CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Causal Models Applied to Studies within the Mining Software Repository Domain

Master's Thesis in Computer Science and Engineering

AMANDA LEVINSSON
LINNÉA FRANSSON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Master's thesis 2024

Causal Models Applied to Studies within the Mining Software Repository Domain

AMANDA LEVINSSON
LINNÉA FRANSSON

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

Causal Models Applied to Studies within the Mining Software Repository Domain

AMANDA LEVINSSON
LINNÉA FRANSSON

Supervisor: Richard Torkar, Department of Computer Science and Engineering
Co-supervisor: Hans-Martin Heyn, Department of Computer Science and Engineering
Examiner: Robert Feldt, Department of Computer Science and Engineering

Master's Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2024

Causal Models Applied to Studies within the Mining Software Repository Domain

AMANDA LEVINSSON
LINNÉA FRANSSON
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Context: Research conducted in the mining software repository domain commonly utilize observational data, due to software repositories serving as a rich source of such data. Simultaneously, there is a clear lack regarding the incorporation of causality in Software Engineering (SE) research, whilst statistical analyses often are conducted.
Objective: To analyse the practical implications of applying causal models to studies from the Mining Software Repository (MSR) conference. Specifically, it is of interest to examine whether researchers accidentally have included variables (colliders) in their analyses which have biased their results.
Method: A computer simulation was utilized as research methodology. This included the steps of (1) identifying a paper with colliders by sampling from the MSR conference and constructing Directed Acylic Graphs (DAGs), (2) a theoretical computer simulation of an SE scenario to prove collider effects, (3) computer simulations utilizing generated synthetic data based on the identified research paper. In addition, an analysis was conducted using the original data from chosen paper.
Results: A lack of transparency amongst the research investigated was identified, where variable selection processes and underlying assumptions were not completely clear. Three papers were investigated in the first step of constructing DAGs. Subsequently, colliders were identified in the paper of Nagy and Abdalkareem [46]. Simulations revealed that the exclusion of collider variables improved the sought after effect sizes. However, no practical implications were possible to determine. Replication package available [1].
Conclusion: A lack of transparency hindered the construction of DAGs, and indicated a threat to advancements in research. This, due to the need of interpreting authors' assumptions in their research. An incorporation of causality and DAGs could, due to the increased transparency it would bring, in the long run result in more robust advancements in research. Additionally, DAGs are recommended as tools to mitigate the risk of accidentally conditioning on colliders.

Keywords: Empirical Software Engineering, Colliders, Directed Acyclic Graphs, DAGs, Mining Software Repository Research, Causal Inference, Bayesian Statistics.

[1] `https://github.com/amalev18/colliders-msr`

Acknowledgements

We would first and foremost like to express our sincere gratitude to our supervisors Richard Torkar and Hans-Martin Heyn. They have contributed with guidance throughout the entire project with their expertise, active participation, and optimism when challenges occurred.

In addition, we have gotten a large amount of valuable insights from Nathan Cassee. We would like to thank you for engaging in this study, and providing your expertise.

Lastly, a thank you should be directed to Dan Gopstein and Justin Cappos for promptly providing the csv-files missing from their replication package.

Amanda Levinsson & Linnéa Fransson, Gothenburg, 2024-06-25

Contents

List of Figures

List of Tables

1

Introduction

In the field of Software Engineering (SE), software repositories serve as a valuable source of observational data [75], facilitating observational research in the research area of mining software repository. This research often relies on retrospective data, extracted from repositories with unknown collection process not designed for software engineering studies [27]. Due to the vast amount of data available, various statistical analyses are commonly employed in mining software repository research [44, 59, 49]. Key tasks in such analyses include identifying correlations in the data and establishing causal relationships between variables. Correlational analysis assess whether two variables change together, often referred to as co-linearity, while causality examines whether one variable influences changes in another variable [19]. It is important to note that correlation does not imply causation, as clarified by Furia, Torkar, and Feldt [19]. The authors further elaborate that incorporating causality improve both the robustness and generalizability of the scientific findings, reducing the reliance on solely associational interpretations of the data. In recent years, the importance of causality has been given increased attention, with researchers employing a causal methodologies receiving the Nobel Prize in 2019 and 2021 [66, 65]. In both cases, the researchers applied causal approaches to the research strategies field experiment and field study in their conducted studies. These types of research strategies are commonly seen in research conducted in the SE domain [71], indicating the importance and impact of utilizing causal inference in SE research.

To estimate causal effects, Randomized Controlled Trials (RCTs) are widely used as the gold standard. However, conducting RCTs is not always feasible due to practical or ethical reasons [19, 68]. Instead, causal effects can be estimated from observational data with other methods [68]. Siebert [68] describes that this approach requires researchers to address potential confounding factors in the data with causal inference, since it otherwise could bias sought estimations of causal effects. Despite causal inference being a core task in science, the topic of causal inference is although lacking in the field of SE. Hernán [29] describes how there is a reluctance amongst researchers to utilize the word causal when describing the goal of their research. Instead of accurately classifying research aims as causal, it is addressed as associational. As a result, confusion arises at fundamental stages in the scientific process, and mistakes are inevitably made [29]. Hernán, Hsu, and Healy [31] describe how there still remain prevalent misunderstandings and confusion in scientific research, due to a century-old refusal to address causal questions.

Mitra, Roy, and Small [42] discusses the temptation among researchers to rely on black-box approaches when addressing scientific questions. This given the extensive amount of data available, and various tools like Machine Learning (ML) and Artificial Intelligence (AI). In mining software repository research, it can be appealing to utilize the large amount of data available, feed them into an ML tool, and quickly receive statistical results to answer some stated research questions. There has been a notable increase in the utilization of statistical analysis of quantitative data in general SE research [15]. However, this combined with the lack of causal inference in research methodologies, and possible confusion about concrete aims of the scientific research, lies a clear risk. Researchers may utilizing statistical methods without knowledge of the underlying principles. Tools like the ML and AI are incapable of drawing causal conclusions, and solely search for correlations in the data [48]. Therefore, these tools are not suitable for research with causal aims. If this is done regardless, there is a possibility of receiving biased results, and further drawing inaccurate conclusions.

For research with purely predictive aims, one might argue that causal insights are unnecessary. Nevertheless, incorporating causality has been shown to be useful regardless. Pfister, Bühlmann, and Peters [57] explain that utilizing causal predictors allows for more robust predictions in the long run, as the statistical model remains invariant even when environment changes. Feder et al. [17] supports this view, describing that uninterpretable black-box predictors might be insufficient when a predictive model is employed in a high-stake scenario. Variables may appear to be useful when predicting an outcome, whilst being completely unrelated to the actual behavior of the outcome variable. Harris, Terry, and Truman [25] illustrate this by using margarine consumption as a predictor for divorce rate. The paper describes that the two variables are highly correlated or co-linear, and the result is statistically significant. Based on this analysis, it might seem reasonable to include margarine consumption in a predictive model. However, unless the purchasing pattern of margarine directly causes divorce, this predictor will solely be useful until a change in the environment affects a change in either of the two behavioral patterns. Even if the research aims are purely predictive, adding causality will elevate how robust and generalizable the scientific findings are [19].

What distinguishes causality from prediction, is the need for expert knowledge. This knowledge is essential to build a perception of the causal system under study and how various variables in the system relate to each other [31]. A prominent conceptual framework developed for this purpose is Directed Acyclic Graphs (DAGs) [48]. DAGs visually reveal causal relationships between variables and provide guidelines for further statistical inquiries. They are particularly useful when investigating treatment effects. In this context, the treatment variable represents the factor under investigation to determine its causal effect on the outcome variable, while the outcome variable is the variable of interest whose changes are analysed in response to the treatment [48]. By leveraging the insights provided by DAGs, researchers can determine which variables should be included in the statistical model when studying a causal relationship. Some variables will be necessary for obtaining correct estimates from the statistical analysis, while others should not be included. The lat-

ter scenario refers to cases involving a collider in the causal system, which is one of the patterns possible to identify in a DAG. A collider is the type of construct where the treatment variable and outcome variable share a common effect [40]. Including the shared effect variable in the statistical model would create a non-causal association between the treatment variables and the outcome variable, therefore inducing bias in the final result [48].

Causal salad is a term in causal inference that describes the approach of randomly adding variables to a statistical model without understanding their actual connection to the outcome variable. The changes in statistical estimates from including different variables are then used to draw causal conclusions [40]. This approach is in particularly harmful when collider variables are included, as they will bias the result. Causal analysis is still lacking in current methodologies applied in SE research [68, 19], even though statistical analyses are commonly performed. Given the enormous amount of data available and the retrospective data collection methods, the risk of including collider variables when performing mining software repository research is considerable. This thesis aims to highlight the effect of including a collider variable in statistical analyses conducted in observational SE research through a series of robust simulations. Real research scenarios, collected from the Mining Software Repository (MSR) conference [1], were used in these simulations. The MSR conference contains research that exploits data available in software repositories. Showcasing the practical implications of including collider variables emphasizes the effects they can have on research quality. Ensuring research quality is crucial for many stakeholders. Research aims to understand phenomena, and an altering of its quality can lead to misleading conclusions. This could affect future researchers who build on potentially erroneous research, hindering effective innovation. Additionally, stakeholders could be directly influenced by the outcome of the research, especially when decisions and modifications are based on the conducted research.

The remainder of the thesis is organized as follows. First the concrete purpose of the study, along with the research questions and limitations. Section 2 provides relevant background information. Section 3 elaborates on related work in the field and situates this thesis within it. Section 4 outlines the methodology used, followed by Section 5, which presents the results. Finally, Section 6 provides a discussions and guidelines for researchers, and Section 7 presents the conclusions.

1.1 Purpose of the Study

The purpose of this study is to investigate the practical implications of applying DAGs to a study from the MSR conference that includes collider variables in its statistical analysis. The results are expected to uncover and visualize previously unexplored causal relationships amongst variables, contributing to a deeper understanding of the factors influencing the estimates. This thesis provides advise to researchers on how to approach the analysis of MSR data, demonstrating how excluding variables influences the conclusions drawn. This study will benefit software

[1] `https://www.msrconf.org/`

developers, data scientists, and researchers by providing insights into the complexity of identifying causal relationships and improving the reliability and assumptions within observational studies.

1.2 Research Questions

With the goal of improving the quality of observational research in SE by adding causality, two research questions have been formulated:

- **RQ1**: What are the benefits and challenges of applying a causal approach to studies within the MSR domain?
- **RQ2**: Why is it necessary to mitigate potential challenges of conditioning on colliders within the MSR domain?

The first research question investigates the current status of perceived transparency and reproducibility in MSR studies. It evaluates the efficiency of adding a causal perspective to statistical analyses in mining software repository research and examines the practical implications of including collider variables. The second research question explores the effects of including colliders in computer simulation. This is investigated through two different cases: a theoretical SE example and synthetic data generated from the real research scenario in RQ1. This includes comparing statistical estimates with and without considering colliders, and analyzing the resulting differences in outcomes given a true effect.

1.3 Limitations and Delimitations

The thesis aims to investigate the research area of software repository mining, focusing specifically on studies from the MSR Conference. The primary objective is to examine the construct of colliders and whether they were incorrectly conditioned on in observed studies. Colliders are particular relevant in this context due to the retrospective data collection in MSR, were data is not purposefully gathered by researchers, increasing the risk of including colliders. To mitigate this risk, it is essential to make explicit assumptions about the research domain. Additionally, colliders presented an opportunity for analysis because their inclusion in an analysis is a matter of decision-making, as they are known and available variables. Unlike confounders, which can be challenging to control, due to the presence of unobserved variables. Therefore, the challenge with colliders lies in the awareness of handling these variables, rather than challenge of cost or difficulties of measuring them. Simply adding many variables to the model can address confounders, as they should be included in the analysis. However, this approach can be problematic for colliders, which also motivates delimiting to this construct. Colliders could potentially also be added by mistake based on their predictive power, which usually have large, but wrong, predictive power.

This work will specifically focus on DAGs and causal inference methods developed by Pearl [53]. Although there are alternative frameworks, such as the potential-

outcomes framework, also known as the Neyman-Rubin Causal Model [61], these will not be included in the scope of this study.

The generalizability of the conclusions drawn from this thesis could be discussed. From one aspect, the context investigated is very precise, focusing on studies within software repository mining, which might be seen as a low level of generalizability. On the other hand, the main topic, causal analysis, is to a high extent generalizable and applicable in a wide range of fields beyond software engineering and software repository mining.

2

Background

This chapter discusses the theoretical concepts utilized in the study. Section 2.1 presents common methodology used in MSR research. Section 2.2 explain the topic of causal inference, with descriptions of graphical causal models, the backdoor criterion, and total and direct effect. Section 2.3 introduces probability theory and Bayesian. Section 2.3.2 describes the use of Generalized Linear Models (GLM).

2.1 Methodology within MSR Research

Mining software repository research is a growing area in Software Engineering, relying heavily on processing and analyzing a large amounts of data [77]. It focuses on retrieving data related to software development and aims to extract knowledge about this data through statistical or data mining techniques. Example areas of interest in MSR studies include software evolution, bug prediction, and effort estimation [78]. According to Tutko, Henley, and Mockus [77], software development practices and tools have evolved over the past years, such as the increased use of version control systems and issue tracking tools, which have introduced new methods of data mining. Hemmati et al. [28] highlight both how the MSR research community has grown massively over the years, and the challenges in conducting MSR research. It includes handling a large number of development artifacts, tools, and analysis techniques for example. It is therefore collected in the paper by Hemmati et al. [28] a set of guidelines and best practices to conduct successful MSR research. These recommendations were gathered into the following themes, connecting to the typical MSR process: data acquisition and preparation, synthesis, analysis, and sharing and replication. One example of a recommendation, from the data acquisition and preparation theme, includes the importance of understanding the project, its domain, and the underlying development process when data is extracted for research.

Vidoni [78] outlines a four-step process in the MSR research methodology: sources, repositories, data, and information.

1. **Sources:** The first step is selecting sources, which are the directories where researchers typically will search for their repositories. Common sources are GitHub, SourceForge and Bugzilla.

