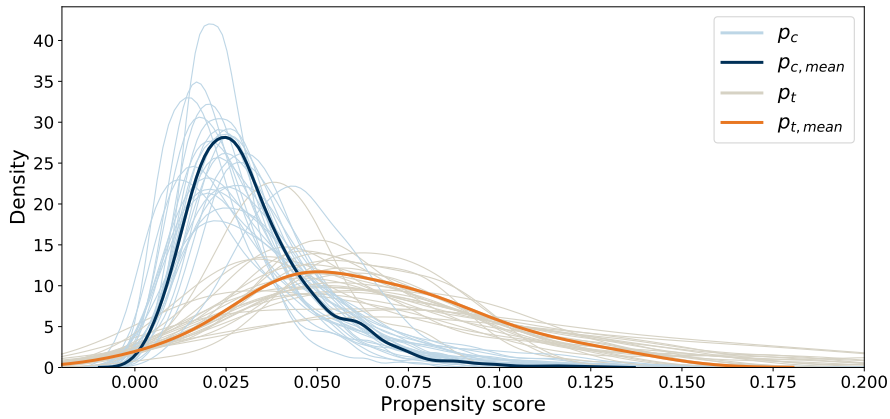

Figure 7.3: Kernel density distribution of the propensity scores of the control (p_c) and treatment (p_t) groups calculated on the mean of posterior distributions, and twenty-five values randomly sampled from the posterior distributions representing uncertainties.

total posterior distributions of all α and β in Fig. 7.2. From the posterior distributions, we can fit the logistic regression from a point estimate that is the mean value of the intercept $\hat{\alpha}$ and the coefficients $\hat{\beta}$, and the uncertainty of the model is quantified from the total posterior distribution.

The propensity scores for the control group (p_c) and the treatment group (p_t) are estimated as $e^{\hat{\alpha}+\hat{\beta}\mathbf{x}_n}/1+e^{\hat{\alpha}+\hat{\beta}\mathbf{x}_n}$. Without matching, the mean propensity score in the control and treatment group is 0.0319 and 0.0633 respectively. In each group, the standard deviation of the propensity score is 0.0175 and 0.0309. In Figure 7.3, we show the kernel density distributions of the propensity scores in the control and treatment groups before performing the matching. Since the entire posterior distribution is available, we illustrate the uncertainty on the propensity scores by randomly drawing 25 samples from the posterior distributions of the intercept and coefficients, and computing the propensity scores from the drawn α and β.

Matched A/B groups

After the Bayesian propensity scores are estimated, the second step it to match pairs from the control and treatment group to minimise the propensity score

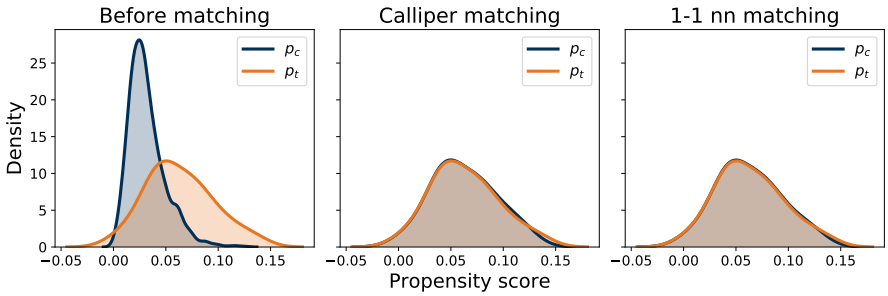

Figure 7.4: Kernel density distributions of the Bayesian propensity scores for the control (p_c) and treatment (p_t) group, when, no matching was done, matched with a caliper at 0.05, and matched with 1-1 nearest neighbour.

Table 7.2: Propensity score in control and treatment groups, before and after matching is applied.

Propensity score	Groups	Mean	Std.
Before matching	A	0.0319	0.0175
	B	0.0633	0.0309
Calliper matching (calliper = 0.05)	A	0.0626	0.0300
	B	0.0633	0.0309
1-1 nearest neighbour matching	A	0.0627	0.0302
	B	0.0633	0.0309

distance δp_n. A kernel density plot of the propensity score distribution before and after matching can be found in Figure 7.4.

Two matching methods are used, both methods find matches without replacement. The first matching method is a calliper matching, in which a maximum propensity distance is specified and matched pairs are produced if a treated sample has its corresponding pair in the control group. The number of samples in the control and treatment group is largely skewed in this dataset, using a calliper of 0.05, every treated sample returned a matched controlled sample. However, if there are too little controlled samples or if the calliper is determined to be too small, calliper matching could return no match for the treated samples. After calliper matching, the mean propensity score in the

control group is 0.0626.

The second matching method is a 1-1 nearest neighbour match. Similarly to calliper matching, 1-1 nearest neighbour match will return one-to-one matched pairs, but it does so without a specified range of propensity score distance. For each treated sample, the algorithm k-nearest neighbours searches for one closest neighbour from the control samples. The mean propensity score in the control group becomes 0.0627 after matching. We show the mean and standard deviations of the control and treatment propensity score in Table 7.2. On this dataset, both matching methods return similar outcome. Both methods find corresponding controlled samples for the treated samples. The average propensity score distance between the control and treatment group is 0.000757 and 0.000608 for the calliper matching and 1-1 nearest neighbour matching respectively. The covariates balance is assessed by comparing the empirical distribution of covariates in the control and treatment group. With a calliper matching, we found an average of 4.1 % reduction in the covariates variance compared to unmatched groups.

Treatment effect

The treatment, i.e., the new software, is expected to reduce the target variable fuel consumption. However, a number of other covariates could influence fuel consumption, such as temperature, trip frequency, trip distance, average speed, and etc. When the control and treatment group is not partitioned at random and with a large population, it is impossible to conclude a causal effect from the software change even a treatment effect is observed. Note that this focus of this study is not the actual software performance, thus, the results from this subsection serve as a demonstration.

The treatment effect is analysed using both the calliper matched and 1-1 nearest neighbour matched groups. The average treatment effect is calculated as the mean difference of the target variable between the control and treatment groups, and all 1138 measured values are min-max scaled. The average target variable is 0.379 and 0.391 for the control group when matched with calliper and 1-1 nearest neighbour, respectively. The average target variable is 0.355 in the treatment group. The average treatment effect is -0.024 and -0.036 for the control group when matched with calliper and 1-1 nearest neighbour, respectively.

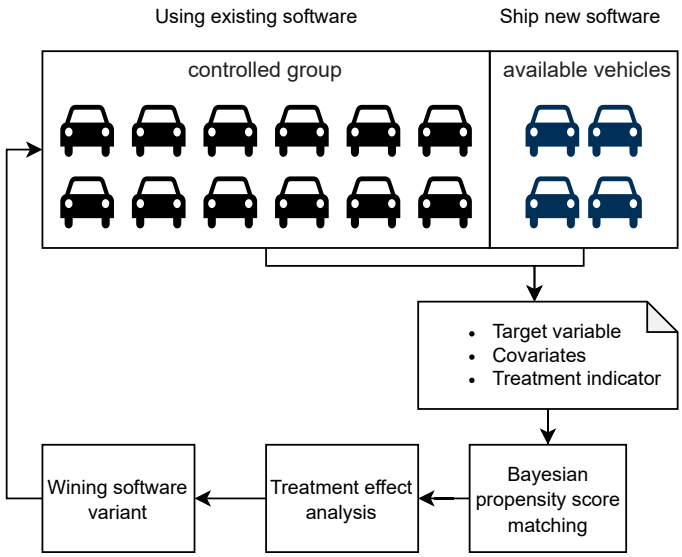

Figure 7.5: Online software evaluation with limited sample sizes, by utilising Bayesian propensity score matching.

Bayesian propensity score matching for observational test

In this subsection, we will describe in detail the process of utilising BPSM for evaluating software online with limited samples, as can be seen in Figure. 7.5. An observational testing, different from large randomised experiment because the partitioning of control and treatment group is not done at random, and such software online testing is usually done with a very limited sample group. Furthermore, unlike a pure observational study where no intervention is applied, we introduce treatment to a small cohort. As discussed previously, with the limitations of automotive embedded software, small-scale observational testing is often the only option in this domain. To utilise this model for evaluating software online when samples are limited and non-randomised, we recommend the following process.

Select treatment software and target variable

The target variable, that is, the metric of the software evaluation, should reflect the customer and business value the software is aiming to deliver. The target variable should be measurable. Moreover, in rare occasions, some software does not need online evaluation as this additional activity does not add more value to the product.

Determine covariates according to treatment

To produce balanced control and treatment groups, covariate selection is important as propensity score matching can only balance the variables that are included in the model. The decision on what covariates to be included shall be made both quantitatively and qualitatively. The optimal covariates should correlate to the target variable but not the treatment [68]. The strict statistical correlation between the covariates and the target variable is only part of the inclusion criteria. The qualitative domain knowledge of the software and its effect should be taken into consideration, especially when a new software is being evaluated and no high quality user data is available.

Eligible users for the control and treatment group

The treatment software will be shipped to a subset of users, often to users who have special user agreements in place. When selecting eligible users for the control group, one needs to make sure their existing software is comparable to the treatment software. The only systematic difference between the control and treatment group should be the applied treatment, else one could encounter confounding treatment effects from multiple software changes, i.e., a factorial treatment. In our study, we mitigate this issue by reading the software part number from all effected control units, and only include vehicles with the same part number in the control group. Additionally, in automotive, some software behaviours are heavily influenced by the devices and their operating locations. One can include users who drive a certain type of vehicle models in a given country, or include vehicle metadata as categorical variables in the covariates input to BPSM.

Data collection

After the sample groups have be determined and the new software is shipped, data collection for both groups starts simultaneously. The control and treatment vehicles are running in parallel so that seasonality effects can be mitigated. The required data collection infrastructure is already in place for our study. It is an important enabler for online evaluation of software and we recommend companies to implement data collection capabilities before running any online experiments. Additionally, a level of understanding of the physical machinery is required when collecting data from embedded systems. We suggest a cross-disciplinary approach when building such data pipelines for vehicle embedded systems.

Run Bayesian propensity score matching

When the online evaluation has been made and data collected, the Bayesian propensity score matching can be done. The propensity score is computed from all the covariates and the treatment indicator in a Bayesian logistic regression, and matching is done accordingly to the matching method and the propensity score. We implement the Bayesian logistic regression in Python Pyro, as can be found in the online appendix. The two simple matching methods used in the paper are implemented together with the case company. The code for the matching algorithms cannot be shared due to our confidentiality agreement, however, some matching algorithms are publicly available in R. Such as package optmatch[1] by [80].

Assess group balance and analyse treatment effect

After the control and treatment groups have been matched with their prospective propensity scores, an assessment of covariates balance should be done. The covariate balance can be accessed through the absolute standardised mean difference, which compares the absolute difference in means per unit of standard deviation. Moreover, the mean and variance of each covariate in the control and treatment group should be compared. Rosenbaum and Rubin [53] suggest an iterative process of diagnostics where additional covariates should be added after an assessment of the groups balance returned from the

[1] github.com/markmfredrickson/optmatch

initial propensity score model. The average effect is analysed by computing the average difference of the target variable between the matched control and treatment groups. The new software variant should be introduced if a treatment effect is detected and indicates an improvement.

7.6 Discussion

In this section, we present the threats to validity in our research and we discuss the generalisability of the Bayesian propensity score matching for observational testing model. Moreover, we share some known limitation to the Bayesian propensity score matching model, and what the limitations entail when BPSM is applied in online software evaluation in the automotive domain.

Threats to validity

The threats to validity of our research approach are presented in this subsection. In this paper, we present a proof-of-concept conducted with our case company on a software which optimises energy consumption of hybrid vehicles. In the treatment group with 38 vehicles, the vehicles are leased to company employees as their company cars and the users have explicit user agreements for participating such tests. The introduction of the new software variant is made aware to the users, however, we do not disclose the details of the software to them. Moreover, both the existing and new variants of the software are developed by the case company and we made no inputs to the software itself.

The set of signals measured are predetermined prior to our study, the development teams measure around 500 signals from vehicles, and our data features are engineered from a selected numbers of signals. We recognise that this means there is a slight risk, some confounding factors might not have been observed in the first place, and their effects on the target variable are unknown to us. In this study, no special action is taken to mitigate this risk as unobserved and unknown confounding factors should be considered as an inherent limitation of the propensity score matching model.

This study is done on one automotive manufacture and one software. We accept this limitation to this approach, as the results and conclusions might

not be applicable to all software developed by the same company or generalisable to the automotive domain.

First, the piece of software studied does not directly interact with users. There is no graphical interface, nor does it require user manual input. We have not explored how the propensity score matching model reacts to stochastic inputs such as user preferences. However, as reported by [17] from LinkedIn, propensity score matching model is used to support online user-facing software evaluations and shows promising improvements when the user groups are non-randomised. We foresee similar non-randomised user groups with preexisting preferences in the automotive setting. Furthermore, we demonstrate the BOAT method using quantitative data measured in the newest vehicles. We argue that such quantitative data is rather independent from the vehicle manufacture. Last but not least, we acknowledge companies within the automotive domain could follow difference processes for software development. Our proposed method that utilises small and non-randomised users for software online evaluation offers a high level of flexibility, and aligns with the core values of the Agile methodology that many automotive companies have adopted [2], [3], [10]. As agility is responsiveness to change [11], thus, we are optimistic of the value of enabling online software evaluation with small samples in a fast, safe, and ethical manner while maintaining causality.

Limitations

In this subsection, we discuss the limitation of the Bayesian propensity score matching for observational testing method. First, unlike a fully randomised control and treatment group split which can balance all covariates, propensity score matching can only balance the covariates that are observed. To make sure the ignorability assumption holds, including the correct covariates is important. But, when the software is new and there is limited usage data, it can be difficult to have comprehensive knowledge of which covariates should be included in the model. In this case, an iterative approach can be applied, in which covariates can be added or removed depending on the group balance [53].

Second, Bayesian inference is an expensive method in terms of modelling efforts and computational resources. The expense can be justifiable since Bayesian models are flexible as they allow prior input, and they are comprehensive as they return the entire posterior distribution instead of a point esti-

mate which can provide values in post-modelling analysis. Bayesian propensity score matching returns better results when sample sizes are small but does not show significant improvement as the sample sizes grow [46]. Therefore, whether to use Bayesian propensity score matching or regular propensity score matching should be a decision made based on the sample size, and a trade-off between computational expense and result improvement. Finally, we have addressed a scenario where two software versions are to be compared with propensity score matching in this paper, but in practise, multiple candidate software might need to be evaluated. Propensity score matching can be used for multilevel treatment effect modelling, however less straightforward, as reported by [81].

7.7 Conclusion

Online software evaluation is gaining attention in the automotive domain, but large-scale randomised experiments are not always an option with limitations in this industry such as safety and ethics. In this paper, we present an alternative method to randomise experiments so that online evaluation can be done on small sample groups, enabled with Bayesian propensity score matching model. This is the first paper to document such a model applied in automotive software engineering.

We describe the theory of Bayesian propensity score matching in detail and demonstrate the model with a proof-of-concept from an automotive company. In the study, we introduce a new software to a treatment group of 38 vehicles and the control group of 1100 vehicles use the existing software. The vehicles in the treatment group are leased to company employees. We observe both groups for a continuous five month period, during which we collected data from over 400,000 trips. Data collection is done through the vehicle sensors, and we produce 14 input features to the Bayesian propensity score matching model. Two matching methods were used, calliper matching and nearest neighbour matching. They produce similar results on our dataset and reduce the variance of the covariates by an average of 4.1%. Finally, we present the software engineering process of utilising Bayesian propensity score matching for evaluating new functions before shipping them to a larger group of users. This working method can be complimentary to Agile methodologies to enable responsiveness to change and to allow development teams making data-driven

decisions.

In our future work in the domain of automotive software online evaluation, we plan to continue to explore and apply different causal inference models. We see a potential in statistical models which enable online evaluations with limited sample size. Additionally, we plan to evaluate more automotive software, user-facing functions included, using causal inference methods and develop toolsets for modelling and analysis.

Online appendix

We attached an online appendix for the Bayesian logistic regression model. The online appendix can be found as a Jupyter Notebook via the following link: github.com/yuchueliu/BPSM.

Acknowledgement

This work is supported by Volvo Cars, by the Swedish Strategic vehicle research and innovation programme (FFI), by the Wallenberg AI Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Fundation and by the Software Center.

CHAPTER 8

Bayesian causal inference in automotive software engineering and online evaluation

The layout has been revised.