2. **Repositories:** Next step includes the sampling from software repositories. Repositories are often referred to as the wells of data, containing the data but

are not the data itself. Some criteria for selecting these repositories could be programming language, creation or update dates, and repository popularity.

3. **Data:** The third step, is extracting the specific data from the repositories. This data can include open issue comments, commit frequency, or other relevant metrics.

4. **Information:** The final step is where various techniques and tools are applied to the extracted data to generate information and answer formulated research questions.

Despite the advancements in MSR, there seems to be a lack of acknowledged guidelines concerning how to conduct systematic MSR studies [78]. Although being considered evidence-based research, where researchers select repositories based on specific criteria and analyze the data to obtain evidence that answers their research questions, they seldom follow a systematic process. This poses a challenge for the field of MSR. Tutko, Henley, and Mockus [77] investigated the standard concerning the presentation of research methodologies in MSR papers through their thorough literature review. Due to the missing presence of a clear standard, one was proposed by the authors for researchers to follow. It was found that information regarding the complete methodology was often lacking, hindering the reproducibility of research. It was stated that a replication package commonly was accompanied by the research paper, which could be a reason for the partly lacking elaborations in the methodology according to Tutko, Henley, and Mockus [77].

2.2 Causal Inference

Causation refers to the cause-effect relationship between two variables, where one variable directly influences the other. The process of identifying and estimating these relationships is called causal inference. Understanding causation and why it is necessary to make sense of data is crucial [52]. This need can be exemplified by estimating the effects of smoking on lung cancer, education on salaries, or carbon emissions on the climate. Additionally, it is important to understand *how* and *why* causes influence their effects. Many researchers describe causal understanding as one of the fundamental goals of science [79, 19, 53].

Traditional statistical methods and data analysis usually aim to identify associations between variables, while causal analysis determines the causes and quantifies the effects [19]. When studying associations and finding strong linear relationships between variables it is called correlation. If there is a misleading correlation between two variables with no signs of cause-effect, such as the correlation between margarine consumption and divorce rates (as discussed in Section 1), it is called spurious association.

To illustrate the difference between various types of analysis, it can be referred to as Pearl's ladder of causality. Pearl [54] introduce a three-level causal hierarchy grounded in various types of tasks, categorized as association, intervention, and counterfactual. The bottom level, association (seeing), deals with questions such as

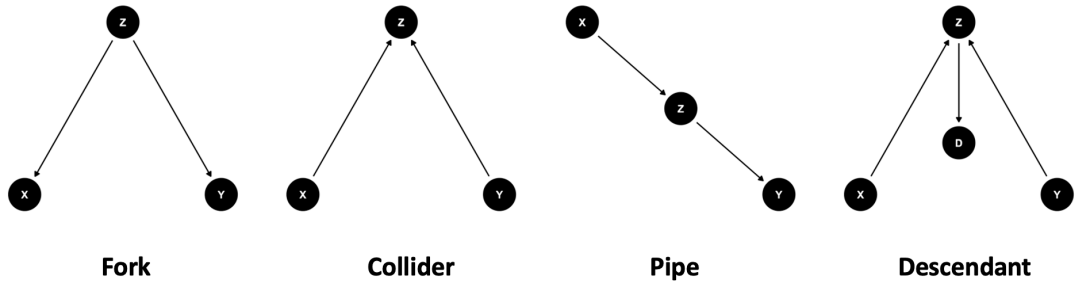


Figure 2.1: The four elemental confounds [40].

"How would seeing X change my belief in Y?". The next level, intervention (doing), can be exemplified by "What if I do X?". The top-level, counterfactual (imagining) can be addressed by "Was it X that caused Y?".

Similarly to Pearl, Hernán, Hsu, and Healy [31] classify tasks into three groups: description, prediction, and counterfactual prediction (also referred to as causal inference). Description involves using data to provide a summary of features of the world, such as computing the proportion of people with diabetes in a healthcare database. Prediction uses data to map some features of the world to other features, answering questions like "What is the probability of having a stroke next year for women with certain characteristics?". Counterfactual prediction uses data to predict features of the world as if the world had been different, for example, studying the mortality rate of a population that had received screening for colorectal cancer versus those that had not.

In general, the gold standard for estimating causal effects is through random controlled trials (RCT) [68]. However, in many cases, random experiments are not possible due to impracticability, costs, or ethical issues [79]. Therefore, other causal inference methods are used to estimate causal effect from observational data [68]. One such method, based on the use of graphical causal models (described in Section 2.2.1), will be further explained in Section 2.2.2.

2.2.1 Graphical Causal Models

Graphical causal models are visual frameworks used to represent hypotheses about causality within a given system [68]. They are valuable for questioning the validity and assumptions made in the causal analysis [51]. The simplest graphical causal model is a DAG [40]. These graphs consist of nodes representing variables, connected by directed single-headed arrows [51]. These connections represent a direct causal effect [68], indicating that changes in one variable cause changes in another, and not the reverse. In DAGs, the relationships are acyclic, meaning there are no paths leading back to a node [51].

The Elementary Confounds

An advantage of DAGs is their capability to identify structures that may lead to

spurious associations and address these issues before starting the estimation process [68]. It is crucial to understand the information flow within DAGs and determine which variables to condition on (or not). Conditioning on a variable, also known as adjusted for, filtered on, or controlling for, means including a variable in the analysis. When a variable is conditioned on, it can either allow a path of association between variables or a close association flow, depending on the diagrams configuration McElreath [40]. If a DAG includes unobserved variables that influence other variables a challenge arises, making it impossible to condition on such unobserved variables. There are four elemental confounding relations - fork, pipe, collider, and descendant - that serve as the building blocks for any type of DAG [40] (see Figure 2.1).

The first type of relation is a **fork**, alternatively known as a confounder, illustrated as X ←Z →Y. In this scenario, the treatment X and outcome Y are influenced by the same factor, leading to a correlation between them. By controlling or conditioning on Z, the common cause, the path of non-causal association is blocked. This conditioning on Z is essential to attain unbiased estimates of causal effects.

The next relation is a **collider**, illustrated as X →Z ←Y, showing that X and Y share a common effect. In contrast to the fork, this path is closed by default, and conditioning on Z would open a non-causal association path between X and Y.

The third relation is a **pipe**, also referred to as a chain [13], represented as X →Z →Y. The treatment X influences Z, which in turn influences the outcome Y. Similar to the fork, the path of association can be blocked by conditioning on Z.

The final type is a **descendant**, similar to the collider but with Z being influenced by D. Given the descendent D holds some information about Z, conditioning on D will also condition on Z, though to a lesser extent. Conditioning on the descendent will open the path of association, similar to a collider, but the consequences depend upon the nature of Z.

2.2.2 The Backdoor Criterion

With the use of DAGs, the relations that exist between variables in the system of interest become apparent. In order to get correct estimates of a treatment variable's effect on an outcome variable, there need to be no spurious associations biasing the result. As previously described, this is managed by conditioning on certain variables in order to close their influence on what is wanted to be estimated. This practice is often referred to as shutting the backdoor. Also formulated as a framework, the backdoor criterion includes finding and further blocking all confounding paths between the treatment and outcome variable. A path is blocked if the confounding variable is adjusted for, conditioned on. The backdoor criterion consists of four steps, and by following each one of these steps will help the identification of what confounding paths are present in the system and what needs to be conditioned on. It will also be clear what variables should not be conditioned on, since it would open

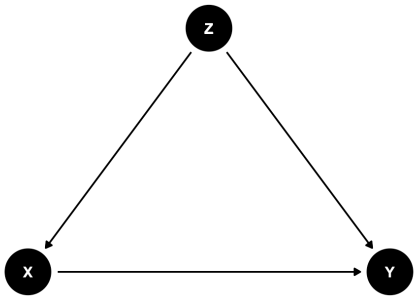


Figure 2.2: Exemplification of a DAG, where the treatments' (X) effect on the outcome (Y) is of interest. Present in this system is however also a confounder variable Z.

already closed paths [40]. Following the steps by McElreath [40], included in the backdoor criterion:

1. Identify and list every path that connects the treatment (X) and the outcome (Y).

2. Using the four constructs previously described, classify each path by whether it is open or closed. A path will solely be closed if a collider is present.

3. Classify according to whether each path is a backdoor path. Such a path will have an arrow entering the treatment variable X.

4. Identify whether any open backdoor paths are present. If so, determine which variable(s) to condition on in order to close these paths if possible.

To exemplify the use of the backdoor criterion, consider the DAG i Figure 2.2. In this example, the causal effect of X on Y is of interest. To be able to acquire an accurate estimate of this causal effect, it is essential to identify what additional variable(s) need to be included in the statistical model. The first step to achieve this goal according to the backdoor criterion, includes the identification of all paths in the system:

(1) $X \rightarrow Y$ (the direct path)

(2) $X \leftarrow Z \rightarrow Y$

There is one additional path apart from the direct one. When comparing this path to the four elementary confounds, it could be stated this is a confounder. Therefore, this path is open. It can also be seen that this is a backdoor path, given that there is an arrow entering the treatment variable (X). Combined, it can be stated that this is an open backdoor path. The last step in the backdoor criteria then states that these types of doors need to be closed. In this example, there are no other paths to consider, hence why it will be necessary to condition on Z (include Z in the statistical model).

2.2.3 Total and Direct Effect

When estimating a causal effect in a DAG, both the total and direct effect can be of interest. As described by Cinelli, Forney, and Pearl [13], the direct effect of a treatment X on an outcome Y is measured by the direct influence of X on Y that is independent of any variables. This is estimated by blocking the backdoor paths, as previously mentioned, to isolate the effect between X and Y. For the total effect, the overall impact of X on Y is of interest. This is estimated by not blocking the backdoor paths, and allowing indirect pathways to influence the outcome.

2.3 Probability Theory and Bayesian Inference

McElreath [40] describes Bayesian inference as simply the counting and comparing of possible events that could occur in the scenario under study. This explanation is exemplified in the section "The Garden of Forking Data" of the book written by McElreath [40]. In short, the example examines marbles in a bag and how many marbles of the two possible colors are present in the bag. An illustrative image in the example demonstrates all possible outcomes using all color combinations of the marbles. The results show that some combinations of colors in the bag are more likely than others, given the information presented in the scenario. Bayesian inference and building statistical models arise from this, where a model can learn from data and produce estimates. The goal is to obtain a posterior probability distribution. As McElreath [40] describes, the posterior distribution contains the information initially described in this paragraph: the relative number of ways each presumed cause of the data could have produced the data. The relative numbers provide information concerning the plausibility of the various presumed causes. Following are terminologies utilized in probability theory [40]:

1. **Likelihood P(B|A):** Describes how many ways a value can produce data, or how likely the observed data is under certain conditions.
2. **Prior probability P(A):** The prior plausibility or knowledge about a parameter before any data is observed.
3. **Posterior probability P(A|B):** The updated plausibility of a parameter after taking into account the observed data.

The mathematical definition of a posterior probability distribution arises from Bayes' Theorem as described by Pearl, Glymour, and Jewell [52]:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{2.1}$$

P(A|B) is the conditional probability of event A given event B occurs, and P(B|A) is the conditional probability of event B given event A occurs. P(A) is the prior probability of event A, and P(B) is the prior probability of event B [52]. This theorem is foundational when designing a Bayesian model [40]. It is used to for example compute the plausibilities after accounting for the data, as mentioned at the beginning of this section, which provides the posterior distribution.

2.3.1 Example of Building a Bayesian Model

Starting with building a linear regression model, there is a need to extract data for the model. In this example from McElreath [40], the aim is to use a Gaussian model of height in a population of adults, with weight as a predictor variable. The outcome variable, height, can be approximated to follow a normal (Gaussian) distribution, with its two parameters describing the distribution's shape, the mean μ, and standard deviation σ. This can be seen on the first row, called the likelihood, which is the probability of the observed height (h_i). The next row is a linear model of the mean, μ_i, where α is the intercept (mean when x_i=0), β is the change in expected height, also known as the slope, x_i is the weight on row i, $\bar{x}$ is the mean weight. The remaining lines are priors, defining distributions for the unobserved variables α, β, σ.

$$
\begin{aligned}
\mathrm{h}_i &\sim \text{Normal}(\mu, \sigma) && \text{[likelihood]}\\
\mu_i &= \alpha + \beta(\mathrm{x}_i - \bar{x}) && \text{[linear model]}\\
\mu &\sim \text{Normal}(178, 20) && [\alpha \text{ prior}]\\
\beta &\sim \text{Normal}(0, 10) && [\beta \text{ prior}]\\
\sigma &\sim \text{Uniform}(0, 50) && [\sigma \text{ prior}]
\end{aligned}
$$

2.3.2 Maximum Entropy Principle

When building a model, it is important to choose a distribution for the likelihood in order to include all the possible scenarios that could arise, together with the constraints on the outcome variable. McElreath [40] explains that the goal is to maximize the information entropy, also known as maximum entropy, to best represent the current state of knowledge. The information entropy is defined by the following function:

$$H(p) = -\sum_i p_i \log p_i$$

This function measures the uncertainty of a probability distribution p with probabilities p_i for the possible events i, based on the average-log-probability. The distribution with the highest information entropy is the one that can occur in the greatest number of ways, making it the most plausible distribution. It spreads the probability as evenly as possible, while remaining consistent with the knowledge about the process. This is known as the maximum entropy distribution.

2.4 Generalized Linear Models

A Generalized Linear Model (GLM) is a more general statistical model than a regular linear model. Section 2.3.1 presents an example of a regular linear model. In that example, the outcome variable is height, h, and is specified to follow the Normal distribution. For a GLM, the outcome variable is not assumed to follow a Normal distribution[40]. An example of a GLM is:

$$y_i \sim \text{Binomial}(n_i, p_i) \qquad \text{[likelihood]}$$
$$f(p_i) = \alpha + \beta(x_i - \bar{x}) \qquad \text{[linear model]}$$

In this example, the outcome variable y represents the number of heads observed in n trials of coin flips. The maximum entropy principle helps to pick the likelihood distribution for the outcome. In this case, y is a count variable representing the number of successful trials of coin flips. The Binomial distribution has the maximum entropy for this scenario, making it the least informative distribution that satisfies the prior knowledge of y.

When other likelihoods than the Normal distribution are used for the outcome variable, the scales of the outcome variable and the parameters can differ. This can easily be demonstrated by comparing the two examples mentioned. For the regular linear model in Section 2.3.1, the outcome variable is height and the parameter connected to the linear model is the mean height of a population. Both are measured in the same unit, centimeters for example. However, the scales differ between the outcome variable and parameters in the GLM. The outcome y in the exemplified GLM has some count unit, while the parameter connected to the linear model is a probability and hence unitless.

The function $f()$ inserted in the model above represents a link function, most commonly the log or logit link. Which to use depends on the likelihood of the outcome variable. The link function manages these various scales mentioned that could exist between the outcome and parameters, and additionally ensures constraints of the variables are kept. Possible constraints for a variable could be remaining between 0 and 1, like for the probability parameter in the statistical model above. For more in-depth elaborations and explanations of GLMs, McElreath [40] has excellent content in his book.

3

Related Work

The purpose of conducting a literature review is to provide a comprehensive overview of existing research, methodologies, and findings related to the study at hand. To the extent of our knowledge, there appears to be a lack of studies using a similar research approach, particularly with a focus on exploring and applying DAGs to existing studies within the field of SE, specifically addressing colliders. Consequently, the aim is then to explore existing work that has proposed causal models and other methodologies to address issues related to colliders or other causality concerns, both within the domain of SE and in other relevant fields. This exploration will allow for an explicit view on current landscape of research, enabling the positioning of this study in relation to others, and identification of gaps in the research area.

While causal inference remains relatively uncommon in SE, there is a growing number of papers applying some form of causal analysis to their data. Siebert [68] conducted a study aiming to understand the extent and application of Pearl's Statistical Causal Inference (SCI) framework to SE. By analyzing 25 SE papers published between 2010 to 2022, and categorizing them into fault localization, testing, performance analysis, and others, he found a recent but fragmented adoption of Pearl's framework. Chadbourne and Eisty [9] shared similar objectives, attempting to identify the SE fields suitability for causal inference, but encountered challenges due to the limited number of related papers in the SE domain. Siebert [68] proposed that the underutilization of SCI in SE may arise from researcher's unfamiliarity with the methods, their perception of complexity, or unnecessary application. However, recent publications targeting a wider audience and initiatives in other fields have aimed to make SCI more accessible. Despite ongoing challenges, there is a growing belief that SCI methods, well-suited for handling observational data, will gain traction in SE research.

There is a need of causality in SE research, as well as more transparency within the area of MSR research. Furia, Torkar, and Feldt [19] underscores the significant amount of observational data in SE research, and the challenge of drawing more than correlational conclusions from such data. In studies by Tutko, Henley, and Mockus [77] and Vidoni [78], it is also observed that there is a lack of information on how the researchers performed their data selection process within MSR research. Tutko, Henley, and Mockus [77] describe that a considerable number of papers provided little or no reproducibility instructions, as many as 33% provided no information about data retrieval. Similarly, Vidoni [78] describes that over a third of the observed

papers did not address how sources, repositories or inclusion/exclusion criteria were selected. Additionally, only 13% discussed biases related to both the repository selection and the data extraction. In empirical SE research, Furia, Torkar, and Feldt [19] emphasizes the necessity to employ conceptual and technical tools in order to incorporate modeling and reasoning about causal relations. They elaborate how neglecting this aspect significantly could restrict the generalizability and rigor of empirical results. Their research highly aligns with this study, indicating the need for adding a not only a causal perspective but also transparency to research within SE, utilizing observational data.

Building on the need for causal perspectives in SE, recent discussions of causal inference extends to related fields, including ML and AI. Mitra, Roy, and Small [42] discusses their list of top ten most likely directions for future research concerning causal inference, including precision medicine, quasi-experimental devices, and causal ML. It is described by the authors how causal ML aims at predicting potential outcomes resulting from alterations in some aspect of the world. This would, according to Mitra, Roy, and Small [42], require careful thought in regards to for example the design of the study, and what variables that should be included in which model. The paper concludes the importance of carefully elaborating and describing the problem under investigation, understanding the data, and thoroughly evaluating the plausibility of assumptions made. This, in comparison to leaning towards addressing research questions using a black-box approach. Nogueira et al. [48] similarly underscores the widespread use of a black-box approach, and the emergence of causality into ML research. It is described that this introduction of causality is due to the need for trustworthy AI, which is demanded given how black-box models increasingly are utilized when socially sensitive decisions are made. The authors elaborate that the reliance on such approaches, without an understanding of the underlying logic, may lead to both biased, and harmful decisions. Pearl [55] elaborates on how causality could be used as a tool to overcome limitations of ML. Forney and Mueller [18] also underscores how the AI community is turning to causality in order to address its limitations. They continue by stating that despite resounding appeals from distinguished researchers, causality remains a missing topic in a vast majority of traditional educational content in AI. Provided in the paper is further an educational primer in order to bridge the gap between causality and AI.

While causal understanding has notably gained traction in closely related fields such as ML and AI, its significance extends across a broad range of disciplines. Causality is found in biomedical research [38], marketing [60], epidemiology [16], and economics [64], amongst others. To exemplify the impact of causal methods, parallels can be drawn to a case study in ecology research that employs similar causal methodologies. In a study by Arif et al. [3], the researchers investigated the effects of climate-induced bleaching on coral reef ecosystems in Seychelles. Using the Structural Causal Modeling (SCM) framework and DAGs, their objective was to understand the factors influencing regime shifts versus potential recovery. Furthermore, they compared their approach with earlier studies on the same system, showcasing that previous findings relied on correlation analysis rather than causal inference, despite using a causal language. In results, the authors could reveal fac-

tors that were not evident in earlier studies. Using SCM and DAGs, allowed them to critically assess the observational data, carefully select which data best was best suited in answering the causal questions and remove spurious correlations.

A crucial aspect to accurately perform statistical analysis is the consideration and mitigation of biases, which is highlighted in the paper by Stanbouly et al. [70]. They describe a bias as a non-random error that can compromise the findings from a study. Specifically collider bias, often also referred to as selection bias, is the focus of this thesis. Numerous papers highlight the potential danger that lies in the context of collider bias, when not mitigated correctly [64, 32, 67]. Digitale et al. [16] clearly distinguishes that the issue lies in conditioning on the collider, and not in the presence of the collider itself. The graphical framework of DAGs is a common approach found in research that is used in order to systematically and visually detect colliders, amongst other confounding structures, which further lie the groundwork for managing them [67, 64, 30]. Schneider [64] elaborates on the commonness of collider bias in economic history research. Several examples of collider bias are brought up in the paper, showcasing scenarios of collider bias and how they additionally were mitigated in the research exemplified. Schneider brings up a study where it was investigated how breastfeeding in early infancy affects a child's weight in later stages of infancy. This scenario is illustrated with a DAG in Figure 3.1.

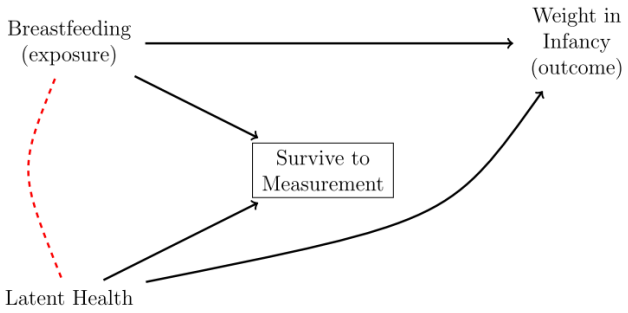


Figure 3.1: A DAG showcasing the studied scenario of the effect of early breastfeeding in infancy on a child's weight in later infancy. The squared box represent the variable that is conditioned on, and the red line illustrates the bias that is induced from conditioning on a collider. Illustration from [64].

An additional variable affecting the weight of an infant is the initial health of the newborn. As described by Schneider, the data used in the study solely included children who survived and were actually able to be measured in infancy. Survival of an infant is affected both by whether a child receives breastfeeding, and its initial health status. That is, both breastfeeding and the initial health share a common effect, survival of a child, resulting in a collider structure in the DAG. Collectively, the research design forces the conditioning on a collider, inducing bias in the analysis. The result from the study showcased an underestimation of the positive effects of exclusive breastfeeding on weight in infancy. This example showcases potential risks of not being aware of the causal structures in the system under investigation in a

study. Hernán and Monge [32] similarly highlight how the conditioning on a collider during the study design or data analysis phase, may lead to bias in the resulting effect estimates under investigation.

In recent years, researchers employing a causal approach in their research methodology have received recognition by recieving the Nobel Prize in Economic Sciences. In 2019, Banerjee, Duflo, and Kremer were acknowledged for transforming development economics through their experimental approach to understand and address global poverty [66]. Utilizing randomized control trials (RCTs) in field experiments, they assessed the causal impact of a specific intervention or program. In 2021, Angrist and Imbens were awarded the Nobel Prize for their methodological contributions to analyzing causal relationships on observational data [65]. They developed a framework for estimating an average treatment effect, particularly in analyzing scenarios in natural experiments.

The aim of this study is to uncover the promising prospects of integrating causality into SE research. Despite its significant impact in various disciplines, such as economics and epidemiology, casual inference methodologies remain underutilized within the field of SE [68]. However, the acknowledgment of causal inference's great potential in these other fields, as evidenced by the recent Nobel Prizes awards [66, 65], underscores the unexplored opportunities for SE and the relevance of this study. By highlighting the risks associated with disregarding causal structure, and emphasizing the need for transparency and robustness in research methodologies, the study seeks to empower SE researchers to make well-informed decisions and return reliable results. By bridging a gap between SE and causal inference methods, this study can drive a transformation in SE research, establishing causal methods as one of the standard approaches when conducting studies.

4

Methodology

A computer simulation was the research strategy utilized in order to investigate the consequences of including collider variables in mining software repository research. This applied approach aligns with Stol and Fitzgerald [71] notion of using simulations to model systems, and assess various scenarios within them. Variables utilized in the simulations were based upon the real variables used in the analysed research, and no manipulation or intrusion of the variables was possible due the studying of complete research. These are also key aspects of a computer simulation highlighted by Stol and Fitzgerald [71]. In addition to the simulation, the methodology included some elements from a field study. These were in regards to the exploratory aims of a field study, where the understanding of how things currently function in a field is of interest, as Stol and Fitzgerald [71] describe. Such aspects were identified and gathered during the process of collecting papers, and building DAGs, before conducting the simulation.

An overview of the research methodology is given in Figure 4.1. The methodology consisted of different parts illustrated in the figure, where the first part is marked with a dashed box in Figure 4.1. This part involved identifying a paper which incorrectly included a collider variable in its analysis. To find such a paper, a thorough data collection and sampling of papers was conducted. The aim of this approach was to find a representative paper within the MSR domain as study object. Initially, the search started in the ACM SIGSOFT Distinguished Paper Award category from the MSR conference, with a specified limit in regards to a papers publication date. For a paper to be considered as a candidate, it had to fulfill a set of predefined criteria. The defined criteria utilized can be studied in Figure 4.2. Given a suitable paper passing the criteria, a DAG was constructed based upon the description of the research domain in the paper and the variables used in the analysis that were included in the replication package. When collider variables were found to be incorrectly adjusted for in the analysis in the research, the subsequent part of the study could begin. Otherwise, the process of collecting data and filtering papers was restarted, but with an extension in regards to previously defined publication date limits and tracks of papers considered at the MSR conference. This first part of the study was therefore of iterative character, and it was repeated until a paper was found which incorrectly included a collider.

Before continuing an investigation with the paper identified in the sampling process, an independent simulation was conducted. This simulation included a simple and

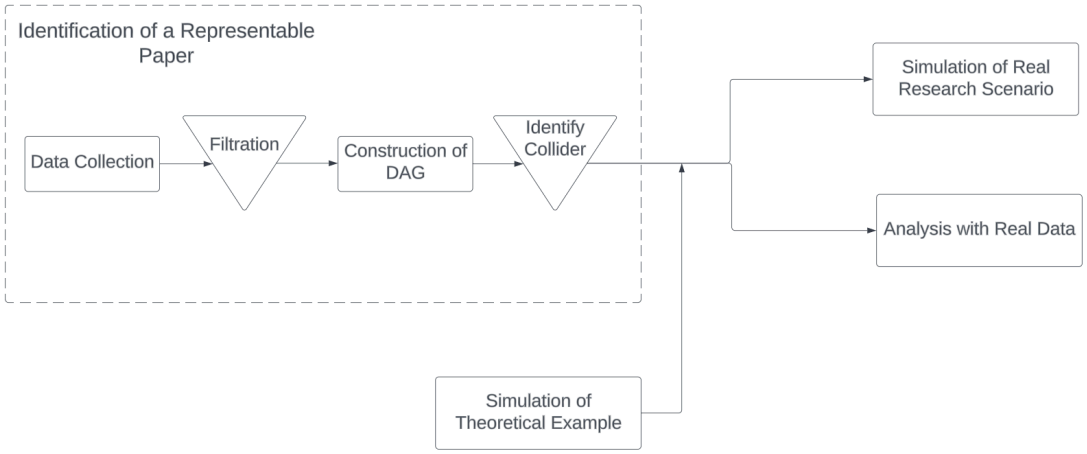


Figure 4.1: Illustrative overview of the methodology conducted in the research. A computer simulation was utilized as research methodology. The initial part of the research included finding a paper representable in SE which included a collider. This part is marked with the squared box. One simulation was conducted independently to motivate further simulation of the real research scenario. Lastly, an analysis was conducted using the real data in the found research paper.

theoretical SE scenario, and was conducted to present the theoretical effects of including a collider variable in a statistical analysis. The outcome of this simulation provided a motivation to why it was important to further investigate the real research scenario from the MSR community. The subsequent step following this examination, was to use these insights and investigate the impact of colliders in the found research paper by first simulating the real scenario. Based on the the DAG constructed in a previous step and the real data provided in the paper, data was simulated to capture the general characteristics of each real variable and the relationships between them. Statistical models were further defined using the variables included in the found study. One model was defined including the collider variables, and one without including them. Following the simulation, an analysis using the real data was conducted to investigate the practical implications in the found research scenario.