8.1 Introduction

Randomised online experiments, such as A/B testing, is a technique for evaluating the impact of software changes towards real users. With the demonstrated success of online experiment implementation in Software-as-a-Service (SaaS) companies [16], [21], [35], the automotive domain starts to raise interest in adopting such a method, and even starts to conduct experiments for evaluating embedded software online [3], [12], [71], [82]–[84]

Despite the known advantages and benefits, the automotive domain struggles to scale experimentation activities [3], [22], [82]. Some of the identified challenges are the limited number of users, limited capability of Over-the-Air (OTA) software deployment, strict user agreements, safety-critical and validation constraints among others. These limitations often create roadblocks that limit the scope and feasibility of conducting experiments in customer vehicles.

To overcome these limitations, practitioners are looking to leverage collected observational data to understand the causal impact of a software change [17].

In our previous study [83], we empirically applied and evaluated the use of causal modelling for software engineering, in which a Bayesian propensity score matching model is applied for generating balanced control and treatment groups in observational software testing. However, as we have experienced further needs in causal inference in the automotive domain due to a number of limitations, to address this need, this paper evaluates the use of three different Bayesian causal models for treatment effect inference from observational studies, applied to automotive software development. This work extends the method BOAT (**B**ayesian causal modelling for **O**vserv**A**tional **T**esting) [83] to include the Bayesian propensity score matching model for producing balanced control and treatment groups, the Bayesian regression discontinuity design for identifying covariate dependent treatment assignment, and Bayesian difference-in-differences model for causal inference on treatment effect overtime. While these models have been widely used in the frequentist setting in other domains of science (such as medicine [45], traffic and transport [46], social studies [47], [85]), this is the first paper to apply and evaluate these models in the Bayesian setting and in the context of automotive software engineering.

We demonstrate the BOAT method with three cases from our industrial collaborations, utilising automotive embedded software deployed on real vehicles and users. Comparing with the existing literature, the contribution of this paper is three-fold.

- We present an overview and discussion of potential outcomes and causality in automotive software development, along with three illustrative examples that reinforce the need for Bayesian causal inference in software online evaluation.

- We apply and evaluate three different Bayesian causal inference models to assess the causal effects in their corresponding examples. These models are the Bayesian regression discontinuity design, the Bayesian difference-in-differences, and the Bayesian propensity score matching.

- We relate the causal assumptions made in causal inference in relation to online observational studies conducted on automotive software, and we discuss their specific implications.

The rest of this paper is arranged as following. We elaborate the importance of causality in automotive software engineering and present the theory and assumption in the potential outcome framework in 8.2. In Section 8.3, we present the BOAT framework and our research method. The three Bayesian causal models and their related cases are described in Section 8.4, 8.5, and 8.6. The discussion and conclusion are in Section 8.7 and 8.8 respectively. Moreover, we include an online appendix to share our Bayesian models and their inference.

8.2 Background

In this section, we introduce the concept of randomised experimentation applied in software engineering, the potential outcome framework, and its relevant theories. Moreover, we give an overview of Bayesian statistics and its inference.

Randomised experimentation

Randomised experiment methods, such as A/B testing, are common practises adopted by SaaS companies [16], [21], [35]. In a two-level experiment, the sample group is split into control and treatment at random and exposed to different versions of the same software. When an experiment is fully randomised, the outcome is independent of the treatment assignment, this is defined as exchangeability. In other words, the control and the treatment groups are interchangeable and do not have any preexisting differences, thus the observed outcome can only be caused by the treatment. Therefore, randomised experiments help us at establishing a causal relationship between the intervention and the outcome.

Causal knowledge helps us cope with change [15]. Through data analytics, i.e., passive observation, we could compute a joint distribution of vehicle usage and performance – however, such a distribution cannot inform us if a change in our product would or would not improve the product performance and user experience. With intervention, randomised experimentation enables direct feedback from the users and helps organisations answer the "what-if" question to software changes. Randomised experiment has long been the gold standard for evaluating software in an online and continuous manner.

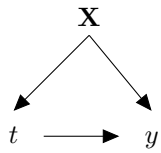

Figure 8.1: A simplified directed acyclic graph showing the relationships of treatment (t), target variable (y), and covariates ($\mathbf{X}$).

In the automotive domain, the ability to conduct large scale and fully randomised experiments is significantly more limited, as reported in [2], [3], [12], [22], [82]. The automotive domain faces many unique restrictions compared to SaaS companies, such as the number of hardware and software variants [3], architecture restrictions [71], safety regulation constraints, number of vehicles available for experimentation [82], driver consent, and the ability to frequently update software [3], [22]. A combination of these challenges leads to many situations where a randomised experiment is not: possible, such as in limited samples; desired, such as in highly regulated systems; or ethical, without explicit consent of the vehicle owners and users on the complete scope of the new software.

When a randomised experiment is not feasible, there present confounding factors that can often cause a spurious correlation and hinder us from drawing causal conclusions [15], [20], [36]. To address this issue, observational studies in combination with causal inference models and causal assumptions need to be applied. We will further discuss the empirical scenarios from the automotive sector and the underlying implications of causal assumptions on software observational testing in the following sections.

The potential outcomes framework

The potential outcome framework [36] describes causal inference from an intervention introduced in randomised experiments. In this section, we discuss the potential outcome from both experiments and observational studies, the later is extensively explored in studies such as [20], [37]. Potential outcome models quantify the treatment effect from the intervention introduced, or from known systematic differences between the control and treatment groups.

Let us consider an experiment in which we introduce two software variants to two groups, control (N_c) and treatment (N_t). We denote the two levels of

software variants as $t = \{0, 1\}$. For each individual in the sample population $n \in N$, we measure a target variable, y, to understand the possible outcomes, and a set of covariates $x \in \mathbf{X}$ that are predictive of the outcomes and potentially influence the treatment. The covariates $\mathbf{X}$, are also often referred to as context, or confounding factors. In practise, the two treatment levels of software could be $t = \{\text{old\_version}, \text{new\_version}\}$, for evaluating a change to an existing software. The potential outcomes of such an evaluation of an energy management software, could be reported as $y = [150, 300]$ measured in Wh/km.

Suppose we are interested in two treatment levels in a study, $t = \{0, 1\}$. In the Rubin potential outcome framework, the average treatment effect (ATE) can be expressed as,

$$ATE = \mathbb{E}[y|t = 1] - \mathbb{E}[y|t = 0] \tag{8.1}$$

where the $\mathbb{E}(y)$ represents the expectation of the outcome from the samples at different treatment levels. In order to infer the treatment outcome, an important assumption made in the potential outcome framework, is the stable unit treatment assumption, stating that the treatment only effect the individual sample the treatment is applied to.

Note that, when the experiment is randomised, the treatment outcome is unconditional to the control and treatment group assignment. In other words, the two groups are exchangeable. Therefore, we can explicitly establish a causal relationship between treatment and the potential outcome in a randomised experiment. Exchangeability is expressed as,

$$[y|t = 0], [y|t = 1] \perp t \tag{8.2}$$

where $\perp$ denotes independence, and $|$ means given the condition, $y|x \perp t$ reads as y is independent of t given x.

The potential outcome and/or treatment assignment in an observational study is influenced by covariates $\mathbf{X}$. In a trivial example, all covariates influence both the treatment assignment and the outcome, a confounding effect. We illustrate such an example in a directed acyclic graph (DAG) in Fig. 8.1, to infer causality from this DAG, a valid adjustment set of covariates should block every path from the treatment t to the target variable y in the DAG. If such a set of covariates exists and can be identified in an observational

study, we assume the exchangeability persists, and it is conditional on the adjustment set ($\mathbf{X}$) given the set of covariates can be observed and identified. Formally,

$$[y|t=0],[y|t=1]\perp t|\mathbf{X} \tag{8.3}$$

In an observational study, it requires that there shall be treated and untreated samples in every combination of the values of the observed confounding factors $\mathbf{X}$ [43], the positivity assumption, is formally expressed as,

$$0 < P(T=t|\mathbf{X}=x|) < 1 \tag{8.4}$$

In addition, covariates can be used to estimate the conditional average treatment effect ($CATE$) in observational studies, namely,

$$CATE = \mathbb{E}[y|t=1,\mathbf{X}=x] - \mathbb{E}[y|t=0,\mathbf{X}=x] \tag{8.5}$$

The inference of conditional average treatment effect is helpful in heterogeneous studies. For example, we can study the treatment effect in subgroups of vehicle models or locations, provided the covariates heavily influence the treatment outcome.

Bayesian statistics and inference

A short overview of Bayesian statics and inference methods will be provided in this subsection. To put the Bayesian causal inference models in context, we present the the basic principle of Bayes' theorem and inference methods used in this study. However, we do not provide a comparison of frequency and Bayesian statistics, as the difference in reasoning is beyond the scope of this paper and we refer such a comparison to other works in software engineering [76], [86], [87].

In Bayesian statics, the probability of an event is expressed as a degree of belief which is based on the prior knowledge of said event. The intuition of Bayesian statistics is that the degree of belief is updated by observing new data, evidence, and the sensitivity of the outcome to the prior reduces as more observations are made. In the application of causal inference, applying Bayesian statistics allows us to incorporate available prior knowledge on model parameters when inferring the counterfactual outcome [88]. The Bayes'

theorem is expressed as,

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{8.6}$$

where,

- $P(A)$ is the prior, the probability of a hypothesis before any evidence is observed and presented. This is often referred to as domain knowledge.
- $P(B)$ is the likelihood, the probability of observing the evidence.
- $P(B|A)$: is the probably of observing the evidence given the prior.
- $P(A|B)$: is the posterior probability given the evidence, the observation, and the prior.

In most cases, the exact posterior distribution of the model parameters cannot be solved analytically, but it can be approximated numerically with Markov Chain Monte Carlo (MCMC) or variational inference methods. In this paper, we approximate the posterior distribution through the No-U-Turn Sampler (NUTS) in Hamiltonian Monte Carlo algorithm. Using a recursive algorithm, NUTS constructs a set of possible candidate point spans widely across the target distribution [77]. NUTS stops automatically if it retraced its steps, hence the name "No-U-Turn".

The prior distributions are an integral part of Bayesian model and they allow researchers the flexibility of incorporating domain knowledge of previous research to create better and more robust models. The priors often act as constraints of plausible probabilities of parameter values. With small sample sizes, the prior or the domain knowledge has a higher influence. While with larger sample sizes, the evidence overcomes the impact of the prior in the posterior. Priors can be specified to be *non-informative*, *weakly informative*, and *informative* for their models. A non-informative prior is based on an unbounded uniform distribution and does not aggregate any information to the posterior and are often non proper. Weakly-informative are those that do not aggregate much information in the posterior parameters and serves as regularisers in the inference and convergence of the MCMC solver. An example of such a prior would be a normal distribution with a large variance compared to the expected parameter value. Finally, informative priors are those that incorporate domain knowledge on the subject and set stricter bounds to parameters

in the model. If the evidence is accordance with the prior, convergence happens faster due to the smaller search space for the MCMC solver. If evidence points out to a parameter outside these bounds, either domain knowledge or data collection should be revised and convergence might be slow.

8.3 The BOAT framework

In this section, we present an alternative approach to randomised experimentation in software engineering, BOAT (**B**ayesian causal modelling for **O**vserv**A**tional **T**esting). In this framework, we combine the notion of quasi-random treatment assignment with data obtained from pure observations, aiming to address the situations where a fully randomised experiment is not feasible. Different from an observational study, in which the treatment is inferred from known systematic differences of the control and the treatment groups, our method allows one to actively intervene with a treatment group that is not randomly sampled. This framework enables development organisations to evaluate their software online without the need of a fully randomised large scale experiment.

BOAT

The BOAT framework is induced from the potential outcome theory, and is applied and validated through exemplary cases with our industry collaborator. We describe the framework in detail and the research method applied for the validation of the BOAT framework. A fully randomised experiment is often challenging to conduct in the automotive domain, in the absent of randomisation, it requires a series of causal modelling techniques to mimic randomisation or to adjust covariates before a treatment effect can be inferred. We list the challenges in randomisation and their corresponding solutions in the BOAT framework as the following.

The first challenge in adopting online experiment is the limited access to the entire user base. To start, the automotive domain has a significantly smaller user based comparing to the SaaS domain, as a result of product diversity and hardware dependency [3], [82]. Moreover, in combination with the limitation of safety-critical software and the lack of explicit user agreements, shipping new software to the entire fleet is typically undesired, impossible, or

unethical. As a result, the control and the treatment groups are likely to be unbalanced and the treatment effects are confounded by one or more unbalanced covariates. In situations when the treatment effect is confounded by more than one covariate, it is impossible to balance the control and treatment groups by stratification, therefore, a propensity score needs to be modelled from all covariates included in the system. Propensity score matching, first proposed by Rosenbaum and Rubin [20], is a method for matching samples from the control and treatment group based on the propensity score calculated from observed covariates, thus adjusting for covariates and estimating unbias treatment effects. The Bayesian propensity score matching model is used for designing a balanced control and treatment group in traffic safety analysis [46] and in automotive software engineering [82].

Second, the performance of automotive software functions are often heavily influenced by temporal factors such as weather and time of week. Such a seasonality effect can be observed in software functions related to energy consumption [89] and crash safety [90], [91]. In practise, if we want to evaluate an energy management software for battery electric vehicles, suppose we compare the energy efficiency before and after the software change on the same vehicles, the conclusion could be bias and confounded by unobserved temporal factors, e.g., temperature. To address this particular issue, we suggest to apply difference-in-differences model in the Bayesian framework. This model is first presented by Card and Krueger [40] in analysing the treatment effects of increasing the minimum wage. In our model, we include a control group of vehicles running on the old software version through observation. Therefore, any bias caused by factors common to the control and treatment is implicitly controlled for, even when the confounders are not observed. Besides econometrics, Bayesian difference-in-differences model is applied to analyse the treatment effect over time for diabetes patients [45].

Third, we see a need of modelling and analysing software studies where the treatment assignments depend on one continuous covariate. Since many software functions in vehicles are only activated or beneficial for users around a certain threshold of a variable, such as speed or trip distance. This is the case when we are analysing the fuel saving potential from route prediction of plug-in hybrid vehicles, as the trip distance heavily influences the prediction accuracy and the fuel saving potential. It is believed that the software is particularly beneficial, when the driver travels slightly further than the pure

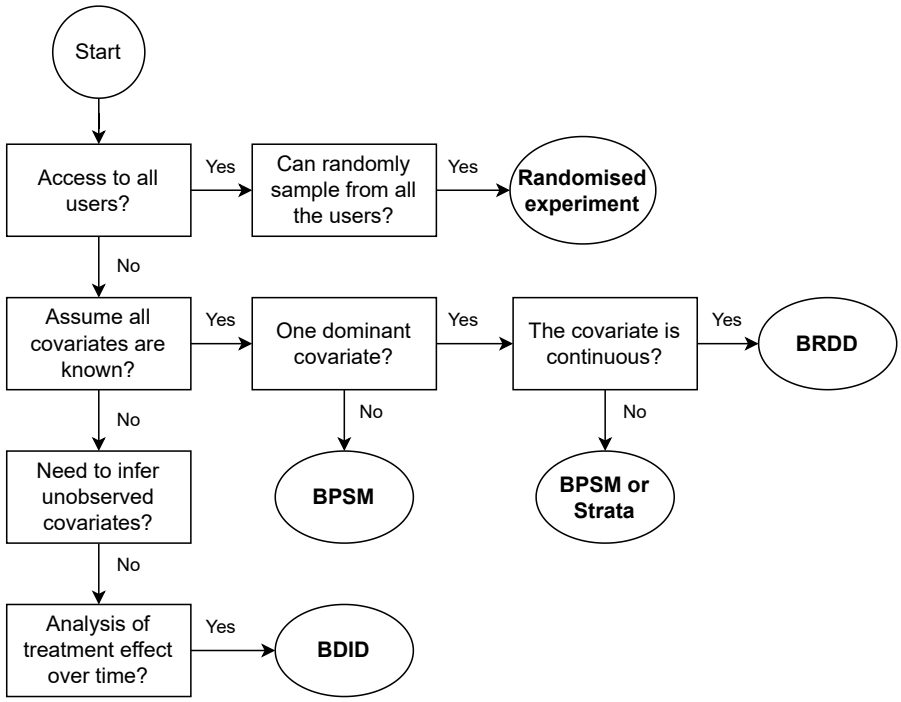

Figure 8.2: A decision flowchart on which Bayesian causal model from the BOAT framework to apply when designing an online software evaluation. (BRDD: Bayesian regression discontinuity, BPSM: Bayesian propensity score matching, BDID: Bayesian difference-in-differences)

electric range of the vehicle on the daily basis. In this scenario, the regression discontinuity design [41] could help us model the treatment causal effect through identifying a threshold of an assignment covariate where the treatment is mostly influenced. Bayesian regression discontinuity design is applied in other areas of science such as economics [92] and medicine [93].