The remainder of this section includes a more in depth description of each separate step conducted in the thesis. Table 4.1 shows a breakdown of the underlying structure and aims of the thesis, which the method serves to fulfill. The breakdown is done using the goal-question-metric (GQM) framework by Basili and Caldiera [6]. As indicated by the metrics used to answer the stated research questions, both qualitative and quantitative data was collected during the study. Runeson and Höst [62] highlight the benefits of incorporating both qualitative and quantitative data when conducting research, in order to gain a deeper understanding of the phenomena under investigation.

Table 4.1: The GQM framework provides a connection between the higher goal of this thesis with concrete research question and metrics, allowing for a concrete evaluation.

Goal	**Purpose**	Improve
	Issue	the quality of
	Object	observational research in SE
		by adding causality
Research Question 1	**RQ1**	**What are the benefits and challenges of applying a causal approach to studies within the MSR domain?**
Metrics/Evidence	**M1**	Perceived transparency and reproducibility
	M2	Challenges encountered in the process of applying causal approaches
	M2	Differences in statistical estimates using real data from research paper
Research Question 2	**RQ2**	**Why is it necessary to mitigate potential challenges of conditioning on colliders within the MSR domain?**
Metrics/Evidence	**M1**	Differences in statistical estimates in computer simulation

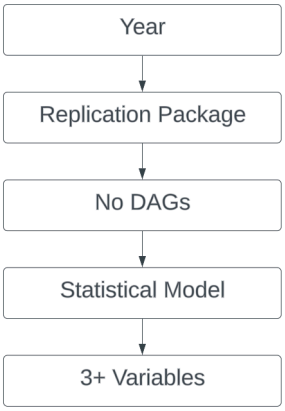


Figure 4.2: Illustrative overview of the sampling process, and the criteria each paper needed to fulfill in order to be considered.

4.1 Data Selection

This section includes the process of gathering data, and sampling from those collected papers in order to find suitable papers to construct DAGs upon. A purposive sampling strategy was applied using the guidelines provided by Baltes and Ralph [5], in order to carry out the data selection. This included a strategic selection process, as opposed to random selection, to collect suitable papers for further analysis. The purposive sampling strategy was chosen to obtain as in-depth information as possible from the sample size investigated, in order to find papers which represented the field as accurately as possible. The goal was however not to being able to claim generalizability from the final results. To follow the purposive sampling strategy, it was crucial to develop a systematic approach and identify papers with the necessary attributes for the final analysis. The MSR conference was chosen as the source of data to represent the field of interest, and papers were selected based on a set of pre-defined criteria. Papers were required to meet all criteria in order to be considered for further analysis. These criteria can be studied in Figure 4.2. More detailed discussions on the methodology will be provided in the upcoming sections, where Section 4.1.1 will describe the data collection, and Section 4.1.2 the sampling strategy.

4.1.1 Data Collection Methods

The selection of mining software repository research as the field of interest was based on its relevance to SE research, given the amount of observational data and various types of statistical analysis employed. The conference from which research papers were chosen was the MSR conference, which is co-located within the International Conference on Software Engineering (ICSE). The MSR conference stands out as the premiere conference in terms of the application of data science, machine learning, and artificial intelligence within SE. With the mission to improving SE practices through the analysis of the enormous amount of data from repositories [45]. Each

year, the MSR conference calls out for submissions across various tracks. These different tracks, have varying submission criteria and undergo thorough evaluation by a specialized committee to ensure quality and relevance before being presented at the conference.

For this study, papers were chosen from those either awarded the ACM SIGSOFT Distinguished Paper Award (DPA), or accepted for the Mining Challenge (MC). DPA papers are known to be outstanding, representing the best works in the field, with around three papers receiving this award annually. In contrast, MC is a competition were the participants are given the same dataset to address a challenge or explore a research question of interest, leading to varying numbers of accepted papers each year. The focus was to sample from the DPA papers due to them being the best papers within the field, but subsequently, MC papers were included as they address typical challenges in the field. Therefore, both tracks were considered representative of qualitative SE research. While the primary aim was not to prove generalizability in this study, an improvement in a single paper could indicate a broader need for adopting causal approach in observational research practices in SE.

4.1.2 Sampling Strategy

A sampling of the collected papers was subsequently carried out in order to select papers possible to build DAGs upon, and further find one paper to use for remainder of the analysis in the thesis. The selection process was done using a pre-defined criteria, and all papers needed to meet all criteria in order to be considered for further analysis. In addition to identifying suitable papers for subsequent steps, the formulated selection criteria additionally provided insights into the metrics regarding transparency and reproducibility of research papers at the conference. Two additional informational checks were formulated in this regard, functioning as partly a data source for these metrics, and partly as another potential source of collider identification in suitable papers. What follows are the specification of each criteria used during the sampling of papers, and the motivation behind each. Lastly, the two informational checks will be described.

Year
Initially, a specified time span had to be selected in order to provide an equitable representation of the current status in the research field, where the most recent research was prioritized. This also due to the recent emergence attention of causal inference after the groundbreaking work by Judea Pearl in the 2000s. Therefore, research conducted in the previous five years in the DPA track was firstly examined. If the subesequent steps turned out unsuccessful, the time span was extended, and the procedure was similarly carried out for the MC track.

After sampling by year of publication, the information extracted from each paper were divided into two phases. An overview of collected information can be found in Table 4.2. If the criteria in the first phase were not fulfilled, there was no interest in investigating the remaining criteria in the second phase. Consequently, information in phase two was only extracted from the papers passing the first sampling phase.

Table 4.2: Information extracted from MSR research papers and corresponding repositories. RP shortened for Replication Package.

Phase	Information extracted	Description	Source
1	Replication Package	Yes/No, URL	Paper
	No DAGs	Yes/No, Causality Discussion	Paper
2	Statistical Model	Yes/No, Type of test(s)	Paper & RP
	Variables	Variable Names for each model	Paper & RP
	Papers' Selection Criteria: Data	Yes/No, Steps when sampling	Paper
	Papers' Selection Criteria: Variables	Yes/No, Steps when sampling	Paper

Replication Package
In the first phase, an investigation was conducted whether a paper provided a replication package for their research. A replication package enabled access to the data utilized in the research. This would function as a source when identifying the exact variables basing their statistical analysis, and be a necessity when conducting the final statistical analysis. The investigation involved searching for references provided by the author where the availability to replicate their research would be stated. If a repository with necessary files to replicate the study was found, the criterion was considered fulfilled. If solely a data set was provided, the criterion was not considered fulfilled. A reconstruction of the work in such a research paper could technically have been made. However, due to time constraints and the risk of misinterpreting authors descriptions of their work, these papers were neglected.

No DAGs
Next step involved assessing whether a DAG was included, indicating that causality had been considered in the research paper. The investigation included searching for a figure visualizing a DAG, as well as using specific search terms such as directed acyclic graph, DAG, causal, bias, collider, and confounder. The criterion was fulfilled when no DAG was present with the purpose of addressing causality in the research conducted, and if the mentioned terms were missing. The criterion was also considered fulfilled if the terms were mentioned but in the purpose of stating that causal perspectives were neglected. This criterion was formulated since it would have been unnecessary to add a causal perspective to research already addressing the topic.

Statistical Model
The inclusion of statistical model was explored either through code in a replication package, mathematical notation, or clear descriptions in research paper. These models could be of various types, such as a Linear Regression Model (LRM) or Mann-Whitney U (MWU) test [7]. If a statistical model was present, the criterion was fulfilled.

3+ Variables
If a statistical model was identified, the next step was to analyze and document the variables used in each test. This process required studying the code in the repositories and extracting information from the research paper. From these variables it was identified whether or not a collider variable was adjusted for in a research paper. To be able to build a DAG, at least three variables are needed. The criterion was hence fulfilled if three or more variables were utilized in the statistical model.

The following are the two specified informational checks for which data was collected from suitable papers.

Papers' Selection Criteria: Data
This informational check investigated whether the authors clearly described their sampling strategy (i.e., how or why) for collecting their data. If the paper could distinguish the data sources used and any filtering decisions, the check was considered to have been successfully addressed within the research paper.

Papers' Selection Criteria: Variables
Similarly, if the paper could account for which variables were incorporated into their models and why, this check was considered addressed .

From all papers which fulfilled every criteria in Figure 4.2, one paper was chosen for the next step which seemed mostly promising in terms of identifying a collider. This included aspects such as having more statistical tests, and more variables utilized in the tests. If a paper was chosen and no collider was found, a new paper from the suitable papers was picked. When no suitable paper remained, the data collected was restarted and expanded.

4.2 Construction of DAGs

Given a paper which met all the criteria, the next step included the construction of DAGs based upon the research conducted in the paper. If several research questions containing a statistical analysis were present in the research, one separate DAG was constructed for each analysis. From the sampling process, it had been confirmed that at least three variables were present in the analysis conducted in the given paper, and which these variables were. In order to construct a DAG, the treatment- and outcome variable needed to be specified, which were supposed to show the sought after causal relationship in the studied research. This was done by studying the presented research question(s), and rationales in the paper.

Thereafter, the process of constructing a DAG based upon these insights was started. DAGitty[1] was the main tool used for this purpose, due to it being a acknowledged tool to use when drawing and analysing DAGs [74]. Each variable was firstly inserted as an entity in DAGitty. They were additionally marked as adjusted for, since

[1] `https://www.dagitty.net/dags.html#`

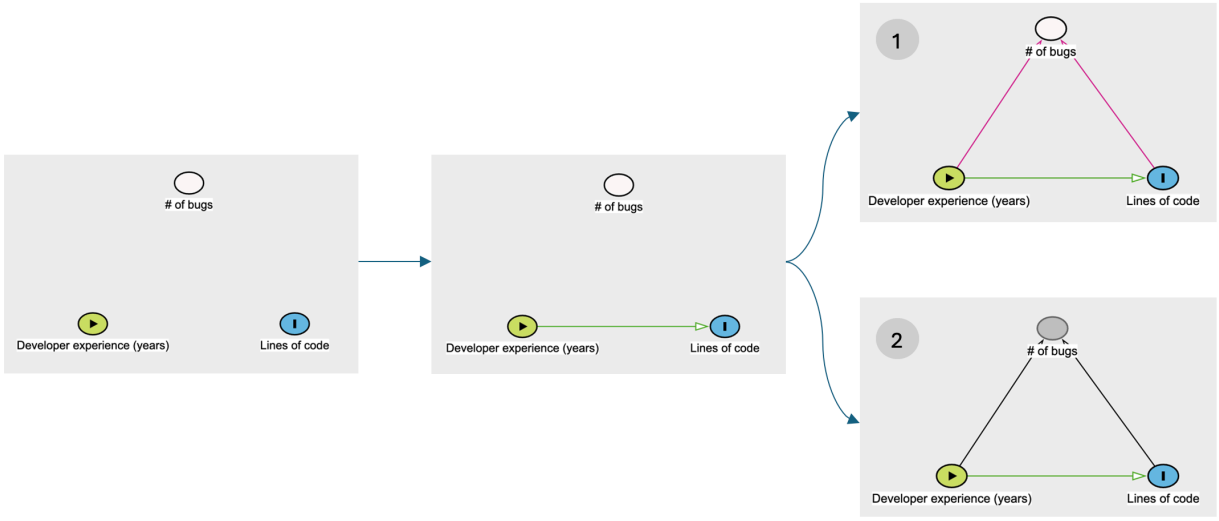


Figure 4.3: Illustration of the DAG construction process in DAGitty. The green variable illustrates the treatment variable, and the blue the outcome variable. If a predictor variable is white, it is included in the analysis. Otherwise, it is grey. The right hand side of the figure presents the difference between including a collider variable in DAGitty, as seen in picture 1, and not including it, as seen in picture 2.

each variable included in the DAG was included in the statistical analysis in the research. The treatment- and outcome variable were also specified. A first arrow was then drawn from the treatment variable to the outcome variable identified, showcasing the direct causal effect of the treatment on the outcome variable. Regarding remaining arrows, an arrow was drawn from one variable to another if there were arguments supporting that one variable had an influence on the change of the other variable. These arguments were both based on the authors own statements made in the paper, or from general domain knowledge in SE. Primarily, the authors own assumptions were considered. However, in the cases where these were not existing, or sufficient, general domain knowledge was utilized. Which source of information that was utilized when, will be distinguished when the results are presented.

To ensure each DAG constructed was as accurate as possible, the construction was made with a critical approach where each arrow was questioned to ensure robustness. This aspect was especially of importance when a collider was thought to have been found, since it needed to be confirmed that it truly was a collider. In these cases, the direction of the arrows was verified carefully. Analysis were also made in regards to if potential arrows could be directed out from the collider variable, which if so would affect the collider structure. The process of constructing DAGs is demonstrated with a general, and simple example in Figure 4.3. In includes the variables number of bugs, lines of code and developer experience, where the aim is to investigate the effect a developers' experience have on the amount of lines of code they write.

Once a DAG was constructed based upon a research paper, it was needed to be confirmed whether or not a collider structure was incorrectly included in the examined research paper. By using DAGitty and its labeling functionality, it could be utilized

to state whether the variables where adjusted for in a correct manner. This analysis is illustrated in Figure 4.3. On the right hand side in the figure, the difference between adjusting for a collider variable, as seen in picture 1, and by not adjusting for it, as seen in picture 2, is presented. DAGitty marks the relation red when a collider variable is included when it should not be, and clarifies that the DAG therefore is incorrectly adjusted. If this was shown to be the case for a given paper, it was chosen to be used for the remainder of this study. Otherwise, a new research paper was picked, and the previous steps were redone for a new paper. When a collider variable was identified in a DAG, an additional DAG was further constructed, showcasing the same scenario as the first DAG but including only the treatment variable, outcome and the two collider variables. This second DAG highlighted more clearly the colliders, since the DAG would be more simplified. In the upcoming analyses, the first DAG including all variables would be the basis for investigating the direct effect, whilst the second would be used when examining the total effect.

Additionally, it is important to consider possible scenarios explaining why the original authors might not have identified colliders in their studies. One possibility is that they were not aware of colliders and did not know how to correct for them. Another possibility is that the original authors implied a different DAG compared to the one constructed in this study, even though they did not explicitly integrate a DAG into the paper, which is a step in this study's sampling process. It is acknowledged that interpretations of relationships between variables might differ among researchers. Regardless of the possible scenarios, the construction of DAGs in this study was based on the assumptions made by the original authors in their text, supported by domain knowledge, and validated through discussions with supervisors.

4.3 Computer Simulation

Two types of simulations were conducted in this study. The first one was the independent simulation of a theoretical SE scenario. This simulation served as a justification of the need to further perform a simulation of the real research scenario. The simulations based upon the research paper therefore followed after this initial simulation. The aim of simulating the research scenario with artificial data was to partly achieve robustness of the created models, ensuring they performed as expected. Partly, it also provided practical insights into including collider variables, where characteristics of data in a real world setting was captured whilst the amount of noise could be controlled. All simulations conducted involved specifying statistical models including the variables present in the DAG for the corresponding scenario, and fitting the models with the use of simulated data.

The simulations were as stated independent. However, the methodologies utilized in the simulations were very alike. Therefore, it will be collectively described in the upcoming sections. Possible differences in approaches will be clarified. Figure 4.4 provides an overview of the process followed in the simulations. The following sections will include a more detailed description of the methodology conducted in the steps. Section 4.3.1 elaborates on the data simulation process. Section 4.3.2 further describes how the statistical models were created, verified, and fitted.

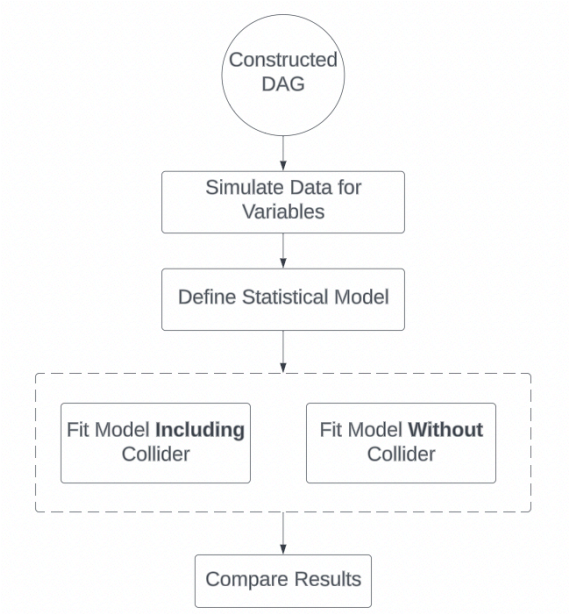


Figure 4.4: An overview of the process of creating the statistical models for the studied scenarios. This includes both a theoretical scenario, and the real research scenario picked. The result from this process was robust models performing as expected, into which real data from the research paper subsequently could be provided.

4.3.1 Simulation of Data

The first step required to conduct each simulation, was to generate synthetic data. In other words, artificial data was created which was supposed to mirror main characteristics of real data. The package utilized for simulating data was the Simstudy package. This package allowed the user to define what variables the data set should include, along with their respective distributions and relationships to other variables [22]. Data could further be generated based on these definitions. Initially, the Simcausal [69] package was also considered to use for simulating data. However, due to the lack of documentation and support for necessary distributions, the package was disregarded. Figure 4.5 provides an illustrative example of how variables could be defined using the Simstudy package [22].

In all simulations carried out, the first step involved the identification of a suitable distribution for each variable, i.e., choosing a distribution that captured the main behaviors of the variable in question. Subsequently, the values for the parameters present in the distributions for each variables were determined. Various expertise was applied in order to determine the parameter values. The sources of expertise utilized varied between the simulations of the theoretical example and the real scenario. These differences will be further elaborated below.

Given a distribution for a variable, and its desired parameter values, the next step included defining it in Simstudy along with the variables influencing it. In addition to the name and distribution of the variable, this included specifying a formula, and either a link function or a variance. The formula could either be an R expression

varname	formula	variance	dist	link
age	10	2	normal	identity
female	-2 + age * 0.1	0	binary	logit
visits	1.5 - 0.2 * age + 0.5 * female	0	poisson	log

Figure 4.5: Exemplification of how variables could be defined using the Simstudy package [22]. In this example, the generated data would contain the three variables age, female and visits. As can be studied also, the variable age influences the variables female and visits. In addition to visits, the female variable also influences the visits variable.

or a simple value, and specified one of the parameters in the distribution. Most commonly, it is the mean the formula specifies [22], but in other cases it could for example be a probability when the variable in question is modelled using the Binomial distribution. R expressions were utilized in scenarios when other variables had an effect on the variable whose formula was to be specified. A variable was required to be defined before it could be put as an influencing variable for another variable. The link function in the package served as a connection between the formula and the parameter value, see further explanation in Section 2.4. A link function was used depending on the distribution, and only if other influencing variables were present. Either log or logit was defined as link functions. The log scale was utilized when the variable was constrained to being positive, whilst the logit scale was used when the variable was required to be between zero and one. However, as indicated, when no variable had an influence on the variable, no link was utilized despite these constraints. This due to the constraint would have be kept regardless given the distribution only. Using these scales allowed for maintaining constraints of a variable, ensuring a more robust data generation. Lastly, if the variance was a parameter of the distribution of the variable, the desired number was simply added to the corresponding field in the variable definition.

The following step after stating desired values for each variable, was to conclude whether or not they should be specified using a link function. This was made using the conditions mentioned above. In these cases, it was required to transform the initially stated parameter value, for the parameter defined by the formula, to the correct scale. To examplify, if the formula corresponded to the probability of success for a binary variable, and the variable had other variables influencing it, a logit link function would have been utilized. If the probability had been defined to 50% to simulate desired data, the formula would have needed to capture the value 0. This, since the inverse logit of 0 is 0.5.

After deciding what variables should have a link function, and transforming necessary parameter values, the following step included defining the formulas for each parameter of each variable. The goal was to approximately achieve the previously defined values, in order to simulate data with the desired characteristics. Several

aspects were taken into account when specifying the a formula for a parameter. The first step involved concluding what variables that had been identified to influence the variable in question, and between what ranges all those variables varied. It was also stated how each variable influenced the variable to specify. That is, whether the influence should be positive or negative. In some cases, influencing variables were logarithmized. This occurred when an influencing variable was simulated with very large numbers, and the variable to be specified was formulated on the log scale or logit scale. With these scales, the interval between which the formula could vary was much smaller. Therefore, the formulas in these cases required smaller values in order to obtain reasonable ranges for the variable to specify, which was the reason for the logarithmization.

In the second step, an intercept in the formula was stated for the variable. This included setting a number showing the expected value of the variable without impact of the influencing variables. This was set near the desired value for the parameter, but with room for change since other variables would either increase or decrease this value with its effect. Setting the effect sizes for all influencing variables was the third step. The effect sizes presents to what extent an influencing variable affects the one to be defined. Common ranges of effect sizes found in research were taken into consideration when setting them for the influencing variables. The effect sizes set for the treatment variable, the outcome variable, and collider variables were set to be in the higher intervals. This to ensure that no effects were washed out due to them being too low. That is, large enough effect sizes enabled the analysis of the effects collider variables.

In addition, effect sizes were also set in consideration with the ranges the final simulated data for each variable were supposed to vary within. That is, the final formula needed so sum up to the parameter value defined in a previous step. The second and third step were partly overlapping, and conducted in parallel. This, in order to achieve reasonable values for the intercept and effect sizes, while in the end obtaining the desired final value for the parameters which would generate the wanted characteristics in the simulated data. The formula needed to approximately obtain the value previously stated on the stated scale. What also was taken into consideration

The final entry in the definition of data was the variance. This field was utilized when variance was a parameter present i a variables' distribution. The approach utilized for this differed between the simulations, and will therefore also be described below. In all simulations, the data was continuously generated during the data simulation process, in order to ensure it turned out as expected. Modifications were otherwise made in the formulas until the data generated was satisfactory. In the upcoming paragraphs, descriptions of the differences in approaches utilized in the simulations of the theoretical example, and real research scenario will be presented.

Theoretical Example

The first dataset generated was the one for the theoretical example. In this case, the entire example was, as the name suggests, theoretical. Therefore, the sole expertise used to build this simulation, aiming to provide as a realistic SE example as possible,

was domain knowledge. That is, commonly known knowledge within the field of SE was utilized in order to set reasonable parameter values for a given context.

Real Research Scenario
For the scenario found in the research paper, two datasets were generated. This due to both the total, and direct effect was of interest to examine. One dataset was a subset of the other: the dataset constructed for the direct effect comprised the complete set containing all variables utilized, whereas the other dataset contained the total effect and therefore included only the collider variables, the treatment variable and outcome variable. The aim with the simulated data for the research scenario was to capture general characteristics of the real dataset utilized in the paper. This firstly included finding a suitable distribution for the simulated variables. The distributions in this scenario were set based on investigations of both descriptions made about each variable in the text, and the type of values each variable included in the real data. The distributions picked were supposed to maintain the characteristics of the variable. For example, if the variable would represent a count of some kind, a distribution would have been required to pick which not contains negative numbers or decimals. In order to set the parameter values for each distribution, several examinations were made in the real data. The main focus when capturing real variable characteristics was for each simulated variable to have approximately the same mean as the real variable. Therefore, the mean of each real variable in the original dataset from the paper was obtained. In addition, the minimum and maximum value for each variable was also examined, including noticing whether or not negative numbers were present for a variable. This allowed for learning about the spread of the variables. For the variables which where simulated using a distribution where a variance could be set, these metrics functioned as a guide when setting these values. The goal, however, was not to capture the entire range in which the variables varied. During the process of setting parameter values based on these investigations, continuous checks were made in order to ensure that the data generated had the expected characteristics. These checks consisted of plots of the chosen distributions with the decided parameter values.

4.3.2 Model Creation and Fitting

The creation of statistical models was done using the brms package in R. The rethinking package was also investigated, the brms package did however provide more convenient functionality for posterior predictive checks. That is, in this scenario, functionality for investigating practical implications of the collider variables on the outcome variable. Two models were defined for the simulation of the theoretical example, one including the collider variable and one excluding the collider. In the simulations of the real example, both the direct effect and total effect was of interest. Therefore, four models were in total defined for the simulation of the research scenario, due to the inclusion and exclusion of colliders in both the investigation of direct effect and total effect. The first step when designing a model was to define the outcome variable of interest, along with an appropriate likelihood. The choice of likelihood was made based on which distribution with biggest information entropy.

The next step included formulating the linear model, connecting a parameter of the outcome to the predictor variables. Depending on the likelihood of the outcome variable, either a regular linear regression model or a GLM was utilized. A GLM was used in the scenarios where the likelihood was distinct from the Normal distribution. In these cases, the outcome variable was constrained in some way, which required a link function inserted around the parameter connected to the linear model. The log link function was utilized when the outcome variable was discrete, whilst the logit link was used when the outcome variable was bounded to a certain range.

The linear model was defined using partly an intercept, which represent the expected value of the outcome variable without influence of any other variables. Partly, the respective variables present in the simulation scenario were entered into the model together with a respective β-parameter for each variable. These β-parameters represent each of the the variables impact on the outcome variable. Subsequently, priors were set for each parameter in the linear model, along with potential parameters present in the likelihood not connected to the linear model. Priors were defined with the use of domain knowledge, and prior predictive checks. The prior predictive checks functioned as a sensitivity analysis. The prior predictive checks were especially important when GLMs were used. This, due to the link function making transformations between different scales of the likelihood parameter and outcome variable, aggravating the interpretation of the priors. The more parameters that were part of the model, the more careful it was required to be in terms of setting appropriate priors. By having many parameters, each with its own prior, there could otherwise have been an explosion of variance to handle for the model.

Different sample sizes were investigated during the fitting of the models. As specified by [40], providing more data to the model helps, the Bayesian estimates however remains valid for any sample size. For the varying sample sizes, 1000, 5000, 8000, and 10000 were utilized in different runs of fitting the models. After running a model, it was verified that the models were healthy and that the chains were converging. This was done through several diagnostic criteria; Rhat ($\hat{R}$), n_{eff}, and traceplots. These diagnostics were examined throughout the entire process of fitting the models. As presented by [40], a Rhat value above 1.00 indicates that the chains have not converged yet and the samples should not be trusted. The effective number of samples, n_{eff}, should not be much lower than the than the number of iterations (excluding warmup) of the chains. If that is the case, the chains are most likely insufficient. Lastly, the traceplot should illustrate zig-zagging traces were they rapidly explore the full region, but also sticks around the same center. Whenever there were any signs of unhealthy models or divergence, the models were investigated, including the priors. Additionally, the number of iterations was increased. If the chains still had problems of converging, standardization of variables were considered as a solution to help for the sampler to run.

When the models were running correctly, the last step was to analyse the final results. In particular, this included comparing the beta estimates of the treatment on the outcome variable. In the simulation of data, this effect was set to a true effect, that was aimed to retrieve. A comparison was made between the models and this effect, to assess how accurately they reflected the value. This comparison applied to all

models, both those including colliders and those without, for both the direct and total effects. Additionally, another method of investigating the impact of colliders on the outcome variable involved analysing practical implications though posterior predictive checks. This would allow for a comparison between the predictions of the fitted model and the actual data, as well as showing any differences in outcomes for the model that included a collider and the one without.

4.4 Analysis Using Real Data from Subject Paper

When the simulations of the research scenario had been conducted, the subsequent step included an analysis using the real data provided in the replication package of the paper. The methodology utilized was similar to the one used for the simulations, with the exception that no data was simulated. Statistical models were designed, where the ones defined in the simulations were used as foundation and further modified. Specifically, the priors set for the parameter values were changed. This due to that data obtained from reality can be less clean than simulated data. Therefore, tighter priors were set for the models managing real data, in order to facilitate for the sampler. When running the models, the same amount of samples was utilized as in the simulations. These samples were obtained randomly from the dataset.