To further illustrate the use cases of the BOAT framework in automotive software engineering, we provide a flow chart in Fig. 8.2. When designing a software online evaluation, the first assessment criteria is the available sample size determined by a power analysis of expected size of the treatment effect. In the scenario of all users can be accessed and a randomised sampling process can be done, a fully randomised experiment is always more ideal for a

strong causal conclusion. Since randomisation for sampling from the total user base not only ensures exchangeability, it also ensures representativeness thus removing sampling bias. The second step is to determine if we can safely assume all the covariates and their relationship to the expected outcome is known and observable. Under this assumption, there is a need to identify if there is one or more covariates, since the balancing of a single and categorical covariate can be achieved through stratification as well. In situations where there are more than one categorical covariate, a Bayesian propensity score matching (BPSM) shall be performed to design a balanced control and treatment group. In the case of one continuous and dominating covariate, the design calls for the use of a Bayesian regression discontinuity (BRDD) model, which utilises the continuous covariate as an assignment variable for assigning samples to the control and treatment group. If the assumption of known covariates cannot be made, which is often the case when evaluating a novel software functionality, or conducting a longitudinal evaluation. In the former scenario, there is usually no available data to analyse the causal structure, and in the latter case, the number of covariate required for identification quickly scales as the study is conducted during a prolonged period of time. The development organisations need to decide if the unobserved (latent) variables need to be modelled and inferred, if not, a Bayesian difference-in-differences (BDID) model can be applied. BDID is an effective strategy to infer treatment effect overtime, without the need of observing any time dependent covariates.

Although the BOAT framework provides a structured guidance for software development organisations for their design decisions of online software evaluations, and to a large extend, enables such online evaluation that is otherwise challenging or impossible particularly in the automotive domain. We recognise that the framework is not in anyway complete. For example, if the causal relation is unknown, a causal discovery process [94] or a graphical model is needed [84]. To that end, if a latent variable is deemed to be important and needs to be modelled, methods such as instrumental variable [95] can be applied. We will further discuss the limitations and the potential extension of the BOAT framework in the discussion section.

Research Method

To validate our proposed BOAT framework and its applicability in automotive software engineering, we employ the design science research method following

guideline from [96]. The BOAT framework, can be considered as a design artifact crated to address an important research and organisational problem – in the absence of randomisation, how do we evaluate software in an online fashion and infer causality. The relevance of the problem is assessed and addressed through the challenges of randomisation experienced in practicse. We list the practices of this research following guidelines from [96] in detail in 8.3. In 8.3, we replace the original description from [96] of the step-wise guideline with the approaches taken in our research activities.

Table 8.1: Guidelines of design science research method [95], and practices applied following the guidelines in this research.

Guideline	**Practise in this research**
Guideline 1: Design as an artifact	We present the BOAT framework to three software development organisations, the framework addresses varies limitations in randomised online experiments, assess practical scenarios to provide Bayesian causal modelling suggestions. The framework is derived from the theory of potential outcome, and all of the modelling approaches within the framework are extensively applied and validated in other areas of science [20], [45]–[47], [52], [66], [85].
Guideline 2: Problem relevance	To ensure the technical solution developed is relevant to the domain, we derive the problem from existing literature addressing the challenges on online experimentation adaptation in automotive [3], [12], [71], [82], [83], and literature stating the challenges of online experimentation in other domains [16], [21], [35]. All of the literature included in the analysis are based on empirical research in their respective domains. Additionally, the scenarios that limit randomised experimentation are also experienced and reported by our industry collaborator.
Continued on next page	

Table 8.1 – continued from previous page

Guideline	Practise in this research
Guideline 3: Design evaluation	The evaluation of the BOAT framework is done quantitatively through three separate empirical cases. The empirical study are designed together with development organisations as suggested by [51], [52], [66], and deployed to a selective number of customer vehicles to simulate three scenarios of online software evaluation in the absent of randomisation. The quantitative data is collected from the vehicles through telecommunication units onboard. We determine the target variable and the covariates together with the development teams which also developed the software changes. Additionally, we assess the validity of the causal assumptions in practise with domain knowledge provided by experts from the development organisations.
Guideline 4: Research contributions	Not to be confused with the research contribution of a publication, the research contribution in design science is assessed by implementability and representational fidelity. The former criteria is satisfied as there are commonly available tools for computing the the causal models, as well as the authors of this paper have implemented the code in Pyro [79]. The later is ensured through the close collaboration with an automotive manufacturer.

Continued on next page

Table 8.1 – continued from previous page

Guideline	**Practise in this research**
Guideline 5: Research rigor	Conducting research with design science often requires mathematical formalism to describe the design artifact [95], [97]. We formulate the Bayesian causal models and assumptions in the BOAT framework mathematically, as well as providing an algorithmic description for model implementation. In the BOAT framework, the relationships between factors within the system is assumed to be known; the inputs, such as covariates, and the outputs are selected based on the domain knowledge. Claims about the quality of design artifact is dependent on the choice of performance metrics. Therefore, we evaluate the causal models following advice from existing literature introducing the models to other areas of science [20], [40], [41].
Guideline 6: Design as a search process	As a research method, design science is iterative and a way to discover an effective solution to the problem. In our research, we maintained close collaboration with the development teams through weekly design meetings, in which we discuss all aspects of the cases including the potential confounding factors, the expected treatment effects, and etc. The selection of covariates is done in an iterative manner to ensure the covariates with strong correlation to the target variable is included in the model. Moreover, the scenarios addressed in the BOAT framework is derived from literature [3], [22] and validated from the state-of-practise in automotive software engineering.

Continued on next page

Table 8.1 – continued from previous page

Guideline	Practise in this research
Guideline 7: Communication of research	The results of our research is communicated in the following three ways; (1) we host popularised science presentations for our industry collaborators on a regular basis every three month. The participants of these presentations usually occupy managerial positions such as product manager, project manager, and product owner. (2) we communicate the BOAT framework, the implemented Python code, and the results to the development organisations through our weekly meetings, during which we also provide tutorials of the Bayesian modelling approach. Participants are usually developers and data scientists alike. (3) we communicate our work to the scientific community through publications.

The successful adoption of a new framework in software engineering requires the framework to be investigated in the business organisations it is applied in, therefore, together with our industry collaborator, we design and validate the framework empirically with three software development organisations. The industry collaborator is an automotive manufacturer with operation in Europe, China, and North America, and this research is part of a long term collaboration. Additionally, to evaluate the Bayesian causal models, we deploy software and collect empirical data from a total of 1,364 vehicles driven by real-world customers. The vehicle data collection took place during an nine-month period, between December 2020 to September 2021. The detailed setup of the three empirical cases are discussed in their corresponding sections.

8.4 Bayesian propensity score matching

In this section, we present the theory, our observational study, and the results from the Bayesian propensity score matching (BPSM) model. The theory and assumption of the model is presented formally, followed by a discussion

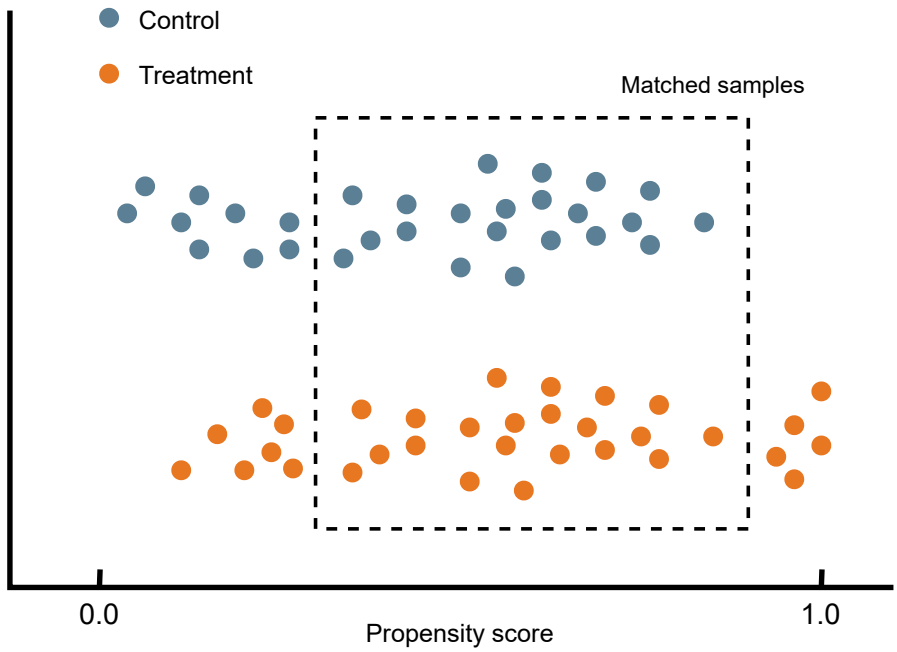

Figure 8.3: An illustration explaining the Propensity Score Matching model. Note, figure does not represent real data.

on different matching strategies. Finally, we introduce the setup of our observational study utilising BPSM as an identification strategy and we present the results.

To estimate the unbias treatment effect in an observational study is challenging, as the treatment assignment and effect could be confounded by covariates. We use the illustration in Fig. 8.1 to demonstrate this scenario. Propensity score matching [20] addresses this issue through covariate adjustment and allows us to generate balanced control and treatment groups even when the sample size is limited. The covariates from matched control and treatment group should form similar empirical distribution, thus reducing bias in the estimated treatment effect. We illustrate the concept of propensity score matching in Fig. 8.3, as can be seen, samples are matched based on their propensity score similarity, also called propensity score distance. In

previous literature, propensity score matching has been used for experiment design when the sample size is small [52], [54], [66], [82], and for causal effect analysis *post facto* [17], [46]. Propensity score matching is done through a Bayesian network, which is reportedly less sensitive to sample sizes [46].

Theory

Propensity score, denoted as $e(x)$, is a function of one or more covariates $x \in \mathbf{X}$. Propensity scores are modelled and matched in the control ($t = 0$) and treatment ($t = 1$) groups, so that the conditional probability distribution of the covariates given the propensity score $p(x|e(x))$ is similar in both groups [20]. The strong exchangeability assumption, extends from the conditional exchangeability, stating that the control and treatment outcome pair is assumed to be independent from treatment assignment, given the observed covariates, formally,

$$([y|t = 0], [y|t = 1])\perp t|\mathbf{X} \tag{8.7}$$

This assumption implies that the treatment or the outcome is only confounded by observed covariates and the covariates that are ignored from the data do not effect the treatment or outcome. The average treatment effect identified through propensity score matching (ATE_{PSM}) is conditional to the treatment t and the propensity score inferred from all observed covariates, $e(\mathbf{X})$. Given the assumption holds, the average treatment effect adjusted with the propensity score is an unbias estimate of the average treatment effect,

$$ATE_{PSM} = \mathbb{E}[y|t = 1, e(\mathbf{X})] - \mathbb{E}[y|t = 0, e(\mathbf{X})] \tag{8.8}$$

There are two steps in propensity score matching. First, we estimate the propensity score through a Bayesian logistic regression, then we perform the matching of samples based on their propensity score distance. We present the two steps separately in the following subsections.

Bayesian logistics regression

In a BPSM, the propensity score is estimated with a Bayesian logistic regression. Sticking to the same notations, the treatment indicator t, follows a Bernoulli distribution,

Algorithm 3 Bayesian logistic regression generative process

Inputs: $\mathbf{X}$ covariates, λ_α prior distribution of α, λ_β prior distribution of β, t_n control/treatment indicator

1: Draw $\alpha \sim \mathcal{N}(\alpha|0, \lambda_\alpha)$
2: Draw $\beta \sim \mathcal{N}(\beta|0, \lambda_\beta)$
3: **for** each vector of covariate $x \in \mathbf{X}$ **do**
4: Draw $t \sim Bernoulli(t|Sigmoid(\alpha + \beta x))$

$$t \sim Bernoulli(t|e(\mathbf{X})) \tag{8.9}$$

where the propensity score $e(\mathbf{X})$ is expressed as,

$$e(\mathbf{X}) = \frac{e^{\alpha+\beta\mathbf{X}}}{1 + e^{\alpha+\beta\mathbf{X}}} \tag{8.10}$$

The regression intercept and coefficients, α and β are latent variables. That is, they are not directly observed but inferred from other variables $\mathbf{X}$. We normalise the prior Gaussian distributions for the regression intercept α, as a result, this prior distribution has a 0 mean and a variance of λ_α,

$$\alpha \sim \mathcal{N}(\alpha|0, \lambda_\alpha)$$

similarly, β has a Gaussian distributions of a 0 mean and variance of λ_β as prior,

$$\beta \sim \mathcal{N}(\beta|0, \lambda_\beta)$$

By Bay's theorem, the posterior distribution of this network is simply the product of the likelihood and the prior. Therefore, for all samples $n \in N$, the posterior distribution is a joint probability of t, α, and β marginalised over $p(t)$, that is,

$$\begin{aligned} &p(t, \alpha, \beta|\mathbf{X}, \lambda_\alpha, \lambda_\beta) \\ &= p(\alpha|\lambda_\alpha) \cdot p(\beta|\lambda_\beta) \cdot \prod_{n=1}^{N} p(t|\alpha, \beta, \mathbf{X}) \end{aligned} \tag{8.11}$$

Bayesian networks are generative models, to generate the joint probability distribution of the regression model, the generative process is stated in

Algorithm 3.

Matching

The second and final step of BPSM is matching samples from the control and treatment groups based on their propensity score distance, and to minimise the average distance for the two groups. The propensity score distance ($\delta e(\mathbf{X})_n$) is defined as the absolute difference of the propensity score of each sample (n) in the control and treatment group,

$$\delta e(\mathbf{X})_n = |e(\mathbf{X})_{n,t=1} - e(\mathbf{X})_{n,t=0}| \tag{8.12}$$

Many matching methods have been explored in the literature, calliper matching [78], 1:1, or n:1 nearest neighbour matching [54], and full matching [52], [69], just to name a few. In general, there are two categories of matching methods, with or without replacement. Matching can be done with or without replacement. Matching with replacement means one sample in one group can be matched with multiple samples in another group, an example of such method is the optimal full matching algorithm [69].

By using matching methods without sample replacement, the matched control and treatment group will yield the number of samples. Calliper matching is a type of matching method, in which a maximum allowed $\delta e(\mathbf{X})_{n,max}$ is predetermined and all samples exceeding the threshold are discarded. Although calliper matching could result in a reduction on sample size if the calliper is too fine, it is an intuitive and computationally efficient matching method [78]. Moreover, the choice of calliper can effect the result bias [98]. As treated samples are usually more expensive to obtain, discarding those samples could be considered unfavourable in the automotive application. Therefore, a second matching method is explored in the study, 1:1 nearest neighbour matching [54]. In a 1:1 nearest neighbour matching, the algorithm looks for a control sample with the closest propensity score distance to a given treated sample, thus, no treated samples will be discarded. In this study, both calliper matching and 1:1 nearest neighbour matching are applied, as our objective is to find control samples to be compared with the treated samples.

Study I: Limited access to users

In the automotive domain, a fully randomised experiment on a large scale is often impossible due to their already limited numbers of users and it is often undesired to serve new software on the entire fleet. When a safety critical software is the subject of interest – a majority of automotive software are in safety critical systems – introducing a novel software feature to a larger number of vehicles might not be desirable; although the likelihood of catastrophic failure is low [3], but any minor disturbances at a scale could still cause profit loss for commercial vehicles. Combine this with the fact that we often only want to examine software features on a specific vehicle model driven in a specific region, it further limits the available sample size. As an alternative to large scale randomised experiments, we propose a small-scale rollout to a limited number of vehicles. However, when the control and treatment groups are no randomised, the exchangeability assumption does not hold and the observed change in the target variable could be confounded on preexisting differences between the groups instead of the treatment itself. Thus, we apply Bayesian propensity score matching for matching comparable treated and untreated vehicles based on a number of observed covariates.

We design a study to simulate this scenario and learn the feasibility of applying BPSM as an identification strategy in observational studies such as this. In Study I, we aim to analyse treatment effects from an energy management software. Based on how the vehicle is historically driven, this software predicts the current trip proprieties such as expected route, and based on the prediction, energy consumption is optimised through actions such as downshifting before a hill climb. This software was difficult, if not impossible, to validate in a lab-like environment as the energy saving potential is strongly dependent on the prediction accuracy, however, since the software feature involves safety critical systems such as engine control, it is undesirable to introduce it to a large group of users for a fully randomised experiment. Instead, we passively observe 1100 vehicles in the control group running on an existing software driven by real-world customers, while only serving the modified software to 38 vehicles as our treated samples. The treated samples, are vehicles leased to employees whom use the vehicles as their regular cars. This study occurred from October 2020 to March 2021, and all vehicles are driven by users reside in Sweden. We discard data generated by brand new vehicles with mileage less than 100 kilometres, and trips with average speed

higher than 200 kilometres per hour. After post processing, we have collected data from a total of 421,881 trips made by 1138 vehicles.

A fundamental assumption of applying propensity score as an identification strategy for observational study is the strong exchangeability assumption (often called ignorability [20]), implies that the treatment outcome is not dependent on unobserved covariates. This strong assumption cannot be checked empirically from pure observational data. Essentially, the assumption implies that all the confounders that potentially influence the outcome are observed and it inherently limits us to draw causal conclusion when no covariate is known. In other words, propensity score matching can only balance covariates that are observed, while full randomisation can balance all covariates, observed or not [66]. In practice, to design a study utilising BPSM requires existing data or knowledge of the software systems, as the results are strongly dependent on covariate selection [68]. To this end, an iterative procedure can be applied when selecting covariate and performing BPSM [53]. In our study, we include a total of 14 covariates in the final BPSM model as a outcome of domain knowledge provided by our industry collaborator and two iterations of model design. We present the descriptive statistics of the covariates included in the final model in Table. 7.1.

Results

In this subsection, we present the results from Study I. First, we show the propensity score inferred from a Bayesian network. Second, we present matched control and treatment group with the two matching strategies discussed in the previous section.

We implement a Bayesian logistic regression in Pyro [79] following the generative process described in Algorithm 3. We set up a NUTS sampler for inference to infer the posterior distribution with a single chain. We generate 3,000 samples of which 200 burn-in. The Brooks-Gelman-Rubin convergence criteria of $\hat{R} < 1.1$ is met, at $\hat{R} = 1.0003$. To triangulate the results, a variational inference model is used. We use a multivariate normal distribution as a guide for the variational inference model. We define 40,000 steps for optimisation and the solver reaches a stable solution after the first 10,000 steps. Two inference methods return similar posterior distributions and point estimates. We show both methods in the online appendix attached and we will only focus on reporting the inference results from the NUTS sampler in this paper.

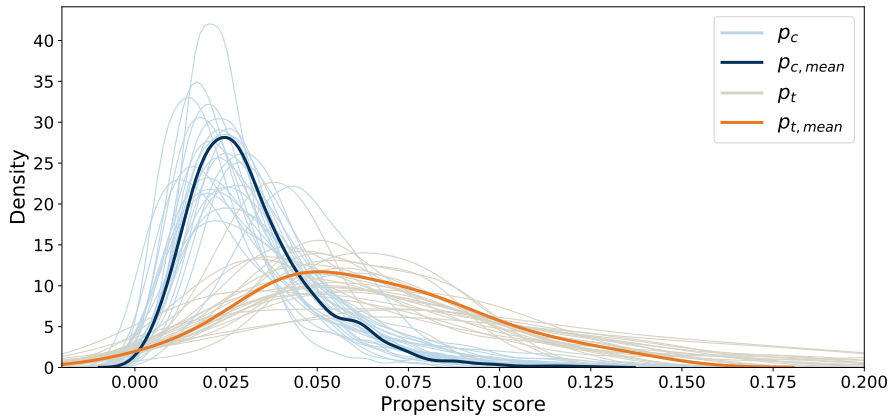

Figure 8.4: Kernel density distribution of the propensity scores of the control (p_c) and treatment (p_t) groups calculated on the mean of posterior distributions, and twenty-five values randomly sampled from the posterior distributions representing uncertainties.

We assign a prior distribution $\beta \sim \mathcal{N}(0, \lambda_\alpha = 1)$ to each regression coefficient β, and similarly, $\alpha \sim \mathcal{N}(0, \lambda_\beta = 1)$ to the regression intercept α. Combining the priors, the posterior, $p(t, \alpha, \beta | \mathbf{X}, \lambda_\alpha, \lambda_\beta)$, is inferred. Then, the propensity score $e(\mathbf{X})$ is calculated follow Equation. 8.10, before matching, the mean propensity score in the control and treatment group is 0.0319 and 0.0633 respectively. In each group, the standard deviation of the propensity score is 0.0175 and 0.0309. We also quantify the uncertainty of the propensity score from the posterior distribution. A visualisation of the propensity score distribution before a matching is performed can be seen in 8.4, in which we plot the propensity score distribution in the control (p_c) and treatment (p_t) computed from the mean point estimates as well as random samples from the posterior distribution of the regression intercept and coefficients.

A matching based on propensity score distance $\delta e(\mathbf{X})_n$ is performed after the scores are computed. In this paper, we perform matching with 1:1 nearest neighbour and caliper matching method, the results are presented in Table 8.2. As can be seen from the table, both matching methods return similar outcome in this study, the mean propensity score distance is 0.000757 and 0.000608 for the calliper matching and 1-1 nearest neighbour matching respectively. First,

Table 8.2: Propensity scores in control and treatment groups, before and after a matching is performed.

Propensity score	Group	Mean	Std.
Before matching	Control	0.0319	0.0175
	Treated	0.0633	0.0309
Calliper matching (calliper = 0.05)	Control	0.0626	0.0300
	Treated	0.0633	0.0309
1-1 nearest neighbour matching	Control	0.0627	0.0302
	Treated	0.0633	0.0309

a calliper matching method is used with a specified maximum propensity score distance, $\delta e(\mathbf{X})_{n,max} = 0.05$, and every treated sample returned a matched control sample. The mean propensity scores in the control group were 0.0626 after applying caliper matching.

Second, we apply 1-1 nearest neighbour matching method. Similarly to calliper matching, 1-1 nearest neighbour match will return one-to-one matched pairs, but the matching is done without specifying maximum allowed propensity score distance. The algorithm k-nearest neighbours searches for one closest neighbour from the control samples to the treated samples. The mean propensity score in the control group is 0.0627 after 1-1 nearest neighbour matching. Both methods find the corresponding control samples for the treated samples. The covariates balance is assessed by comparing the descriptive statistics such as variance and the empirical distribution of covariates in the two groups. With a calliper matching, we found an average of 4.1 % reduction in the covariates variance compared to unmatched groups.

The treatment effect is analysed for both the calliper matched and 1-1 nearest neighbour matched groups. The average treatment effect is calculated as the mean difference of the target variable between the control and treatment groups, and all values are min-max scaled. The average target variable is 0.379 and 0.391 for the control group when matched with calliper and 1-1 nearest neighbour, respectively. The average target variable is 0.355 in the treatment group. The average treatment effect is -0.024 and -0.036 for the control group when matched with calliper and 1-1 nearest neighbour, respectively.

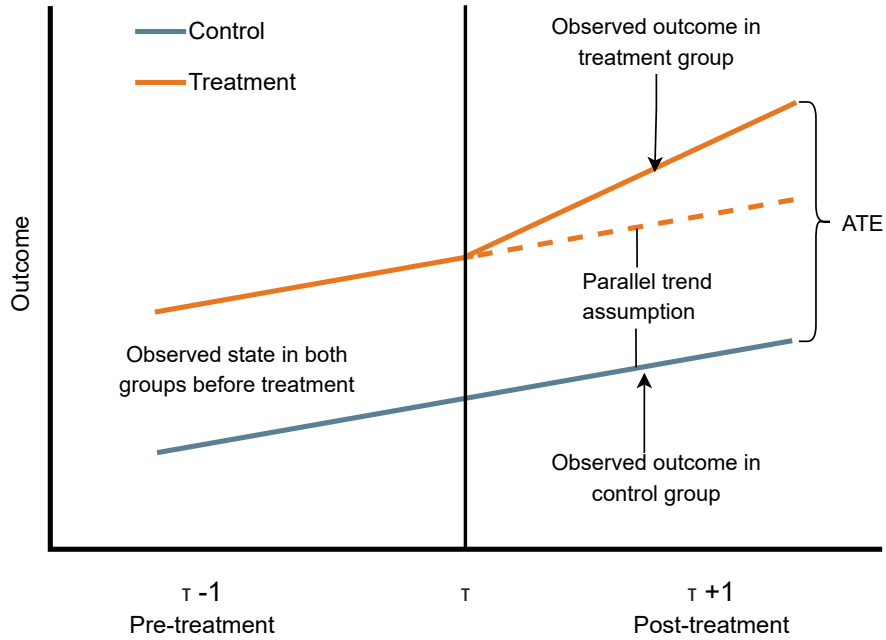

Figure 8.5: An illustration explaining the Difference-in-Differences model and how the average treatment effect (ATE) is estimated.

8.5 Bayesian difference-in-differences

In this section, Bayesian difference-in-differences (BDID) theory and our observational study is presented. We describe the theory and assumptions in the model formally, we present our study setup and how the model is utilised as an identification strategy for analysing treatment effects over time.

Difference-in-differences, proposed by Card and Krueger [40], is a discrete time dynamic causal model. The model is designed to identify and control time-dependent covariates in observational studies, disregarding if the covariates are measured or not [45], and model the average treatment effect. Different from cross-sectional treatment effect estimates, where the treatment effect is aggregated overtime, or time-series treatment effect estimates, where time is treated as a continuous variable. This model can be used to measure treatment effects in discrete time steps. We demonstrate the concept of model

in Fig. 8.5, as shown in the figure, the model replies on the parallel trend assumption, which implies the treatment group and control group are assumed to follow a similar trend for the target variable without a treatment. The factual outcome is the observed target variable after the treatment is applied, the counterfactual outcome, what would have happened if a treatment is never applied, is inferred from the parallel trend assumption. Bayesian difference-in-differences have been applied in studying the influence of policy change on diabetes treatment quality [45]. But there is no documented application of BDID in software engineering to the best of our knowledge.

Theory

By including a group of untreated samples through passive observations from the same time period as the treated samples, all time dependent covariates are implicitly controlled for in a DID model, observed or not. Recall the directed acyclic graph in Fig.8.1, we now extend it to include a τ variable to represent time dependent latent variables (Fig.8.6), to conclude causal effect, both the covariates $\mathbf{X}$ and the time dependent latent variables τ need to be adjusted for. In the difference-in-differences model, the most important assumption is the parallel trend assumption. It implies the counterfactual - what would have happened in the absent of a treatment - is an assumption inherently unobserved. This assumption supports the exchangeability assumption as stated in Eq.8.2, i.e., the treatment assignment is not based on the outcome, rather that the outcome is influenced by the applied treatment. We can express this assumption formally, note that we adopt the same notations from Section 8.3 and 8.4. Additionally, we use $\tau = \{-1, 0, 1\}$ to denote the time periods before, during, and after a treatment is applied.

$$\mathbb{E}[y^0(1) - y^0(-1)|t = 1] = \mathbb{E}[y^0(1) - y^0(-1)|t = 0] \tag{8.13}$$

In Eq. 8.13, $y^0(-1)$ is the target variable with treatment level 0 at time step $\tau = -1$ (pre-treatment status in Fig.8.5), and $y^0(1)$ is the target variable with treatment level 0 at time step $\tau = 1$ without a treatment being applied. We use the superscript to represent the counterfactual status of the treatment group, if a treatment is never applied. The parallel trend assumption states that the target variable measured from the control and treatment group will follow similar trend over time, if no treatment is applied at τ, this is often

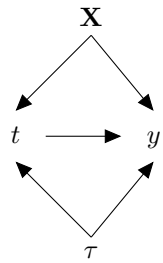

Figure 8.6: A simplified directed acyclic graph showing the relationships of treatment (t), target variable (y), covariates ($\mathbf{X}$), and time dependent latent variables summarised as τ.

referred as the counterfactual outcome. The parallel trend assumption can be check from observational data through means such as data visualisation.

The average treatment effect identified through DID model (ATE_{DID}) is estimated as the following,

$$\begin{aligned} &ATE_{DID} \\ &= (\mathbb{E}[y(1)|t=1] - \mathbb{E}[y(-1)|t=1]) - \\ &\quad (\mathbb{E}[y(1)|t=0] - \mathbb{E}[y(-1)|t=0]) \end{aligned} \tag{8.14}$$

That is, the difference of the target variables in the treated group measured at time step $\tau = -1$ and $\tau = 1$, subtracted with the difference in the control group measured during the same time period, namely, the difference in differences. The target variable y can be estimated as a linear regression from the data observed,

$$y \sim t + \tau + \alpha + \beta\mathbf{X} + \epsilon \tag{8.15}$$

In the regression model, we also assign a dummy variable t indicating if a treatment is applied. The latent variables regression intercept α and coefficient β have a Gaussian distribution as prior, formally,

$$\alpha \sim \mathcal{N}(\alpha|0, \lambda_\alpha) \tag{8.16}$$

Let us consider a total j numbers of covariates $\mathbf{X}$, and regression coefficient is a vector of length j,

Algorithm 4 Bayesian difference-in-differences generative process

Inputs: $\mathbf{X}$ covariates, λ_α prior distribution of α, λ_β prior distribution of β, treatment effect y_n

1: Draw $\alpha \sim \mathcal{N}(\alpha|0, \lambda_\alpha)$
2: Draw $\beta \sim \mathcal{N}(\beta|0, \lambda_\beta)$
3: Draw $\epsilon \sim \mathcal{N}(\epsilon|0, \sigma^2)$
4: **for** each vector of covariate $x \in \mathbf{X}$ **do**
5: Draw $y \sim \mathcal{N}(y|t + \tau + \alpha + \beta^T x, \sigma^2)$

$$\beta_j \sim \mathcal{N}(\beta_j|0, \lambda_{\beta_j}) \tag{8.17}$$

Moreover, since we cannot describe all variations in the data with a linear model, a error term ϵ is included in the model which represent the observation noise. We have,

$$\epsilon \sim \mathcal{N}(\epsilon|0, \sigma^2) \tag{8.18}$$

The joint distribution is factorised as the blow, it is a straight forward application of Bay's theorem. Moreover, we list the generative process for this join distribution in Algorithm 4.

$$\begin{aligned} &p(y, \alpha, \beta|\mathbf{X}, \sigma, \lambda_\alpha, \lambda_\beta) \\ &= p(t) \cdot p(\alpha|\lambda_\alpha) \cdot p(\beta_j|\lambda_{\beta_j}) \cdot \prod_{n=1}^{N} p(y|t, \tau, \alpha, \beta_j, \sigma, \mathbf{X}) \end{aligned} \tag{8.19}$$

Study II: Seasonality effect

A large portion of software in the automotive domain is influenced by the vehicle operating conditions, e.g., precipitation, temperature, humidity, and icing [89]–[91], and most importantly, the mobility needs of people. In the first case, these operating conditions are often seasonal and from the diversity of vehicle markets today, they are difficult to predict beforehand and often requires the software to be evaluated longitudinally to cover a wider range of operating conditions and increase confidence in the conclusion. Additionally, it is naturally reasonable to assume that there are seasonality effects that cannot be observed in an effective manner nor can it be predicted, such as public

events, extreme weathers and so on. The travel demand of users can be largely unpredictable similar to the operating conditions of the vehicles. To adjust for time dependent covariates over a relatively long period of time, requires one to observe a high number of covariates that are frequently unknown when the study is designed. For example, external factors such as cost of fuel, traffic, and parking fare attributes to car owners' preferences on the travel modal [99], and most of which factors are challenging to observe from the perspective of the vehicle. Thus, in a longitudinal software evaluation, there is a need for models that can control covariates even when they are not observed.

Study II is designed to explore the scenario described above, that is, (a) when the performance of the target software treatment is highly dependent on external and seasonal factors such as temperature, (b) and there are potentially latent variables that cannot be observed in an effective manner such as travel demands of individuals. The study is designed to assess the applicability of the BDID model in addressing the challenge and to verify the causal assumption in BDID has real-life relevance. The software deployed in study II is a battery management system for electric vehicles. Battery inside of an electric vehicle have an ideal window of operating temperatures, at the start of a trip, the battery management system will have to either warm up or cool down the battery into said window of operation, this is done at a cost of driving range. Therefore, the ideal preconditioning operation shall take place during charging prior to the trip, utilising the electricity from the grid instead of from the battery. Moreover, if there is a large difference between the ambient temperature and ideal operating temperature, the energy required to precondition the battery is naturally higher. In other words, it is reasonable to expect a dynamic seasonality effect on the final average treatment effect. The treatment, is a software solution that controls and optimises the precondition of battery during chagrining, and it is expected to improve range as the vehicle no longer needs to heat up or cool down the battery during driving operations. Therefore, decrease the energy consumption (measured in Wh/km), and if the software performs as expected, the average treatment effect shall be negative.