5

Results

This chapter details the results obtained from the study, divided into four parts. Section 5.1 presents the results retrieved from the sampling of papers at the MSR conference. Sections 5.2–5.5 describe the construction of DAGs. This was done for one theoretical SE scenario, and three papers sampled from the MSR conference. Section 5.6 presents the result from the computer simulation, starting with the theoretical scenario and followed by a simulation of the real research case in the paper Co-occurrence of Refactoring. Finally, Section 5.7 details the results from utilizing the real data from the same paper, Co-occurrence of Refactoring.

5.1 Data Sampling from MSR Conference

This section presents the results obtained from the selection of suitable papers published at the MSR conference. Two different tracks from the conference were specifically utilized for the sampling: the DPA[1] track and the MC[2] track. The results from the sampling of these two tracks are presented separately. First, Section 5.1.1 details the results from the DPA track, and Section 5.1.2 covers the MC track. More detailed results from the sampling, including a color-coded spreadsheet, can be found in Appendix A.

5.1.1 Distinguished Paper Awards

Sampling Iteration 1
Papers awarded the DPA between the years of 2018–2023 were considered in the first sampling iteration. The results from this iteration are demonstrated in Figure 5.1. 3 papers out of 17 in total considered fulfilled all criteria, as can be studied in the figure. The paper chosen for the subsequent step of constructing DAGs, from the three suitable papers, was Prevalence of Confusing Code in Software Projects - Atoms of Confusion in the Wild [23]. A description of the research presented in the paper is briefly summarised in Section 5.3. This paper was chosen since it contained several statistical tests including a large number of variables, which increased the probability of encountering colliders in their statistical analyses. The remaining two suitable papers did not contain as many variables and were not considered as promising, so they were not investigated further.

[1]ACM SIGSOFT Distinguished Paper Awards
[2]Mining Challenge

Figure 5.1: Result from the first filtering, where papers awarded the DPA between 2018–2023 were investigated. Each row contains the number of papers that fulfilled the criterion on the given row, and therefore also all rows above.

No colliders were possible to identify in the chosen paper from this iteration. The sampling process therefore continued with an extension in regards to constraints stated before sampling papers. The time span of when papers had received the DPA was expanded to the years between 2015–2017. The result from this sampling is presented in the next paragraph. The MC track was additionally added to consider when sampling papers, which is described in more detail in Section 5.1.2.

Sampling Iteration 2

Figure 5.2 outlines the results from the second iteration of sampling papers, where papers awarded the DPA between the years 2015–2017 were considered. 6 papers were sampled from, and 1–2 suitable papers were identified. There were some challenges in concluding the precise number of variables utilized in one of the research papers, which resulted in the uncertainty presented in the last row of the figure. Further investigation would have been required to determine this. However, since a suitable paper had already been found during the MC sampling described in Section 5.1.2, this sampling iteration was aborted.

Figure 5.2: Result from the extended filtering, where papers awarded the DPA between 2015–2017 were investigated. Each row contains the number of papers that fulfilled the criterion on the given row, and therefore also all rows above

DAGs

A causal perspective and the use of DAGs was a missing element in the DPA track, which can be seen in Figure 5.1 and 5.2. No causal DAG was found in the 23 papers examined during the two sampling iterations. One non-causal DAG was identified, and a few papers brought up aspects of causality. However, when such aspects were mentioned, they typically included statements about the inability to draw causal conclusions based on their research, or that causality was not taken into account. One paper examined included more elaborations in regards to causality [26]. The authors of the paper state challenges of producing empirical insights with the use of observational data found in repositories, especially pointing out causation to be complex whilst crucial for science. The research further aimed at utilizing simulation-based testing on a set of research cases as a means to navigate the potential threats to validity the stated challenges might have on research. No causal perspective on the authors' own conducted research was however applied.

Reproducability and Transparency

Table 5.1 illustrates the amount of papers, per iteration, which provided a replication package. Two additional papers did in the first iteration provide their dataset used in their paper. However, as described in Section 4.1.1, these papers were filtered out due to time constraints. Twelve papers were in total replicable in the DPA track, meaning that there was the possibility of reproducing the results of the authors. Table 5.1 additionally highlights a positive trend: a higher proportion of papers have included replication packages in more recent years.

Table 5.1: Number of papers providing a replication package. Each row contains the total number of papers investigated in each iteration, and the number of those papers which had a replication package. The final column presents the proportion of papers enabling others to replicate their work. Replication package is shortened as RP in the table.

Iteration	Total Papers	# of Papers with RP	Percentage
Iteration 1: 2018 – 2023	17	10	58.8%
Iteration 2: 2015 – 2017	6	2	33.3%

The result obtained from the collection data concerning selection criteria information presented in the papers can be found in Table 5.2. This result includes data from the 12 papers between the years of 2015–2023 which passed the phase 1 of the sampling. A challenging aspect encountered during the paper selection process was the identification of variables. This was due to the lack of clear elaborations on what variables that had been utilized in the research papers examined. Getting a good enough understanding of the specific underlying variables utilized was a highly time-consuming process, and indicated a low level of transparency amongst the papers.

Table 5.2: Presentation of the amount of papers including explanations regarding how their data and/or variables utilized in their research were collected and selected. The total number of papers considered are the ones that would have been possible to replicate. Replication package is shortened as RP in the table.

Informational Check	# of Papers Presenting the Information	Percentage
Papers' Data Selection	11	91.7%
Papers' Variable Selection	2	16.7%
Total # of Papers Containing RP: 12		

5.1.2 Mining Challenge

Sampling Iteration 1

The second track examined in addition to the DPA track, was the MC at the MSR conference. One sampling iteration was carried out in this track, and the years of 2022, and 2023 were investigated. The result from this sampling can be studied in Figure 5.3. Consequently, four papers out of the twelve papers considered fulfilled every criterion. Initially, the paper Between JIRA and GitHub: ASFBot and its Influence on Human Comments in Issue Tracker [43] was selected for the subsequent step of constructing DAGs. An overview of the research conducted in that paper can be found in Section 5.4. As no collider was found in this paper, another paper was chosen for DAG construction: On the Co-Occurrence of Refactoring of Test and Source Code [46]. The results of this analysis can be found in Section 5.5. Eventually, a collider was found and the sampling process was therefore ended.

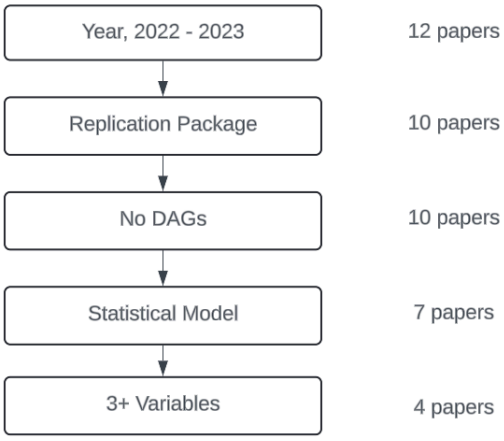


Figure 5.3: Result from the filtering of papers from the MC $2022 - 2023$. Each row contains the number of papers that fulfilled the criterion on the given row, and therefore also all rows above.

DAGs

In alignment with sampling from the DPA track, no DAGs or causal perspectives were included in the papers from the MC track. In one paper examined, the authors discussed the limitation of their results being potentially affected by selection bias. However, they did not elaborate further on the implications for causality.

Reproducability and Transparency

Table 5.3 presents the number of papers that provided a replication package in the MC track. As can be studied, ten papers out of twelve contained a replication package.

Table 5.3: Number of papers providing a replication package. The row contains the total number of papers investigated, and the number of those papers that had a replication package. The final column presents the proportion of papers enabling others to replicate their work. Replication package is shortened as RP in the table.

Iteration	Total Papers	# of Papers with RP	Percentage
Iteration 1: 2022–2023	12	10	83.3%

Determining variables used in the research papers in this track was a smooth process. As illustrated in Table 5.4, 90% of the papers provided clear descriptions and reasoning behind the choice of variables. Some papers even included a table demonstrating each variable included in their analysis and descriptions of them. The entry for selection criteria of data in the table is marked as NA for this track because the competition provided the dataset for the participants. This eliminated the need for researchers to justify the data source.

Table 5.4: Presentation of the amount of papers including explanations regarding how the variables utilized in their research were collected and selected. The total number of papers considered are the ones that would have been possible to replicate. Replication package is shortened as RP in the table.

Informational Check	**# of Papers Presenting the Information**	**Percentage**
Papers' Data Selection	NA	NA
Papers' Variable Selection	9	90.0%
Total # of Papers Containing RP: 10		

5.2 DAG Construction: Theoretical Scenario

The constructed SE example presents a scenario where one would be interested in investigating the effect a developer's experience has on the number of lines of code the developer writes in a day. To examine this scenario, the number of bugs the developer inserts in the code in a day was added as a predictor. Three variables were therefore included in this simulation:

- DE: Developer experience in years
- LOC: Lines of code written by a developer in a day
- B: Number of bugs inserted by a developer in a day

The relationship between the three variables can be studied in the DAG in Figure 5.4. In this scenario, the variable number of bugs, B, is a collider. What follows are the rationals used to draw each arrow in the DAG:

- **DE** $\rightarrow$ **LOC**: The longer a developer has been coding, the more likely it is that they can produce more concise and efficiently written code. That is, they would probably be able to express the same functionality in fewer LOC than a less experienced developer.
- **DE** $\rightarrow$ **B**: The number of bugs inserted in the code by a developer is likely to decrease with increased experience since fewer mistakes are made with more experience.
- **LOC** $\rightarrow$ **B**: As the LOC increases, there is more opportunity for bugs to be introduced. Therefore, it is likely that the number of bugs is influenced by the amount of LOC

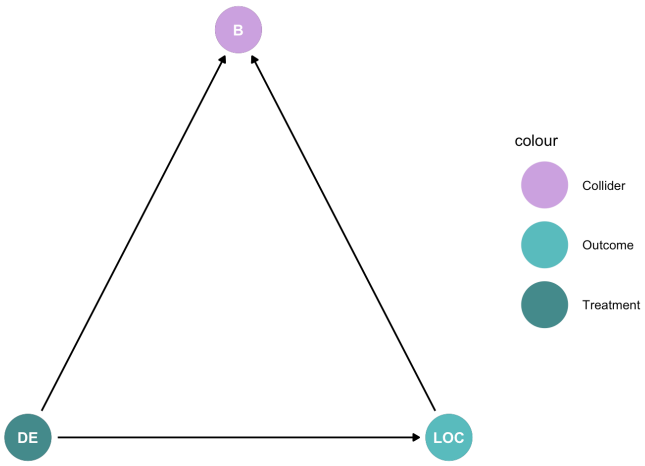


Figure 5.4: DAG illustrating a potential and simple research scenario in SE.

5.3 DAG Construction: Atoms of Confusion

The paper titled Prevalence of Confusing Code in Software Projects: Atoms of Confusion in the Wild by Gopstein et al. [23] received the DPA in 2018. The study aims to illustrate the relationship between small code patterns, referred to as atoms, and the occurrence of buggy code. The authors hypothesize that these atoms can cause misunderstandings amongst programmers, subsequently leading to bugs. They analyzed 14 prominent open-source C and C++ projects in their research, including well-known projects like Linux and Git. These projects were categorized into specific application domains such as operating systems and version control, with two representative projects selected for each domain. This was done to make comparisons by application type. See the projects and domains in Table 5.5. The study investigated 15 distinct types of atoms previously identified in the research, aiming to demonstrate the prevalence, impact, and confusion associated with each small coding pattern. Key research questions, Q[3], addressed in the paper included:

- **Q1:** How often are atoms used in real software?
- **Q2:** Do atoms occur with different frequency in different projects?
- **Q3:** Does project age influence the rate of atoms?
- **Q4:** Are atoms removed more often in bug-fix commits?
- **Q5:** Do projects with more atoms also have more bugs?
- **Q6:** Does prevalence correlate with amount of confusion?
- **Q7:** Are atoms commented more often than other code?

The researchers measured the usage and prevalence of each atom across selected projects. They then correlated these occurrences with external factors like bugs and

[3]The acronym Q is used to avoid confusion with the research questions (RQs) of this study

Table 5.5: Projects analyzed in Atoms of Confusion

Project	Domain
Linux	Operating System
FreeBSD	Operating System
Gecko	Browser Renderer
WebKit	Browser Renderer
GCC	Compiler Suite
Clang	Compiler Suite
MongoDB	Database
MySQL	Database
Subversion	Version Control
Git	Version Control
Emacs	Text Editor
Vim	Text Editor
Httpd	Webserver
Nginx	Webserver

comment rates using statistical tests. The findings reveal that atoms are prevalent in real projects and are meaningful in software development.

Selection of Research Questions for DAG Construction
Out of seven research questions from the original paper, four (Q3, Q4, Q5, and Q6) were in-depth analyzed in this study, and further used to build DAGs upon. The first two RQs (Q1, and Q2), were excluded from the start since they did not contain any statistical model, and were therefore not of interest. The remaining question (Q7) was investigated but not used for DAG construction, even if it included a statistical model and a sufficient number of variables. The decision to reject the last research question (Q7), was prompted by the unsuccessful attempts of finding colliders in the first six research questions. Therefore, it was decided to prioritize other research papers, with a possibly higher chance of identifying colliders.

The analyses and attempts to construct DAGs for the four research questions mentioned can be found in Sections 5.3.1- 5.3.4. Each section provides a summary of the paper's statistical analysis, followed by a description of the statistical model and identified variables. A complete DAG, or in some cases an attempt towards a complete DAG, is further presented in a figure, accompanied by explanations for each arrow drawn in the DAG. Section 5.3.5 presents the overall results from the DAG construction for the paper Atoms of Confusion.

5.3.1 DAG outlining Q3: Project age influence on rate of atoms

The research question Q3 was the first question approached for the DAG construction, and was formulated as: "Does project age influence the rate of atoms?". Gop-

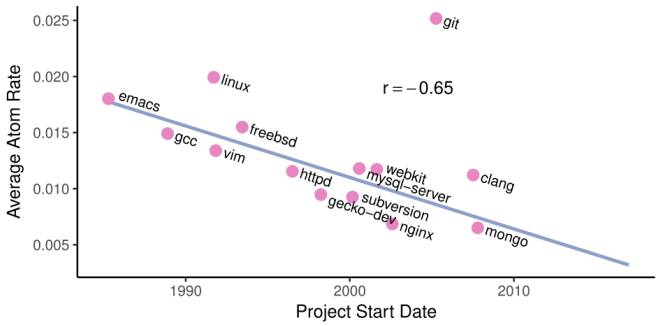


Figure 5.5: The average rate of atoms in each codebase, over the lifetime of each project, ordered by the year in which the project was created. Illustration from [23].