Table 8.3: Descriptive statistics of the target variable and covariates as inputs to Bayesian difference-in-differences model, and a description of how the variables are computed. Each variable is aggregated to the vehicle level and min-max scaled.

Variables	Variable description	Group	Mean	Std.
Energy consumption	total electrical energy consumed / total distance	Control	0.277	0.296
		Treated	0.257	0.276
Covariates				
Time period	dummy, 0 for pre-treatment and 1 otherwise	Control		
		Treated		
Treatment	dummy, 0 for control group and 1 for treated group	Control		
		Treated		
Average ambient temperature [$°C$]	average temperature measured at car	Control	0.667	0.318
		Treated	0.954	0.465
Minimum ambient temperature [$°C$]	minimum temperature measure at car	Control	0.630	0.294
		Treated	0.557	0.272
Maximum ambient temperature [$°C$]	maximum temperature measure at car	Control	0.429	0.484
		Treated	0.745	0.626
Average trip distance [km]	total trip distance / total number of trips	Control	0.257	0.192
		Treated	0.314	0.310
Maximum trip distance [km]	longest trip occurred during the observation	Control	0.229	0.322
		Treated	0.270	0.392
Average coolant temperature [$°C$]	coolant temperature measured at battery outlet	Control	0.559	0.333
		Treated	0.821	0.396
Average discharge [Wh]	average battery energy discharge / number of trips	Control	0.612	0.587
		Treated	0.506	0.582
Average initial battery level [Wh]	average battery capacity measured at start of a trip	Control	0.635	0.608
		Treated	0.531	0.616
Mean state-of-charge [%]	average displacement of battery state-of-charge	Control	0.609	0.579
		Treated	0.484	0.562

The study took place between the 1st of August 2021 to the 30th of September 2021, note that during our study, there is a two-week duration that is a typical vacation period in Sweden where the vehicles users reside. We choose

to conduct the study during this period as the weather conditions are dynamic as well as the travel demands, to further demonstrate the power of the BDID model. During the period, we collected data from 24,286 trips and in total of 616,212 kilometres. The control group, similarly to study I, is running the existing version of the software and we do not intervene with the vehicles besides passive data collection. The treatment group, are randomly sampled from a larger fleet of vehicles leased to company employees, and these vehicles received the battery preconditioning software as described above. There are in total 176 vehicles included in the study, similarly to study I, we discard data generated by vehicles with odometer less than 100 kilometres and all trips with average speed higher than 200 kilometres per hour as it exceeds the digital speed limiter implemented in the vehicles. We have in total 9 covariates, and their descriptive statistics are presented in Table. 8.5. All values are presented min-max scaled due to nondisclosure agreement with our industry collaborator.

While the Bayesian difference-in-differences model essentially allows a *post facto* analysis of time dependent treatment effect analysis, it is based on the assumption of parallel trend, as formally defined in Equation 8.13. In practise, this trend states that the control and the treatment group should follow similar trend overtime, if no treatment is applied, implying the difference in between the groups comes from the unobserved time dependent confounding factors. For BDID to identity treatment effect, the assumption needs to be empirically validated through for example visualisation, i.e., by comparing the target variable over time between the treatment and the control group before an intervention is introduced. Another validation method is to fit the BDID model before and after the treatment is applied, to test if the functional form of the counterfactual is correct [100]. Empirically, some observe the samples pre-treatment for as long as possible, to discovery any unknown or underlying trend over time [101]. Furthermore, some matching is required when selecting the control group to compare with the treated group. In practise, this matching process can be done by selecting untreated samples that are as similar as possible to the treated samples, such as vehicle model and engine types, markets, and so on. To ensure the two groups are comparable, we select vehicles with the same vehicle type and have the same battery capacity, and all of the vehicles are registered and driven in Sweden.

Table 8.4: Average energy consumption (Wh/km) for the control and the treatment group at each time step.

	Control	Treated	Difference
τ_{-1}	205.30	190.45	- 14.85
τ_1	218.93	204.36	- 14.57
Change	13.627	13.915	**-0.280**

Results

In this subsection, the results from study II is presented. We show the BDID regression model inferred from a Bayesian network, we illustrate the parallel trend assumption, and the final average treatment effect of the software change.

First we inspect the parallel trend assumption through visualisation and the result is presented in Fig. 8.7. First, we compute the average of the target variable before a treatment is applied, at discrete time step τ_{-1} for both control and treatment groups, and the target variable at the time when the new software is introduced, τ. As can be seen from the figure, the target variable from both groups follow a upward trajectory. Last, we compute the average of the target variable after the treatment is applied at discrete time step τ_1. The visualisation result confirms our assumption that both the control and treatment groups follow similar trend over time, if no treatment is applied, and the target variable observed in the treatment group changes trajectory after the treatment application.

The BDID regression is implemented in Pyro, similarity to the Bayesian logistic regression in study I. We follow the generative process as prescribed in Algorithm 4. The MCMC NUTS sampler is set in Pyro with 3.000 samples and 200 burn-ins with two chains. The Brooks-Gelman-Rubin convergence criteria of $\hat{R} < 1.1$ is met ($\hat{R} = 1.0007$ for α, $\hat{R}_{\beta,mean} = 1.0066$ for all β, and $\hat{R} = 0.999$ for the error term σ). We attach the trace plots in the online appendix.

We assign a weakly informative prior distribution of $\alpha \sim \mathcal{N}(0, 1)$ and $\beta \sim \mathcal{N}(0, 1)$ to the regression intercept and coefficients to introduce scale information to regularise inference, in this case, min-max scalded covariates. We infer the posterior distribution $p(y, \alpha, \beta|\mathbf{X}, t, \tau, \sigma, \lambda_\alpha, \lambda_\beta)$, in Figure.8.12 (attached at the end of the chapter), we present the posterior distribution of

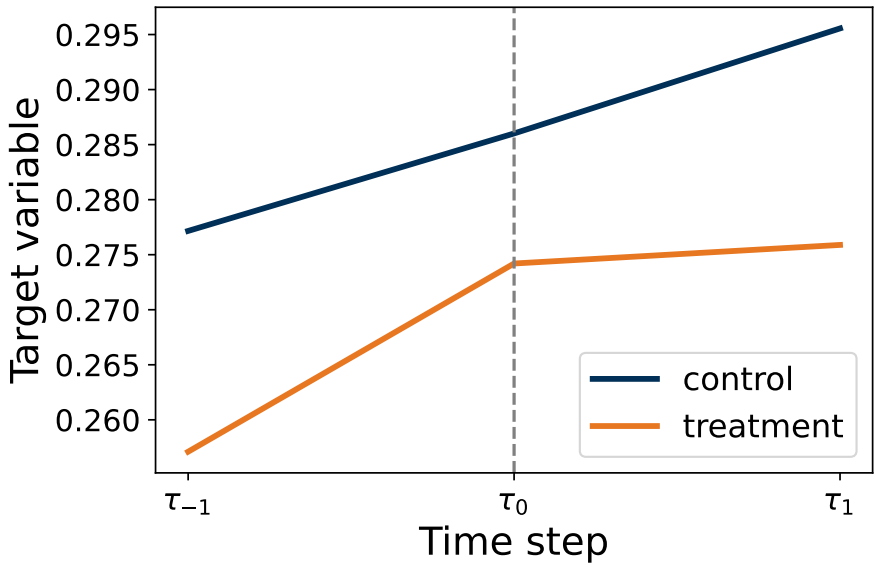

Figure 8.7: Target variable y measured before the treatment (τ_{-1}), when the treatment is applied (τ), and after the treatment (τ_1), for the control and the treatment groups, y is min-max scaled.

the regression intercept (α) and the regression coefficients β. As can be seen, the posterior distribution of the two dummies variables indicating the time period and if a treatment is applied (β_1 and β_2) are informative of the treatment outcome as expected. While the posterior distributions for covariates describing the high voltage battery activity level such as total energy discharge and state-of-charge change (β_9 to β_{10}), contribute positively to average energy consumption, however, the uncertainty of the effect is high.

Last, we include a difference in differences average treatment effect analysis following Equation.8.14. First, we compute the difference of expected target variable $\mathbb{E}[y]$ at time step τ_{-1} between the control ($t = 0$) and the treatment ($t = 1$) groups, this value can be interpreted as the preexisting differences between the groups as a result of unobserved confounding effects. Second, we calculate the difference of $\mathbb{E}[y]$ between the control and treatment group at time step τ_1, this difference is a sum of the preexisting differences and the treatment effect if the parallel assumption holds. The results are presented in

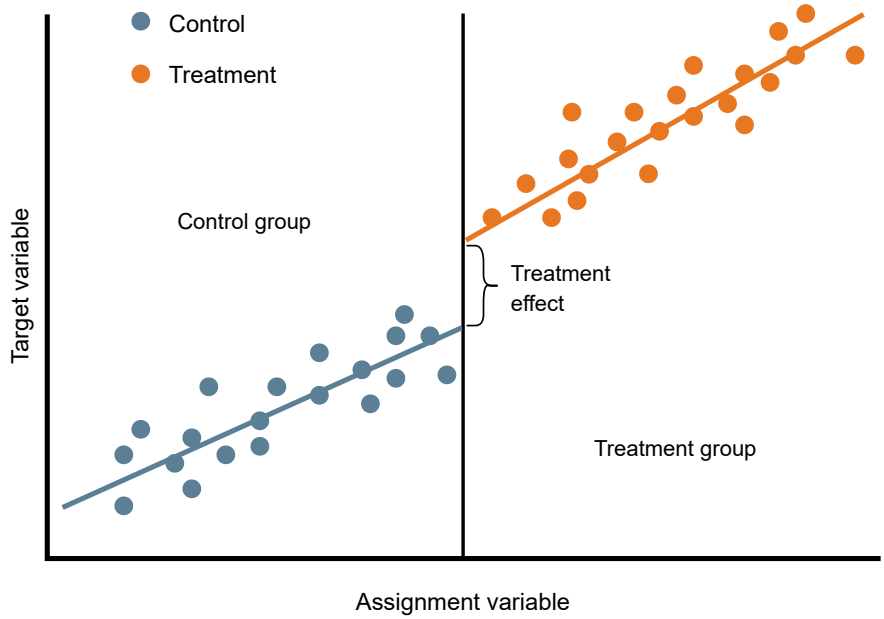

Figure 8.8: An illustration explaining the Regression Discontinuity Design model, and how the average treatment effect is estimated.

Table.8.4.

8.6 Bayesian regression discontinuity

In this section, we present the Bayesian regression discontinuity design (BRDD) theory and observational study III that is designed to demonstrate the use case of BRDD for evaluating automotive software. The theory and assumptions of the model is presented formally along with the algorithm for Bayesian inference. We present the study III, the setup, data collection method, and how BRDD is used as a strategy for identifying continuous covariate dependent treatment assignment.

Regression discontinuity design, proposed by [41], is a causal modelling approach aiming to determine the treatment effect when the treatment assignment is confounded by one continuous covariate by assigning a cut-off point.

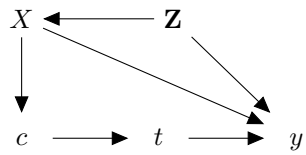

Figure 8.9: A simplified directed acyclic graph showing the relationships of treatment (t), target variable (y), assignment variable (X), the cut-off point (c), and other confounding factors ($\mathbf{Z}$).

This continuous covariate is referred as the assignment variable, in Fig.8.8, we illustrate the principle of RDD. Observations of the target variable is made around the threshold, in this case, average treatment effect can be inferred without the need of a randomised experiment. In practise, visualising of the assignment variable and the target variable is a simple yet powerful tool to inspect their relationship [102]. While RDD allows inference of treatment effect with the absent of randomisation, the model alone does not explicitly or implicitly conclude causality as it does not identify other unobserved confounding effects. Moreover, RDD design essentially infer treatment effect with single covariate $X = x$, in that perceptive, the model has a limited degree of external validity. However, the RDD model is similar to a randomised experiment with bias below 0.01 standard deviations on average especially analysed with the Bayesian approach [92], indicating a high internal validity. In our study III, we investigate the scenario when a software is only useful in reducing the fuel consumption of the vehicle, if the average trip distance of the given vehicle is over a certain threshold, without *ex-ante* randomisation.

Theory

In [41], RDD is discussed in the context of regression, and in this subsection, we will describe it in the potential outcome framework provided the conditional exchangeability assumption holds as formulated in Equation. 8.3. We illustrate the relationship between variables in a RDD in 8.9. As can be seen, the outcome y is influenced by the assignment variable as well as the predetermined cutoff point c. As mentioned in paragraph before, a RDD does not automatically eliminate other confounding factors in the system, we illustrate that with $\mathbf{Z}$ in the directed acyclic graph. The fundamental concept of RDD is that the treatment assignment is deterministic by a covariate X (we call

this the assignment variable) with a fixed threshold, the assignment variable X is assumed to be correlated to the target variable y, and their correlation is smooth. Under this assumption, any discontinuity of the target variable y as a function of X is interpreted to be the causal treatment effect around the predetermined threshold. Let $X = c$ be the predetermined cut-off point of the assignment variable with c being an arbitrarily determined value, formally,

$$\mathbb{E}[y|X = x, t = 1], \text{ and } \mathbb{E}[y|X = x, t = 0] \tag{8.20}$$

are continuous in x.

This assumption also implies that the probability distribution of y given the covariate $X = x$ is smooth. This assumption is stronger than needed, as the continuity without a treatment effect is only expected around the cut-off point $X = c$ and the assumption above covers such a scenario. Different from a matching problem, the requirement for overlap requires control and treated samples to have all possible combinations of the covariates, in a DID model, for all values of x, the propensity of treatment assignment is either 0 or 1, i.e., on either side of the cut-off point c. We call this a sharp design, as opposed to fuzzy design. In practise, a fuzzy design might be more attractive, as it is reasonable to include samples close to either side of the cut-off point.

Without the need of extrapolating due to the lack of overlap, at the cut-off point $X = c$, we can infer the average treatment effect from regression discontinuity design (ATE_{RDD}) as,

$$ATE_{RDD} = \mathbb{E}[y|X = c, t = 1] - \mathbb{E}[y|X = c, t = 0] \tag{8.21}$$

That is, the difference between average observed target variable y with or without the treatment $t = \{0, 1\}$, at a given cut-off point $X = c$. This average treatment effect can be estimated as we have made the smoothness assumption in Equation. 8.20. We would like to empathise that although we demonstrate a linear regression for the prediction of target variable in a BRDD, a polynomial regression can be applied to handle more complex relations between the assignment variable and the target variable. When the smoothness assumption holds, the target variable y can be predicted using a simple linear regression for the sharp design at $X = c$, stated as following centred around the cut-off point,

Algorithm 5 Bayesian regression discontinuity design generative process

Inputs: **X** covariates, λ_α prior distribution of α, λ_β prior distribution of β, treatment effect y_n

1: Draw $\alpha \sim \mathcal{N}(\alpha|0, \lambda_\alpha)$
2: **for** each $\beta \in \beta_j$ **do**
3: Draw $\beta \sim \mathcal{N}(\beta|0, \lambda_\beta)$
4: Draw $\epsilon \sim \mathcal{N}(\epsilon|0, \sigma^2)$
5: Draw $y \sim \mathcal{N}(y|\alpha + \beta_1(x-c) + \beta_2 t + \beta_3(x-c)t + \beta_4 Z, \sigma^2)$

$$y \sim \alpha + \beta_1(x-c) + \beta_2 t + \beta_3(x-c)t + \beta_4 Z + \epsilon \tag{8.22}$$

where, the α is the regression intercept, β_j are the regression coefficients. They are both latent variable inferred from a Bayesian network. t is a dummy variable indicating if a treatment has been applied, and ϵ represent the linear noise in the model. We assign a Gaussian distribution as a prior to the regression intercept and coefficients, namely,

$$\alpha \sim \mathcal{N}(\alpha|0, \lambda_\alpha) \tag{8.23}$$

and,

$$\beta_j \sim \mathcal{N}(\beta_j|0, \lambda_{\beta_j}) \tag{8.24}$$

An error term ϵ is included in the model which represent the observation noise as a linear model has its limitations for describing the noisy reality. We have,

$$\epsilon \sim \mathcal{N}(\epsilon|0, \sigma^2) \tag{8.25}$$

The joint distribution is factorised as the blow applying Baye's law. We describe the generative process for this join distribution in Algorithm 5.

$$\begin{aligned} &p(y, \alpha, \beta_j|Z, t, x, c, \sigma, \lambda_\alpha, \lambda_{\beta_j}) \\ &= p(t) \cdot p(\alpha|\lambda_\alpha) \cdot p(\beta_j|\lambda_{\beta_j}) \cdot \prod_{n=1}^{N} p(y|Z, t, x, c, \alpha, \beta_j, \sigma) \end{aligned} \tag{8.26}$$

Study III: Covariate dependent treatment assignment

In absent of randomisation, assuming automotive software functions to be independent from their operation environment or usage by the customers is a naïve approach for estimating the treatment effect of software changes. To that end, the performance or even the activation of certain automotive function is dictated by the usage, in other words, we frequently run into the situation in which a covariate that determines treatment assignment. To understand the performance of this type of software is important for the following two reasons, first, it brings insights on the usefulness of a given software feature, validating assumptions made during development against how the product is actually utilised. Second, it allows the development organisations to evaluate the software effectiveness in conditions that are most determining of the effect. In study III, we present a case that illustrate the importance of causal inference when the treatment effect is strongly dependent on a covariate.

Plug-in hybrid vehicle, is a type of electrified vehicle with two sets of propulsion, combustion engine and electric motors. This type of vehicles usually have limited pure electrical range, and to maximised the benefit of eclectic drive such as high efficiency during city driving, zero direct emission; automotive manufacturers typically have a number of control software solutions to optimise the distribution of the electrical and chemical energy on a given trip. A simple version of the optimisation strategy, is to prioritise the electrical energy whenever available and deplete the battery first before using the combustion engine. This type of strategy usually works well when the driver is expected to travel less distance than the electrical range, and not on highways where the combustion engine works more efficiently than the electrical motor. Alternatively, the optimisation can be done through a prediction of trip distance and destination – if the trip is predicted to be farther than the electrical range, the car will not prioritise the use of electrical energy and deplete the battery early in the trip, with the rationality that the drivers is predicted to enter the city later where direct emission from the combustion engine is undesirable. Thus, the assignment of this software is determined by the trip distance by design. The performance of this type of software function is highly dependent on how the cars are driven, more specifically, on the trip distance. Thus, to evaluate such a software feature, we chose a cutoff point of the assignment variable trip distance, at around the designed electrical range of the vehicle where the software is expected to have the most impact, and apply the BRDD

model for treatment effect inference and modelling.

Study III is conducted in Sweden, on a fleet of 50 plug-in hybrid vehicles. The study took place between the 19th of October 2020 to the 28th of Febuary 2021, and during which, 12,231 numbers of trips are observed and the vehicles have driven a total of 191,552 kilometres. The software is designed so that if the predicted trip distance is less than the electrical range, the car will prioritise and use the electrical energy first. If not, the vehicle will optimise the energy usage between the electrical motor and the combustion engine according to the predicted trip distance and destination. We apply the same data post-processing logic as study I and II, namely, data collected from brand new vehicles and trips with higher than possible average speed are excluded from the model. In study III, the target variable is the average fuel consumption, and the assignment variable is the total trip distance.

There are two important assumptions in a regression discontinuity design. First, we assume unobserved covariates do not effect the treatment effect or assignment. This is a strong assumption, similarly to what is previously discussed for BPSM and most causal inference problems, to satisfy this assumption, it requires prior knowledge to how the software interact with the users and inputs from the domain experts. The second assumption of BRDD is the expectation of the target variable $\mathbb{E}[y]$ is continuous with respect to the assignment variable X. Mathematically speaking, function $f(x)$ continuity at $x = c$ can be determined if $\lim_{x \to c} f(x)$ exist, and $\lim_{x \to c} f(x) = f(c)$. A continuity check should not be performed on observational data which is per definition discontinuous, instead, the continuity assumption can be check with a density test, as suggested by [103]. Last but not least, the choice of cut-off point $X = c$, requires that the assignment mechanism to be known to the development teams. In practise, the cut-off point might not be a sharp differentiation but rather a bandwidth, which can be determined either through the design intend of the software or through observational data collected prior to a treatment is introduced.

Results

In this section, we present the results from BRDD. As discussed in the previous subsection, there are other covariates that could potentially confound the treatment effect, in $\mathbf{Z}$. In this analysis, to adjust for the covariates, we condition on one covariant that is the total displaced state-of-charge, $Z \in \mathbf{Z}$.

This covariate indicates how much battery is used during a given trip, naturally, if a trip distance is fixed, the more battery is used, the less the fuel consumption there is. Thus, to ensure the trips are comparable disregard the software treatment, we only look at $\mathbb{E}[y|Z > 90]$, they are trips during which the battery has been depleted. The cut-off point is arbitrarily determined at $c = 60$, which is the approximated pure electrical range of the vehicle model we are observing. We do not min-max scale the value in Fig.8.10, due to that the cut-off point of the assignment variable represents a physical measurement and we choose to articulate the physical meaning through presenting the unscaled value. As a min-max scaled value is difficult to interpret, to reflect the physical meaning of the measurement while maintaining our confidentiality agreement, we remove the units of the measured target variable instead. Two linear regression is fitted on either side of the cut-off point, to illustrate a discontinuity of the regression line at cut-off as a representation of the software treatment effect, as can be seen in Fig.8.10, at $c = 60$.

Following Algorithm 5, we implement the BRDD regression model in Pyro. The MCMC NUTS solver is set with 2 chains of 2,000 samples each and we discard the first 200 steps as warm-up steps. The model, in wide format, has the following four input features, $x - c$ (so that the regression is fitted centred around the cut-off point), with x being the assignment variable of trip distance and $c = 60$, $t = \{0, 1\}$ (to effectively control the regression model), $(x - c)t$ ($(x - c)t = 0$ for the controlled group, and 1 otherwise), and the change of the state-of-charge of the battery named Z. The convergence criteria $\hat{R} < 1.1$ is met at $\hat{R} = 1.0004$ for α, $\hat{R}_{\beta,mean} = 1.0017$, $\hat{R} = 1.0000$ for the error term σ. The trace plots are attached in the online appendix. For the prior distributions, we have a weakly informative prior of $\alpha \sim \mathcal{N}(0, 1)$, and we select a more informative prior for $\beta \sim \mathcal{N}(0, 0.5)$. We choose a weakly regularising prior for the standard deviation $\sigma \sim$ Half-Cauchy$(0, 5)$, as it approximate uniform distribution and it is weakly informative near 0. We plan to start with a non-informative prior for the error term and adjust if the solver does not converge. The solver meets the convergence criteria with the priors mentioned above. The posterior distribution from this Bayesian network, $p(y, \alpha, \beta|Z, t, x, c, \lambda_\alpha, \lambda_\beta, \sigma)$, is inferred. The posterior distribution of the regression intercept, α, returned a Gaussian distribution centred around $\alpha_{mean} = 0.626$, with a standard deviation of $\alpha_{std} = 0.0056$. Similarly, the posterior distribution of the error term ϵ is a Gaussian distribution centred

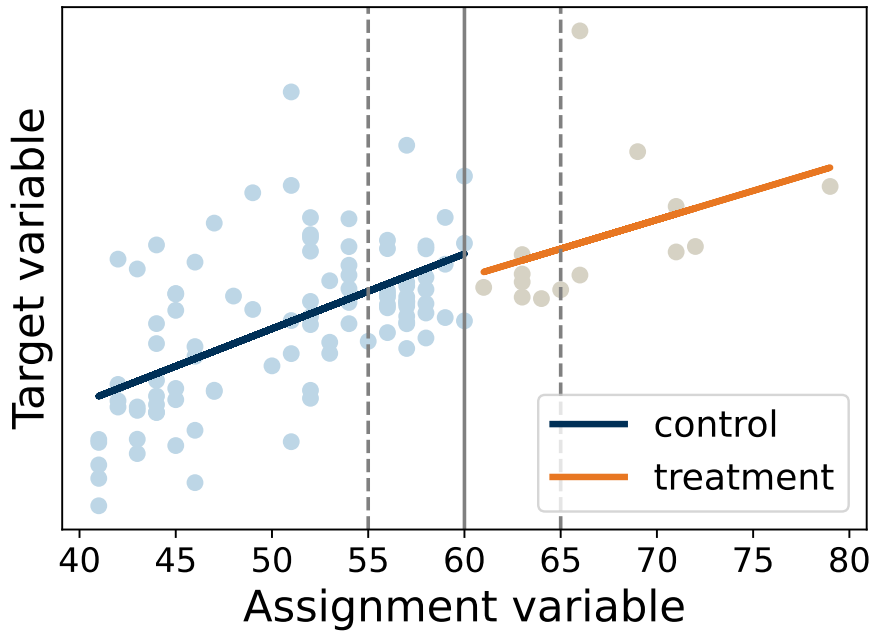

Figure 8.10: Target variable measured before and after the cut-off point ($c = 60$), with respect to the assignment variable.

around $\sigma = 0.213$ with standard deviation of 0.0010. We present the posterior distributions in Fig.8.11.

The outcome for BRDD, is two linear regression lines at either side of the cut-off point. Using the posterior distribution, the regression expression (y_c) to the left hand side of the cut-off point can be expressed as,

$$y_c \sim \alpha + \beta_1 x + \beta_4 Z + \epsilon \tag{8.27}$$

and the regression expression for the treated samples (y_t) to the right hand side of the cut-off point can be expressed as,

$$y_t \sim (\alpha + \beta_2) + (\beta_1 + \beta_3)x + \beta_4 Z + \epsilon \tag{8.28}$$

The average treatment effect ATE_{RDD} is inferred following Equation.8.21,

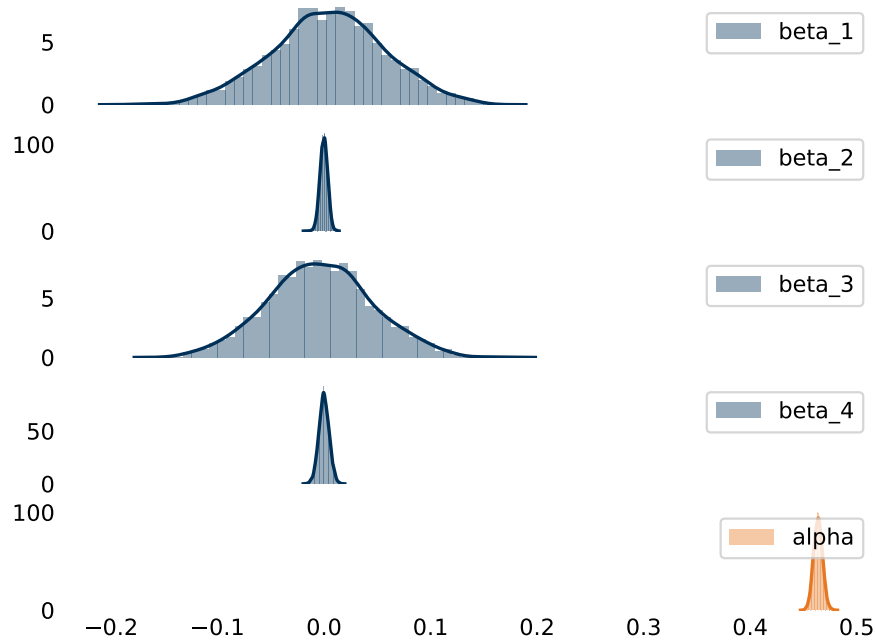

Figure 8.11: Posterior distribution of the regression coefficients β, and the regression intercept α.

we find that, at the cut-off point ($X = c$), the expected target variable treated and untreated differ by $ATE_{RDD} = y_t(x = c) - y_c(x = c) = -1.1954$. This is an unbias estimation of the local conditional treatment effect. As can be seen in Fig.8.10, the discontinuity at the cut-off point can be interpreted as the treatment effect.

8.7 Discussion

In this section, we provide a discussion on the advantages and limitations of the BOAT framework from the perspective of causal inference and piratical applications in the automotive domain.

Causal assumptions and domain knowledge

Causality cannot be inferred from observational data alone [15], behind every causal conclusion, there are some complementary causal assumptions that might not be testable empirically. Before adjusting for confounding biases, some judgements must be made based on domain knowledge as also discussed by [15], [51], [52], [66].

Take propensity score matching as an example, different from a randomisation process, in which all covariates will be balanced in the control and the treatment groups observed or not, causal inference with covariate adjustment requires the covariates to be observed. Performing propensity score matching in combination with observational study, requires a set of carefully chosen covariates which rely heavily on domain knowledge. To that end, similar requirements on domain knowledge is also experienced with other models in the BOAT framework. In order to select samples in the control group for observation that are as similar as the treated samples as possible, a matching process is recommended. The judgement on "as similar as possible" is a judgement that cannot be made without existing data on the cohorts and domain knowledge. Likewise, domain knowledge is required for selecting the cut-off point for the assignment variable in a regression discontinuity design.

The requirement on domain knowledge implies that causal inference from observational studies need manual input when applied in software engineering practises. While randomised online experiments can be automated to a large extend, as demonstrated in the SaaS domain [16], [21], [35], causal inference with observational data is a process that would potentially require more manual efforts. The extra overhead of efforts from developers could potentially pose a challenge in implementing causal inference in combination with observational studies in automotive software engineering. Thus, the application of BOAT framework on a larger scale cannot be done without some form of data-driven causal discovery methods.

Extension to BOAT

As shortly discussed in Section.8.3, the BOAT framework does not cover all potential scenarios in causal inference of observational studies in the automotive domain. Since there are a few more techniques that can be applied when inferring causal treatment effect without randomised experiments. In this sub-

section, we offer a discussion on our decision to why some of the models are not included in the BOAT framework as if now. These models, useful in many domains as literature reports, we have yet to find their feasible applications in the automotive sector.

First, the positive decision to whether there is a need to infer the latent variable is intentionally left out from the flow chart, when a latent variable needs to be inferred, methods such as instrumental variable can be applied [95]. Instrumental variable method uses a latent variable to explain the correlation to the error term. The method should account for unexpected behaviour between variables, however, the explanation it provides cannot be interpreted with a physical meaning. In many automotive software where interpretability is considered crucial, especially for development organisations to take design decisions. Moreover, instrumental variable method has the tendency to produce bias results when the sample is small, which is a known limitation in the automotive domain.

Second, a popular school of causal inference method, structural causal model and do-calculus [15] offers a comprehensive approach to causal inference provided the causal structure is known. This causal structure is represented in a DAG, such as the trivial example in Fig.8.1. Each component in a DAG has their graphical and numerical representations, then through the language of do-calculus, for example $p(Y|do(T), X)$, we can represent intervention and infer treatment effect from a DAG. In order for a structural causal model to be effective, the structure of the model, i.e., the DAG, needs to be learnt either through domain-knowledge or though a data-driven causal discovery process. The formal is time consuming and potentially subjected to biases of individuals, the latter requires large amount of data yet does not address limitations such as sampling bias, measurement error, and confounding effects [94].

Finally, there are other methods addressing preexisting differences between the control and the treatment groups, such as inverse propensity weighting. While achieving similar objective as propensity score matching (adjusting for confounding factors), inverse propensity weighting is a parametric method and it is known to be creating imbalance groups when the sample size is insufficient.

8.8 Conclusion

In this paper, we introduce the BOAT framework for software engineering in the automotive domain, enabling online evaluation of the software in a causal fashion when a fully randomised experimentation is impossible, undesired, or unethical. Applying the Bayesian causal inference models, we demonstrate how a causal conclusion can be drawn in absent of randomisation utilising the high flexibility of Bayesian inference towards sample size, as demonstrated in other areas of science [45]–[47], [85]. Combining theory with practise, we include three illustrative cases from the automotive domain for further enforce the need of causal inference in software engineering. The three cases are conducted together with our industry collaborator, we introduce three software to a fleet of vehicles driven by real-world customers. We relate the causal assumptions to scenarios experienced in practise, aiming to provide a guideline on when and how to better apply the causal modelling for inferring the software effects from different software evaluation needs.

We provide a decision making flowchart along with the three causal inference models included in the BOAT framework. The flowchart is design with the objective of guiding development organisations on which causal inference models should be used to address their corresponding challenges in real life. In the BOAT framework, we include three models, they are, (1) Bayesian propensity score matching for generating balanced control and treatment groups without randomisation, (2) Bayesian difference-in-differences for controlling unobserved seasonal factors over time, (3) Bayesian regression discontinuity design for analysing the treatment effect when treatment assignment is determined by a continuous covariate. All of the three models and their assumptions have their implications in practise, as experienced from the automotive domain when attempting to evaluate software without randomisation, in this work, we provide a formal discussion of the inference models, as well as their corresponding real world implications and applications. The three cases are designed together with software engineering teams in our case company, to simulate challenges experienced when evaluating software online without randomisation. With the development teams, we introduce new software treatment to a fleet of vehicles, conduct data collection, and use the empirical data as inputs to the BOAT framework, additionally, we assess the causal assumption in relation to the empirical cases. We find the causal models in the BOAT framework to be highly applicable in automotive software

engineering, and they enable the development organisations to evaluate software changes in an online and causal manner. Furthermore, we find there is a strong dependency on domain knowledge when designing an online observational study as the cause-and-effect is not always known, and when validating some of the causal assumptions empirically.