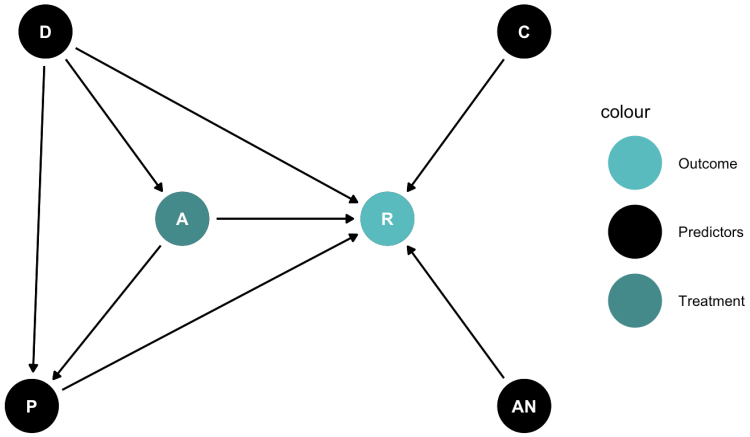


Figure 5.6: DAG showcasing the system under study in Atoms of Confusion Q3. P = project ID, D = domain, A = date, R = rate, c = count, AN = all.nodes.

stein et al. [23] describe a dramatic change in the software industry over the past 30 years and aim to understand if the atom rate differs in newer versus older projects. When conducting the statistical analysis, they used the initial commit date of a project to compare against the average atom rate over the project's lifetime. This was plotted in the paper and can be seen in Figure 5.6. From the results, the authors concluded that newer projects use fewer atoms than older projects.

In the file [4] found in the repository provided in the paper, a Robust Linear Model (RLM) was identified to have been used in this research question (Q3). The variables identified to have been provided to the model were: project ID, domain, date, rate, count, and all.nodes. The identified variables can be found with shortenings and descriptions in Table 5.6. The date variable was stated as the treatment variable and the rate variable as the outcome, given the question under study and the identified variables. The DAG constructed showcasing the scenario in Q3 can be studied in Figure 5.6. The motivations behind each arrow drawn are specified in Appendix B.1. No collider variables were possible to identify in this DAG.

[4] `https://github.com/dgopstein/atom-finder/blob/master/src/analysis/code_age.R`

Table 5.6: Variables identified from Q3 in replication package.

Shortening	Feature	Description
P	project ID	One out of the 14 observed projects.
D	domain	One out of the seven domains.
A	date	The project's initial commit date in format YYYY-MM-DD. Referring to as project again text.
R	rate	A derived variable from the computation count / all.nodes, representing the average atom rate.
C	count	Number of abstract syntax tree (AST) nodes containing an atom
AN	all.nodes	The total amount of AST nodes.

5.3.2 DAG outlining Q4: Commit type influence on atom removal rate

The next research question, Q4, seeks to answer: "Are atoms removed more often in bug-fix commits?". Gopstein et al. [23] investigated the relation between atoms and bugs by comparing the rate at which atoms are removed in bug-fixing commits versus non-bug-fixing commits. The GCC project was selected as the data source due to its size, history, and bug tracking. In conclusion, the authors could show that atoms are 1.25x as likely to be removed in bug-fix than non-bug-fix commits. Figure 5.7 shows the authors found relative likelihood for each atom type to be removed.

By studying the file [5] in the repository, two statistical tests could be identified: a χ^2-test (statistical model) and a Fisher Exact test. The variables from these tests can be found in Table 5.7. It can be observed that some variables were derived from other identified variables.

The main objective in Q4 appeared to be investigating whether the type of commit (bug-fix or non-bug-fix) influenced the atom removal rate. The construction of a DAG for this research scenario was challenging. Many variables were utilized and, in addition, intricately defined which hindered the DAG construction. How the authors approached Q4 with their statistical analysis was not clear enough to smoothly construct a DAG. Figure 5.8 demonstrates a conceptual framework, containing the relationships between the variables utilized in the authors' statistical analysis. This was created to understand what variables were used where in the computation of atom removal rate (the outcome of interest). A DAG was attempted to be constructed, which can be studied in Figure 5.9. The reasoning behind each arrow is

[5] `https://github.com/dgopstein/atom-finder/blob/master/src/analysis/gcc_bug_counts.R`

Table 5.7: Variables identified from Q4 in replication package.

Short	Feature	Description
A	atom ID	Each of the 15 observed atom types.
IB	is.bug	Showing if the commit resolved a bug or not (binary value). Derived from is.bug = n.bugs > 0.
B	n.bugs	Number of bugs resolved in commit (integer).
C	source.change	Represents the difference in the number of characters between the source code after (CA) and the source code before the commit (CB). Calculated as *source.change := source.chars.after - source.chars.before*
CB	source.chars.before	The total number of characters in the source code before a commit (integer).
CA	source.chars.after	The total number of characters in the source code after a commit (integer).
I	increased	The number of atoms that have increased in quantity after a commit is made. Computed as *increased = sum(count.after > count.before)*
D	decreased	The number of atoms that have decreased in quantity after a commit is made Computed as *decreased = sum(count.after < count.before).*
CB	count.before	The number of atoms before a commit, represented as an integer.
CA	count.after	The number of atoms after a commit, represented as an integer.
AR	atom.removal	The atom removal rate, based on the ratios; TT, TF, FT and FF.
TT	TT	True positive rate, true-true, represent the ratio of where a bug (is.bug = 1) is identified and the number of atoms decreased after the commit is done. Computed as *TT = sum(is.bug*decreased/source.change).*
TF	TF	True negative rate, true-false, represent the ratio of where a bug (is.bug = 1) is identified and the number of atoms increased after the commit is done. Computed as *TF = sum(is.bug*increased/source.change).*
FT	FT	False positive rate, false-true, represent the ratio of where a bug (is.bug = 0) is not identified and the number of atoms decreased after the commit is done. Computed as *FT = sum((1-is.bug)*decreased/source.change).*
FF	FF	False negative rate, false-false, represent the ratio of where a bug (is.bug = 0) is not identified and the number of atoms increased after the commit is done. Computed as *FF = sum((1-is.bug)*increased/source.change)).*

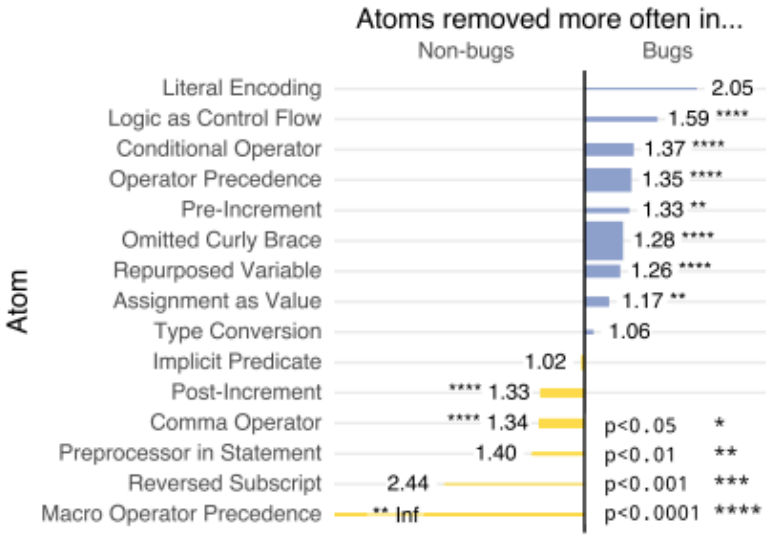


Figure 5.7: Rate at which atoms are removed in bug-fix and non-bug-fix commits. Illustration from [23].

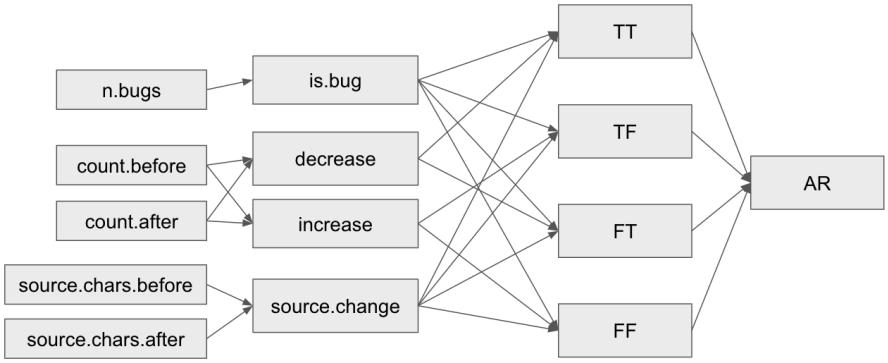


Figure 5.8: A conceptual framework for the derived variables in Q4.

detailed in Appendix B.2.

Once again, no colliders were possible to identify. However, the DAG cannot be claimed to be completed in this scenario. The aim and methodology of the statistical analysis were difficult to comprehend fully. It was particularly challenging to understand the selection and utilization of variables and concretely identify the authors' underlying assumptions.

5.3.3 DAG outlining Q5: Atoms influence on bugs

The subsequent research question, Q5, the authors stated was: "Do projects with more atoms also have more bugs?". They wanted to investigate the relationship between atoms and bugs on a higher, project level. This was done by conducting two statistical analyses: one including Common Vulnerabilities and Exposures (CVE) data and the other with bug data from each projects respective bug tracking repos-

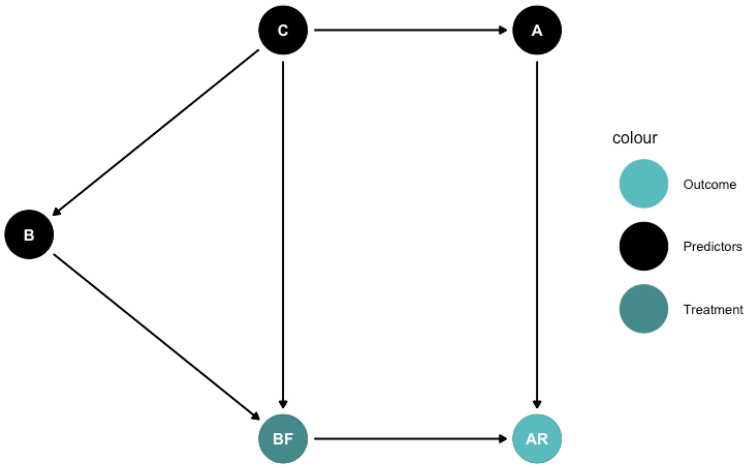


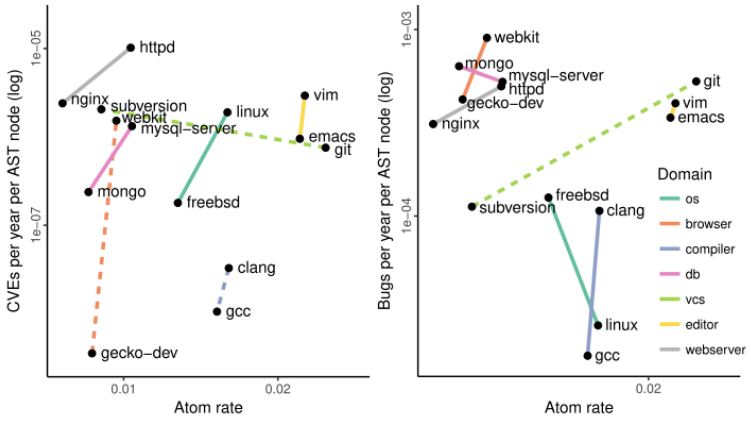
Figure 5.9: An incomplete DAG of the system under study in Atoms of Confusion Q4. BF = bug-fix commit (TT + TF), AR = atom.removal, B = n.bugs, C= source.change, A = change in atoms (increase/decrease).

itory. They paired the data by domain and analyzed the results based on those relationships. The final results from both tests can be found in Figure 5.10. From this result, the authors could conclude that the project with more atoms tended to have more CVEs and bugs by domain. However, they added that there is some uncertainty about the results due to the small sample size and questionable data quality.

The analysis for these two tests was found in separate files, one for bugs [6] and one for CVEs [7]. Both tests were conducted with similar characteristics: performed with a Linear Model (LM) and with many common variables. This was why only one DAG was constructed for this research question, where part (b) with reported bugs was selected. This selection was made due to the closer connection to the authors' research question, where the relation between atoms and bugs was stated as the relation of interest. Table 5.8 presents all variables for the selected statistical model. Figure 5.11 demonstrates the DAG constructed from these variables. Motivations of the arrows drawn are stated in Appendix B.3. As can be seen in the DAG, no colliders were found in this DAG either.

[6] https://github.com/dgopstein/atom-finder/blob/master/src/analysis/project-bugs.R

[7] https://github.com/dgopstein/atom-finder/blob/master/src/analysis/project-cves.R



(a) Project CVEs **(b) Project Bugs**

Figure 5.10: Plot visualizing projects' atom rate compared to defects normalized by time and project size. Projects are connected by domain, and some lines are dotted to indicate questionable data. Illustration from [23].

Table 5.8: Variables identified from Q5 (b) in replication package.

Shortening	**Feature**	**Description**
P	project ID	Each of the 14 observed projects.
D	domain	Each of the seven domains.
A.R	atom.rate	The atom rate in a project X. Computes as *atom.rate:=(X.all.nodes-X.non.atoms)/ X.all.nodes*.
A.N	all.nodes	The total amount of AST nodes in a project.
N.A	non.atoms	Number of AST nodes that does not contain an atom.
A	atom ID	Each of the 15 observed atom types as separate features.
B.C	bug.count	Number of bugs for each project. Manually added through *bug.count <- c(30930,...,1723)*.
S	since	Date of the first data point for each project. Manually added through *since <- as.Date(c('2002-11-06',...,'2011-05-15')*
B.R	bug.rate	The bug rate for each project, from initial data point until 2018-01-01. Computed as *bug.rate := bug.count/ as.numeric(as.Date("2018-01-01")-since)*.