In our future work, we aim to incorporate causal discovery process when designing an online experiment or observational study, since we cannot always assume the cause-and-effect of a system is known [34]. Moreover, we plan to explore data-driven causal discovery methods to inform and potentially automate the design of experiments, with the objective of increasing the efficiency and the effectiveness of online experimentation in automotive software engineering.

Online Appendix

The online appendix can be found as a Jupyter Notebook via the following link: github.com/yuchueliu/BOAT.

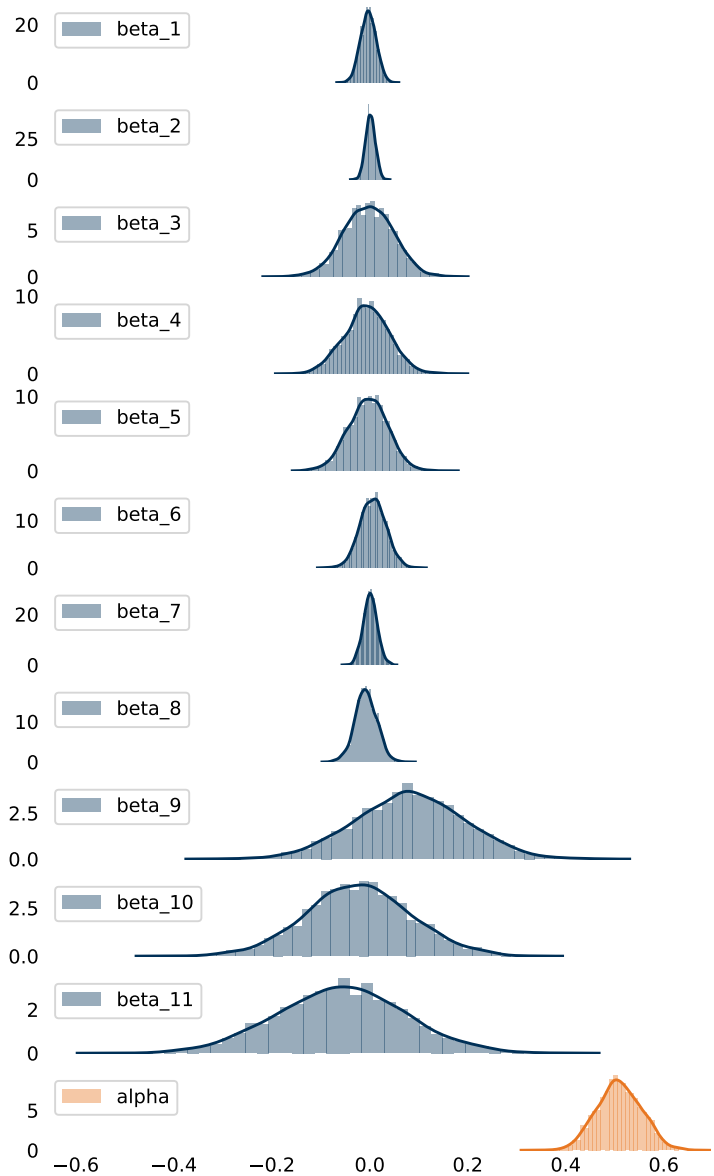

Figure 8.12: Posterior distribution of the regression coefficients β, β is ordered as Table. 8.5, and the regression intercept α.

CHAPTER 9

Concluding remarks and future work

In this chapter, I present the concluding remarks and the outlook of the future research in this doctorate research. This thesis presents a framework of how to enable online experimentation in the automotive domain, provided the challenges and limitations experienced in the sector. The thesis consists of four publications, each addressing one aspect of the obstacles posed in the adoption of online software experimentation. In the end, a future outlook of this doctorate research is presented.

9.1 Conclusion

In this thesis, an exploratory research is done to discover and analyse the state-of-practise and limiting factors for online experimentation adoption. After identifying the inherent limitation of the sample size in the domain, an experimental design method is empirically evaluated. Moreover, an alternative approach to fully randomised experimentation is proposed, aiming to address limitations resulted from software non-functional requirements, such as safety and privacy. An overview of the background and theory to online experimentation in the potential outcome framework is introduced. The research

objective and methods are presented, along with a discussion on threats to validity.

RQ1: How to run large-scale online experiments in the automotive domain in a fast and reliable fashion?

This research question RQ1 is addressed through studying the existing online experimentation architecture and exploring the limitations in the automotive domain. With a combination of case study and literature review, we find that most, if not all, existing architectures are not applicable in the automotive domain as they do not explicitly address the limitations experienced in the domain. As a result, we propose a hybrid architecture, which combines edge and cloud infrastructure to enable fast and reliable software parameterised changes and remote data collection. This architecture is then compared with the current state-of-practise through a case study, and we find many components are common to what has already been adopted in two other automotive companies. We introduced the software architecture to a fleet of 50 vehicles, to test the robustness of the design, and to validate its usability. This architecture is designed to specifically address limitations that are manifested from the hardware dependency and non-functional requirements of the embedded software.

RQ2: How can the inherent limitation of sample sizes in the automotive domain be addressed?

After identifying a potentially inoperable challenge of online experimentation in automotive, we formulate RQ2. This research question focuses on the inherent limitation of sample size. To answer this question partially, we conduct a study that utilises causal modelling for experiment design, presented in Chapter. 6. In this research, we apply and evaluate an experiment design method that allows balanced control and treatment groups to be generated provided pre-experimental data. With our industry collaborator, we evaluate the approach of designing experiment qualitatively through an *in situ* case study running for six month, and a online experiment designed with the method on 28 vehicles. The design approach is highly adoptable in software engineering, and the causal model is proven to reduce both variance and biases in the modelling of the treatment effects comparing to a fully randomised experiment at

the same sample size. While the method has been applied in other areas of science, this is the first paper to evaluate it in the context of software engineering. This study shows that a data informed experiment design can in fact reduce the need of large samples for balancing covariates between the control and treatment group.

Additionally, when sample size is small due to a limited access to available vehicles, the treatment assignment becomes confounded on covariates such as a scenario we demonstrate in Chapter. 7. In this case, the randomised experiment is no longer an option, the software evaluation becomes an observational study, and the treatment effect needs to be inferred through causal modelling.

RQ3: Can causality be concluded in the absence of randomisation in a software online evaluation?

In the absence of randomisation due to limited sample size, undesired due to safety requirements, or unethical due to the lack of user consent, the causal assumption of exchangeability is violated and causality cannot be concluded without further assumptions. We conduct studies from three separate cases in an automotive company, on a fleet of 1200 cars, and three different software features to empirically evaluate Bayesian causal inference models. We combine Bayesian causal inference models with observational testing, proposing an alternative approach to fully randomised experimentation in the absence of randomisation. Furthermore, we relate the built-in assumption in the Bayesian causal models to specific challenges experienced in practise, that further enforces our proposal of observational testing powered by Bayesian causal inference. We address issues related to non-randomised samples due to non-functional requirements, seasonality and externally dependent treatment assignment due to the dynamic operating conditions of the vehicles.

9.2 Future work

Recall the framework, Fig. 1.1, presented in Chapter 1, note that it is now extended with future outlook marked in light blue within the same framework, presented in Fig. 9.1. In the continuation of this doctorate research, the main research efforts will remain enabling online experimentation in the automotive sector with a stronger focus in causal modelling.

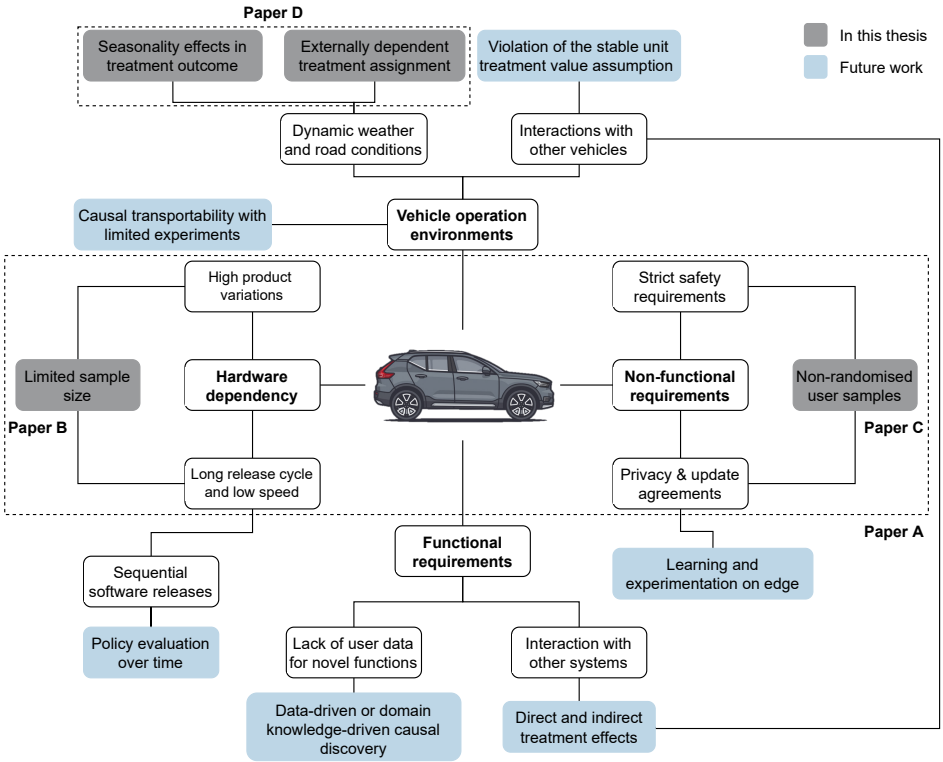

Figure 9.1: Challenges of online experimentation adoption in automotive, proposed approaches and solutions, and future outlook of this doctorate research.

Similarly to most research efforts in causal inference [34], we assume the input and output variables are known in this thesis. However, in practise, the causal structure might be vastly unknown prior to the experiments. This means that we cannot confidently design and conduct and experiment, and be certain of its benefits for the development organisations. Therefore, a causal discovery process might be required prior to design of an experiment especially in complex systems such as autonomous drive or active safety related functions. We intend to explore data- and knowledge-driven causal discovery methods, and run empirical studies to analyse their applicability in automotive software engineering. Causal graphical models are a direct output of

causal discovery; utilising the graphical models, experiments can be designed utilising the theory of D-separation [15], [38]. Moreover, we can analyse direct as well as indirect treatment effects from the same experiment. A causal graphical model will enable us to study the effect of software change in a more comprehensive and systematic fashion, and it will allow us to study the interactions between different systems [104]. However, interactions with other samples such as other vehicles, could result a direct violation in the stable unit treatment value assumption that is critical to an experiment, in which case, a network effect needs to be understood [105].

As a result of software release practises in the automotive sector, a model for sequential software evaluation is of strong interest in the continuation of this research. This type of modelling calls for more advanced machine learning models as the changes are also dependent on temporal factors. We consider modelling such a sequential software change as a policy evaluation problem, we aim to present an approach that is the most suited for supporting decision making over time [106].

From the limitations that is privacy and user consent for software changes, we see a value in building a learning system and experimentation on edge devices. Such a model also brings value in terms of personalisation - online experimentation can inform you of the average treatment effect, but lack information in individual treatment effects [107]. For that reason, a software that is optimised for everyone does not necessarily guarantee satisfaction of individual customers. Therefore, models that empower experimentation and learning on individual vehicle or even user level is of research interest for privacy preservation and product optimisation.

Last but not least, the empirical application and validation of the theory of causal transportability [108], [109] with limited further experiment is highly applicable in the automotive domain, especially in software features that are considered impossible to validate due to the large amount of experiments expected. Imagine an ideal scenario, in which we could validate a piece of autonomous drive feature, such as object detection, in a randomised and small-scale experiment, and the causal learning can be formally transferred to a wider context with confidence without the need of further experimentation.

References

[1] J. Bosch and H. H. Olsson, "Toward evidence-based organizations: Lessons from embedded systems, online games, and the internet of things," *IEEE Software*, vol. 34, no. 5, pp. 60–66, 2017.

[2] D. I. Mattos, J. Bosch, and H. H. Olsson, "Challenges and strategies for undertaking continuous experimentation to embedded systems: Industry and research perspectives," in *Lecture Notes in Business Information Processing*, Springer International Publishing, 2018, pp. 277–292.

[3] D. I. Mattos, J. Bosch, H. H. Olsson, A. M. Korshani, and J. Lantz, "Automotive A/B testing: Challenges and lessons learned from practice," IEEE, Aug. 2020.

[4] K. Forsberg and H. Mooz, "The relationship of systems engineering to the project cycle," *Engineering Management Journal*, vol. 4, no. 3, pp. 36–43, Sep. 1992.

[5] F. Falcini, G. Lami, and A. M. Costanza, "Deep learning in automotive software," *IEEE Software*, vol. 34, no. 3, pp. 56–63, May 2017.

[6] J. Bosch, "Speed, data, and ecosystems: The future of software engineering," *IEEE Software*, vol. 33, no. 1, pp. 82–88, Jan. 2016.

[7] A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch, "The evolution of continuous experimentation in software product development: From data to a data-driven organization at scale," in *2017 IEEE/ACM 39th*

International Conference on Software Engineering (ICSE), IEEE, May 2017.

[8] A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch, "Effective online controlled experiment analysis at large scale," in *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, IEEE, Aug. 2018.

[9] H. H. Olsson and J. Bosch, "Climbing the "stairway to heaven": Evolving from agile development to continuous deployment of software," in *Continuous Software Engineering*, Springer International Publishing, 2014, pp. 15–27.

[10] U. Eliasson, R. Heldal, J. Lantz, and C. Berger, "Agile model-driven engineering in mechatronic systems - an industrial case study," in *Lecture Notes in Computer Science*, Springer International Publishing, 2014, pp. 433–449.

[11] L. Gren and P. Lenberg, "Agility is responsiveness to change," in *Proceedings of the Evaluation and Assessment in Software Engineering*, ACM, Apr. 2020.

[12] F. Giaimo, H. Andrade, and C. Berger, "The automotive take on continuous experimentation: A multiple case study," in *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, IEEE, Aug. 2019.

[13] H. H. Olsson and J. Bosch, "From opinions to data-driven software r&d: A multi-case study on how to close the 'open loop' problem," in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, IEEE, Aug. 2014.

[14] A. Fabijan, H. H. Olsson, and J. Bosch, "The lack of sharing of customer data in large software organizations: Challenges and implications," in *Agile Processes, in Software Engineering, and Extreme Programming*, Springer International Publishing, 2016, pp. 39–52.

[15] J. Pearl, "Causal inference in statistics: An overview," *Statistics Surveys*, vol. 3, no. none, Jan. 2009.

[16] A. Deng, Y. Xu, R. Kohavi, and T. Walker, "Improving the sensitivity of online controlled experiments by utilizing pre-experiment data," in *Proceedings of the sixth ACM international conference on Web search and data mining - WSDM '13*, ACM Press, 2013.

[17] Y. Xu and N. Chen, "Evaluating mobile apps with A/B and quasi A/B tests," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, psm: in linkedin, ACM, Aug. 2016.

[18] A. Fabijan, P. Dmitriev, C. McFarland, L. Vermeer, H. H. Olsson, and J. Bosch, "Experimentation growth: Evolving trustworthy A/B testing capabilities in online software companies," *Journal of Software: Evolution and Process*, vol. 30, no. 12, e2113, Nov. 2018.

[19] R. Kohavi and R. Longbotham, "Online experiments: Lessons learned," *Computer*, vol. 40, no. 9, pp. 103–105, Sep. 2007.

[20] P. R. Rosenbaum and D. B. Rubin, "The central role of the propensity score in observational studies for causal effects," *Biometrika*, vol. 70, no. 1, pp. 41–55, 1983.

[21] H. Xie and J. Aurisset, "Improving the sensitivity of online controlled experiments," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, Aug. 2016.

[22] F. Giaimo, H. Andrade, and C. Berger, "Continuous experimentation and the cyber–physical systems challenge: An overview of the literature and the industrial perspective," *Journal of Systems and Software*, vol. 170, p. 110 781, Dec. 2020.

[23] U. Eklund and J. Bosch, "Architecture for large-scale innovation experiment systems," in *2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture*, IEEE, Aug. 2012.

[24] F. Giaimo and C. Berger, "Design criteria to architect continuous experimentation for self-driving vehicles," in *2017 IEEE International Conference on Software Architecture (ICSA)*, IEEE, Apr. 2017.

[25] D. I. Mattos, A. Dakkak, J. Bosch, and H. H. Olsson, "The HURRIER process for experimentation in business-to-business mission-critical systems," *Journal of Software: Evolution and Process*, Oct. 2021.

[26] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne, "Controlled experiments on the web: Survey and practical guide," *Data mining and knowledge discovery*, vol. 18, no. 1, pp. 140–181, 2009.

[27] R. Kohavi, A. Deng, B. Frasca, R. Longbotham, T. Walker, and Y. Xu, "Trustworthy online controlled experiments," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '12*, ACM Press, 2012.

[28] P. L. Li, P. Dmitriev, H. M. Hu, X. Chai, Z. Dimov, B. Paddock, Y. Li, A. Kirshenbaum, I. Niculescu, and T. Thoresen, "Experimentation in the operating system: The windows experimentation platform," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, IEEE, May 2019.

[29] F. Fagerholm, A. S. Guinea, H. Mäenpää, and J. Münch, "The RIGHT model for continuous experimentation," *Journal of Systems and Software*, vol. 123, pp. 292–305, Jan. 2017.

[30] S. Satyal, I. Weber, H.-y. Paik, C. D. Ciccio, and J. Mendling, "AB testing for process versions with contextual multi-armed bandit algorithms," in *Advanced Information Systems Engineering*, Springer International Publishing, 2018, pp. 19–34.

[31] D. I. Mattos, J. Bosch, and H. H. Olsson, "Multi-armed bandits in the wild: Pitfalls and strategies in online experiments," *Information and Software Technology*, vol. 113, pp. 68–81, Sep. 2019.

[32] R. Kohavi, A. Deng, B. Frasca, T. Walker, Y. Xu, and N. Pohlmann, "Online controlled experiments at large scale," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13*, ACM Press, 2013.

[33] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, Jul. 1958.

[34] B. Scholkopf, F. Locatello, S. Bauer, N. R. Ke, N. Kalchbrenner, A. Goyal, and Y. Bengio, "Toward causal representation learning," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 612–634, May 2021.

[35] D. Tang, A. Agarwal, D. O'Brien, and M. Meyer, "Overlapping experiment infrastructure: More, better, faster experimentation," in *Proceedings 16th Conference on Knowledge Discovery and Data Mining*, Washington, DC, 2010, pp. 17–26.

[36] P. W. Holland, "Statistics and causal inference," vol. 81, no. 396, pp. 945–960, Dec. 1986.

[37] D. B. Rubin, "Bayesian inference for causal effects: The role of randomization," *The Annals of Statistics*, vol. 6, no. 1, Jan. 1978.

[38] J. Pearl, "On the consistency rule in causal inference," *Epidemiology*, vol. 21, no. 6, pp. 872–875, Nov. 2010.

[39] S. R. Cole and C. E. Frangakis, "The consistency statement in causal inference," *Epidemiology*, vol. 20, no. 1, pp. 3–5, Jan. 2009.

[40] D. Card and A. Krueger, "Minimum wages and employment: A case study of the fast food industry in new jersey and pennsylvania," *American Economic Review*, vol. 84, Apr. 1993.

[41] D. L. Thistlethwaite and D. T. Campbell, "Regression-discontinuity analysis: An alternative to the ex post facto experiment.," *Journal of Educational Psychology*, vol. 51, no. 6, pp. 309–317, 1960.

[42] J. Angrist and G. Imbens, "Identification and estimation of local average treatment effects," Tech. Rep., Feb. 1995.

[43] D. Westreich and S. R. Cole, "Invited commentary: Positivity in practice," *American Journal of Epidemiology*, vol. 171, no. 6, pp. 674–677, Feb. 2010.

[44] R. van de Schoot, S. Depaoli, R. King, B. Kramer, K. Märtens, M. G. Tadesse, M. Vannucci, A. Gelman, D. Veen, J. Willemsen, and C. Yau, "Bayesian statistics and modelling," *Nature Reviews Methods Primers*, vol. 1, no. 1, Jan. 2021.

[45] J. Normington, E. Lock, C. Carlin, K. Peterson, and B. Carlin, "A bayesian difference-in-difference framework for the impact of primary care redesign on diabetes outcomes," *Statistics and Public Policy*, vol. 6, no. 1, pp. 55–66, Jan. 2019.

[46] L. Li and E. T. Donnell, "Incorporating bayesian methods into the propensity score matching framework: A no-treatment effect safety analysis," *Accident Analysis & Prevention*, vol. 145, p. 105 691, Sep. 2020.

[47] S. Chib and L. Jacobi, "Bayesian fuzzy regression discontinuity analysis and returns to compulsory schooling," *Journal of Applied Econometrics*, vol. 31, no. 6, pp. 1026–1047, Aug. 2015.

[48] S. Gupta, L. Ulanova, S. Bhardwaj, P. Dmitriev, P. Raff, and A. Fabijan, "The anatomy of a large-scale experimentation platform," in *2018 IEEE International Conference on Software Architecture (ICSA)*, They mentioned useful to run A/A test, IEEE, Apr. 2018.

[49] X. Amatriain, "Beyond data: From user information to business value through personalized recommendations and consumer science," *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2013.

[50] J. Bosch and U. Eklund, "Eternal embedded software: Towards innovation experiment systems," in *Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change*, Springer Berlin Heidelberg, 2012, pp. 19–31.

[51] E. A. Stuart, "Matching methods for causal inference: A review and a look forward," *Statistical Science*, vol. 25, no. 1, pp. 1–21, Feb. 2010.

[52] Z. Xu and J. D. Kalbfleisch, "Propensity score matching in randomized clinical trials," *Biometrics*, vol. 66, no. 3, pp. 813–823, Nov. 2009.

[53] P. R. Rosenbaum and D. B. Rubin, "Reducing bias in observational studies using subclassification on the propensity score," *Journal of the American Statistical Association*, vol. 79, no. 387, pp. 516–524, Sep. 1984, PSM iterative process.

[54] E. A. Stuart and N. S. Lalongo, "Matching methods for selection of participants for follow-up," *Multivariate Behavioral Research*, vol. 45, no. 4, pp. 746–765, Aug. 2010.

[55] B. Kitchenham, T. Dyba, and M. Jorgensen, "Evidence-based software engineering," in *Proceedings. 26th International Conference on Software Engineering*, IEEE Comput. Soc.

[56] M. Ali Babar and H. Zhang, "Systematic literature reviews in software engineering: Preliminary results from interviews with researchers," Oct. 2009, pp. 346–355.

[57] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, Dec. 2008.

[58] J. Maxwell, "Understanding and validity in qualitative research," *Harvard Educational Review*, vol. 62, no. 3, pp. 279–301, Sep. 1992.

[59] G. Walsham, "Interpretive case studies in IS research: Nature and method," *European Journal of Information Systems*, vol. 4, no. 2, pp. 74–81, May 1995.

[60] M. Broy, "Challenges in automotive software engineering," in *Proceeding of the 28th international conference on Software engineering - ICSE '06*, ACM Press, 2006.

[61] A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch, "The benefits of controlled experimentation at scale," in *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, IEEE, Aug. 2017.

[62] P. Dmitriev, S. Gupta, D. W. Kim, and G. Vaz, "A dirty dozen: Twelve common metric interpretation pitfalls in online controlled experiments," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17, Halifax, NS, Canada: Association for Computing Machinery, 2017, pp. 1427–1436, ISBN: 9781450348874.

[63] B. Kitchenham, "Procedures for performing systematic reviews," Department of Computer Science, Keele University, UK, Keele University. Technical Report TR/SE-0401, 2004.

[64] D. K. Vasthimal, P. K. Srirama, and A. K. Akkinapalli, "Scalable data reporting platform for A/B tests," in *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, IEEE, May 2019.

[65] D. C. Montgomery, *Design and analysis of experiments*. John wiley & sons, 2017.

[66] D. B. Rubin, "Using propensity scores to help design observational studies: Application to the tobacco litigation," *Health Services and Outcomes Research Methodology*, vol. 2, no. 3/4, pp. 169–188, 2001.

[67] D. B. Rubin and N. Thomas, "Matching using estimated propensity scores: Relating theory to practice," *Biometrics*, vol. 52, no. 1, p. 249, Mar. 1996.

[68] M. A. Brookhart, S. Schneeweiss, K. J. Rothman, R. J. Glynn, J. Avorn, and T. Stürmer, "Variable selection for propensity score models," *American Journal of Epidemiology*, vol. 163, no. 12, pp. 1149–1156, Apr. 2006.

[69] B. B. Hansen, "Full matching in an observational study of coaching for the SAT," *Journal of the American Statistical Association*, vol. 99, no. 467, pp. 609–618, Sep. 2004.

[70] T. Treasure and K. D. MacRae, "Minimisation: The platinum standard for trials?" *BMJ*, vol. 317, no. 7155, pp. 362–363, Aug. 1998.

[71] Y. Liu, J. Bosch, H. H. Olsson, and J. Lantz, "An architecture for enabling A/B experiments in automotive embedded software," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, IEEE, Jul. 2021.

[72] Y. Zhang, H. Li, N. Sze, and G. Ren, "Propensity score methods for road safety evaluation: Practical suggestions from a simulation study," *Accident Analysis & Prevention*, vol. 158, p. 106 200, Aug. 2021, PSM: applied in traffic safety.

[73] W. G. Cochran and S. P. Chambers, "The planning of observational studies of human populations," *Journal of the Royal Statistical Society. Series A (General)*, vol. 128, no. 2, p. 234, 1965.

[74] N. Ramasubbu and R. Balan, "The impact of process choice in high maturity environments: An empirical analysis," Jan. 2009, pp. 529–539.

[75] M. Tsunoda and S. Amasaki, "On software productivity analysis with propensity score matching," in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, IEEE, Nov. 2017.

[76] R. Torkar, R. Feldt, and C. A. Furia, "Bayesian data analysis in empirical software engineering: The case of missing data," in *Contemporary Empirical Methods in Software Engineering*, Springer International Publishing, 2020, pp. 289–324.

[77] M. D. Hoffman and A. Gelman, "The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo," Nov. 18, 2011.

[78] P. C. Austin, "Optimal caliper widths for propensity-score matching when estimating differences in means and differences in proportions in observational studies," *Pharmaceutical Statistics*, vol. 10, no. 2, pp. 150–161, Mar. 2011.

[79] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. A. Szerlip, P. Horsfall, and N. D. Goodman, "Pyro: Deep universal probabilistic programming," *J. Mach. Learn. Res.*, vol. 20, 28:1–28:6, 2019.

[80] B. B. Hansen and S. O. Klopfer, "Optimal full matching and related designs via network flows," *Journal of Computational and Graphical Statistics*, vol. 15, no. 3, pp. 609–627, Sep. 2006.

[81] G. W. Imbens, "The role of the propensity score in estimating dose-response functions," *Biometrika*, vol. 87, no. 3, pp. 706–710, 2000, ISSN: 00063444.

[82] Y. Liu, D. I. Mattos, J. Bosch, H. H. Olsson, and J. Lantz, "Size matters? or not: A/B testing with limited sample in automotive embedded software," in *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, IEEE, Sep. 2021.

[83] ——, "Bayesian propensity score matching in automotive embedded software engineering," in *2021 28th Asia-Pacific Software Engineering Conference (APSEC)*, IEEE, Dec. 2021.

[84] D. I. Mattos and Y. Liu, "On the use of causal graphical models for designing experiments in the automotive domain," Apr. 2022.

[85] D. S. Lee and D. Card, "Regression discontinuity inference with specification error," *Journal of Econometrics*, vol. 142, no. 2, pp. 655–674, Feb. 2008.

[86] C. A. Furia, R. Feldt, and R. Torkar, "Bayesian data analysis in empirical software engineering research," *IEEE Transactions on Software Engineering*, pp. 1–1, 2019.

[87] D. I. Mattos, J. Bosch, and H. H. Olsson, "Statistical models for the analysis of optimization algorithms with benchmark functions," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 6, pp. 1163–1177, 2021.

[88] K. H. Brodersen, F. Gallusser, J. Koehler, N. Remy, and S. L. Scott, "Inferring causal impact using bayesian structural time-series models," *The Annals of Applied Statistics*, vol. 9, no. 1, Mar. 2015.

[89] N. K. Ahmed and J. Kapadia, "Seasonality effect on electric vehicle miles traveled in electrified vehicles," *SAE International Journal of Alternative Powertrains*, vol. 6, no. 1, pp. 47–53, Mar. 2017.

[90] J. Bao, P. Liu, and S. V. Ukkusuri, "A spatiotemporal deep learning approach for citywide short-term crash risk prediction with multi-source data," *Accident Analysis & Prevention*, vol. 122, pp. 239–254, Jan. 2019.

[91] M. B. Kamel and T. Sayed, "Accounting for seasonal effects on cyclist-vehicle crashes," *Accident Analysis & Prevention*, vol. 159, p. 106 263, Sep. 2021.

[92] D. D. Chaplin, T. D. Cook, J. Zurovac, J. S. Coopersmith, M. M. Finucane, L. N. Vollmer, and R. E. Morris, "The internal and external validity of the regression discontinuity design: A meta-analysis of 15 within-study comparisons," *Journal of Policy Analysis and Management*, vol. 37, no. 2, B. S. Barnow, Ed., pp. 403–429, Feb. 2018.

[93] S. Geneletti, A. G. O'Keeffe, L. D. Sharples, S. Richardson, and G. Baio, "Bayesian regression discontinuity designs: Incorporating clinical knowledge in the causal analysis of primary care data," *Statistics in Medicine*, vol. 34, no. 15, pp. 2334–2352, Mar. 2015.

[94] C. Glymour, K. Zhang, and P. Spirtes, "Review of causal discovery methods based on graphical models," *Frontiers in Genetics*, vol. 10, Jun. 2019.

[95] G. W. Imbens and J. D. Angrist, "Identification and estimation of local average treatment effects," *Econometrica*, vol. 62, no. 2, p. 467, Mar. 1994.

[96] Hevner, March, Park, and Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, p. 75, 2004.

[97] K. Peffers, T. Tuunanen, M. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, vol. 24, pp. 45–77, Jan. 2007.

[98] J. Wang, "To use or not to use propensity score matching?" *Pharmaceutical Statistics*, vol. 20, no. 1, pp. 15–24, Aug. 2020.

[99] K. Gao, M. Shao, K. W. Axhausen, L. Sun, H. Tu, and Y. Wang, "Inertia effects of past behavior in commuting modal shift behavior: Interactions, variations and implications for demand estimation," *Transportation*, Jul. 2021.

[100] A. Kahn-Lang and K. Lang, "The promise and pitfalls of differences-in-differences: Reflections on *16 and Pregnant* and other applications," *Journal of Business & Economic Statistics*, vol. 38, no. 3, pp. 613–620, Apr. 2019.

[101] J. D. Angrist and J.-S. Pischke, "The credibility revolution in empirical economics: How better research design is taking the con out of econometrics," *Journal of Economic Perspectives*, vol. 24, no. 2, pp. 3–30, May 2010.

[102] G. W. Imbens and T. Lemieux, "Regression discontinuity designs: A guide to practice," *Journal of Econometrics*, vol. 142, no. 2, pp. 615–635, Feb. 2008.

[103] J. McCrary, "Manipulation of the running variable in the regression discontinuity design: A density test," *Journal of Econometrics*, vol. 142, no. 2, pp. 698–714, Feb. 2008.

[104] L. Bottou, J. Peters, J. Quiñonero-Candela, D. X. Charles, D. M. Chickering, E. Portugaly, D. Ray, P. Simard, and E. Snelson, "Counterfactual reasoning and learning systems: The example of computational advertising," *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 3207–3260, Jan. 2013, ISSN: 1532-4435.

[105] B. Jiang, X. Shi, H. Shang, Z. Geng, and A. Glass, "A framework for network ab testing," Oct. 24, 2016.

[106] C. Shi, X. Wang, S. Luo, H. Zhu, J. Ye, and R. Song, "Dynamic causal effects evaluation in a/b testing with a reinforcement learning framework," *Journal of the American Statistical Association*, pp. 1–13, Mar. 2022.

[107] S. Wager and S. Athey, "Estimation and inference of heterogeneous treatment effects using random forests," *Journal of the American Statistical Association*, vol. 113, no. 523, pp. 1228–1242, Jun. 2018.

[108] E. Bareinboim and J. Pearl, "Causal transportability with limited experiments," in *the 27th AAAI Conference on Artificial Intelligence*, Jan. 2013, pp. 95–101.

[109] ——, "A general algorithm for deciding transportability of experimental results," *Journal of Causal Inference*, vol. 1, no. 1, pp. 107–134, May 2013.