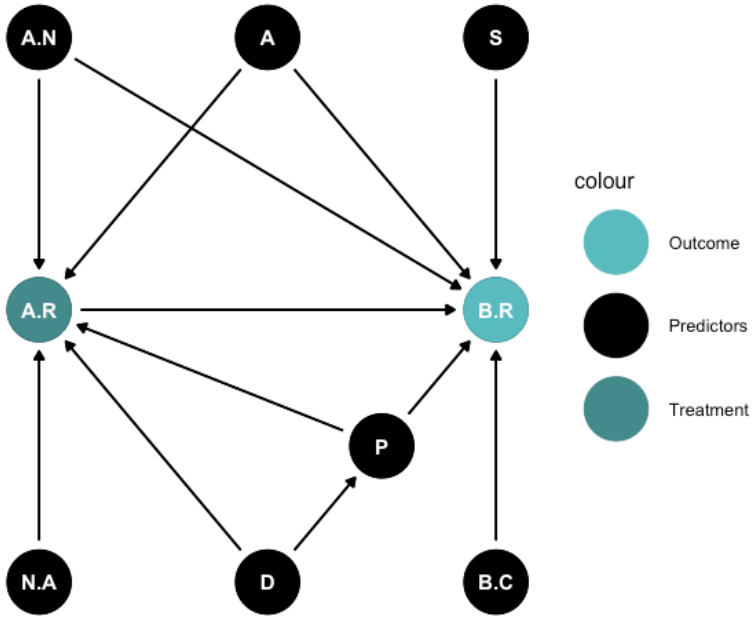


Figure 5.11: A DAG showcasing the system under study in Atoms of Confusion Q5. A.R = atom.rate, A.N = all.nodes, N.A = non.atoms, D = domain, B.R = bug.rate, B.C = bug.count, S = since, P = project ID, A = atom ID

5.3.4 DAG outlining Q6: Atom rate influence on level of confusion

Research question Q6 aimed to investigate: "Does prevalence correlate with amount of confusion?". Gopstein et al. [23] described that several studies have shown a relationship between how frequently code occurs and how well idioms are understood by the developer, which was further studied in this research question. Figure 5.12 shows a plot of the resulting relationship identified. What can be studied in the figure is that a strong, negative, logarithmic correlation ($r = -0.45$) was found. This result showed that developers more frequently use less confusing atoms. The authors additionally stated the following: *"We cannot determine whether the relationship between confusion and prevalence is causal, and if so, in what way."*. This was the single research question where the authors explicitly addressed causality.

The analysis of Q6 can be studied in the replication package [8], where the correlation is calculated between two variables of interest. Five variables were utilized to conduct the analysis. These can be studied in Table 5.9. Given the question under study and the identified variables, all.atom.rates were stated as the treatment variable and effect.size as the outcome variable. The DAG constructed for Q6 can be seen in Figure 5.13. Motivations of the arrows drawn are stated in Appendix B.4. As can be seen in the DAG, no colliders were found in this DAG.

[8] `https://github.com/dgopstein/atom-finder/blob/master/src/analysis/atom_counts.R`

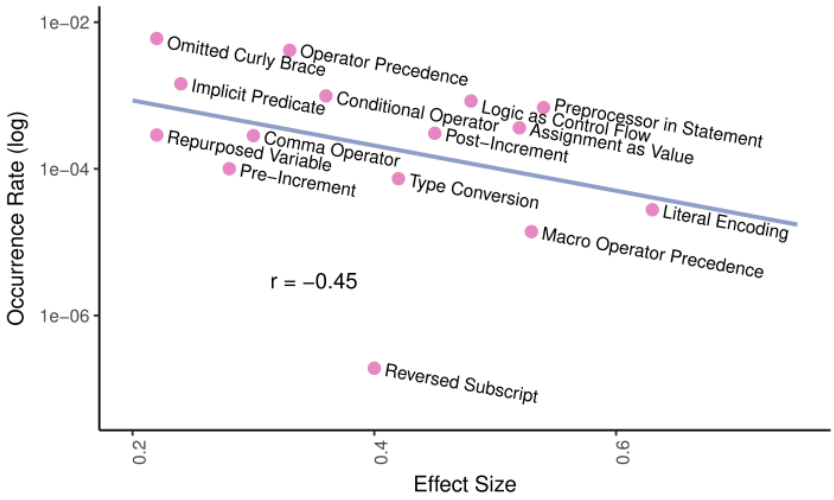


Figure 5.12: Comparing confusingness with prevalence. Illustration from [23].

Table 5.9: Variables identified from Q6 in replication package.

Shortening	**Feature**	**Description**
A.R	all.atom.rates	The rate which an atom occur. Derived from *all.nodes* and *non.atoms*.
A.N	all.nodes	The total amount of AST nodes in a project.
N.A	non.atoms	Number of AST nodes that does not contain an atom.
E.S	effect.size	Effect size indicates the degree to which the atom contributes to confusion compared to a transformed pair, ranging from 0 to 1.
A	atom ID	Each of the 15 observed atom types as separate features.

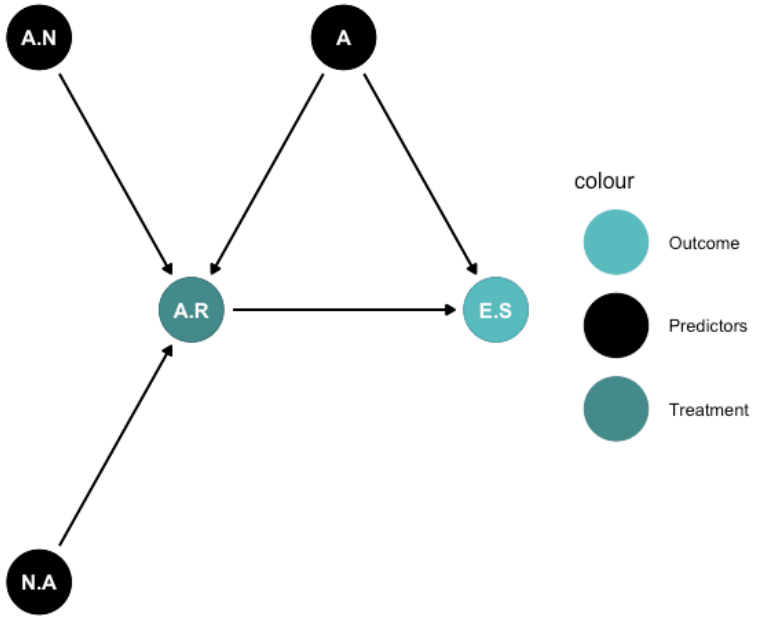


Figure 5.13: A DAG showcasing the system under study in Atoms of Confusion Q6. A.R = all.atom.rates, A.N = all.nodes, N.A = non.atoms, E = effect.size, A = atom ID

5.3.5 Construction Process

This paper posed several challenges during the construction of DAGs based on the stated research questions. The provided replication package contained many and in addition large files. It was also difficult to navigate the repository due to the lack of clear documentation. Additionally, a few comments were inserted inside the code explaining which part of the code did what. The first step when building DAGs for this paper was therefore to connect what files corresponded to what research question.

Another aspect that hindered the construction process was that each file contained analyses and calculations without apparent purpose to the research. This especially complicated the identification process of finding the key variables the authors used when answering each research question. Q5 is an example of this described scenario, which aimed at investigating atoms affect on bugs. However, calculations and analyses were also conducted regarding the relationship between Common Vulnerabilities and Exposures (CVES) and atoms. While this could be seen as a relevant aspect of the examined research question, the authors neither included a discussion concerning this aspect nor a motivation regarding why it was added.

Furthermore, many of the variables identified were also derived variables. Meaning, they were computed from other variables. For example, various rates were used in the research. It was not always clear how these derived variables were computed, which was necessary to understand when creating a visual representation of a research scenario. It was particularly challenging to grasp how the authors defined

the variable Atom Removal Rate. A positive aspect during the construction process was that the authors included elaborations concerning the underlying assumptions of the research context. These assumptions mainly reflected the authors' general perceptions rather than specific reasons for selecting certain variables. Two examples of found assumptions are:

"The software industry has evolved dramatically in the past 30+ years since the oldest project in our corpus was born."

"Atoms of confusion are not a homogeneous group by any definition. Instead, each pattern is unique in its syntax, semantics, and the way it causes misunderstandings in readers."

Collectively this DAG construction process can be described as intricate. However, no colliders were possibly identified in the DAGs created. A new paper was therefore required to examine, which will be described in the upcoming section.

5.4 DAG Construction: ASFBot

The paper Between JIRA and GitHub: ASFBot and its Influence on Human Comments in Issue Trackers, authored by Moharil et al. [43], participated in the 2022 MC. The authors elaborate on the growth of Open-Source Software (OSS) projects, and how such projects in recent years have adopted various automations for managing repetitive tasks. One common type of automation in OSS is bots, where ASFBot by Apache Software Foundation (ASF) is one of them. The study aims to understand how the adoption of the ASFBot influences discussions in issue-tracking systems (ITS). A data set, SmartShark, was given for the MC of 2022, where they in this study used the data to investigate the following two research questions, Qs[9]:

- **Q1:** Does the ASFBot impact the number of comments made by humans which mention Pull Requests (PRs) and fixes across the issuetrackers of Apache projects?
- **Q2:** Does the ASFBot impact the number of comments made by humans in the issue-trackers of Apache projects?

Moharil et al. [43] conducted an exploratory case study, investigating the impact of ASFBot on the issue tracking system (ITS) of nine different Apache projects. These projects were carefully selected, with the criteria of being active at least one year before and after an ASFBot adoption. The final projects were: maven, mina-sshd, santuario-java, commansbcel, jackrabbit, roller, gora, openwebbeans, directory-studio. The authors utilized regression discontinuity design (RDD), a common technique for studying the impact of interventions in SE. The results from Q1 show an instant decrease in the median monthly comments mentioning PRs and fixes after ASFBot adoption. However, the monthly trend of decreasing comments was reversed after adopting the ASFBot. For Q2, no effect was shown regarding the number of human comments after the ASFBot adoption.

[9]The acronym Q is used to avoid confusion with the research questions (RQs) of this study

Table 5.10: Variables identified for ASFBot in paper of Moharil et al. [43].

Shortening	Feature	Description
ASF	ASFBot	A binary value, indicating the adoption of the bot at a certain time [43].
D	Number of Developers	"We compute the number of monthly developers who commented on an issue [43]."
I	Number of Issues	"We collect the number of unique monthly issues [43]."
Y	Project Age	"We compute this as the time in years since the project has been active until the last available comment [43]."
C	Median Number of Comments	"For Q1, this refers to the median number of monthly human comments that contain keywords related to pull requests and fixes. For Q2, comments is the median number of human comments per issue in the ITS [43]."
P	Project Name	Represent each of the nine projects [43].

5.4.1 DAG outlining: The presence of the ASFBot on amount of comments

The two research questions concerning ASFBot are similar, employing the same statistical approach and variables. The distinction lies in the variable Median Number of Comments C: Q1 focused on comments containing keywords related to pull requests while Q2 examined comments per issue in the ITS. The DAG construction for this paper incorporated both research questions. In the paper ASFBot, Moharil et al. [43] use Regression Discontinuity Design (RDD) to conduct their statistical analysis. This analysis can be found in detail in the replication package [10], together with files for bot detection and data cleaning, collection, and exploration. The statistical model behind the RDD follows a linear mixed-effects model along with the longitudinal effects [43]. The variables identified to have been used in this model can be studied in Table 5.10. In addition to these variables, two variables related to time were utilized in the analysis. The DAG constructed can be studied in Figure 5.14. Motivations of the arrows drawn are stated in Appendix B.5. No collider was identified in this paper either. No scenario was identified where an arrow could have been drawn from the comments C variable, which would be required for a collider. Therefore, a new paper was investigated.

[10] `https://figshare.com/s/9e9bcdcab730801dab4b`

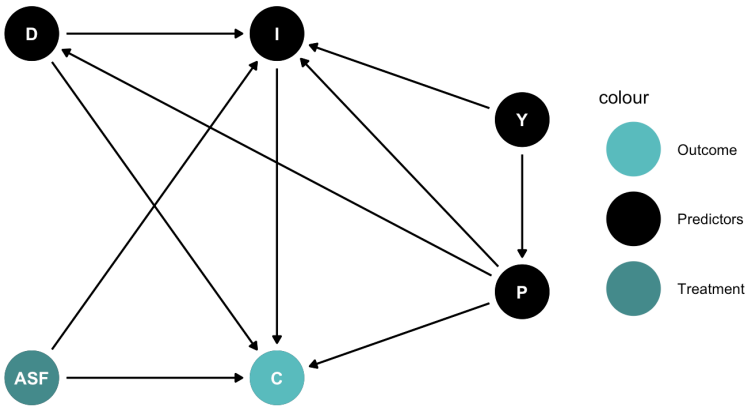


Figure 5.14: DAG showcasing the research scenario including the ASFBot. ASF = ASFBot, D = Numbers of Developers, I = Number of Issues, Y = Project Age, C = Median Number of Comments, P = Project Name

5.4.2 Construction Process

The variables were clearly stated and explained in this paper, facilitating the construction of the DAG. Additionally, the statistical model utilized was also clearly demonstrated. It became apparent however that the study was longitudinal, where time variables were included in their statistical analysis. This challenged the construction process. Due to controversy regarding whether or not time can be a causal variable in a DAG, it was decided to stick with a cross-sectional approach in this study. Therefore, the time variables were not included in the DAG.

Some background was given by the authors regarding the variables utilized for their research, and why they were used. An example of such an explanation is the following:

"To account for variability originating from the fact that each project is unique, we modeled an additional variable called project_name, which represents the names of the nine projects as a random effect."

In general, the authors' elaborations were however perceived as still lacking. It was for example not stated why Project Age was included in their statistical analysis.

5.5 DAG Construction: Co-Occurrence of Refactoring

The paper titled On the Co-Occurrence of Refactoring of Test and Source Code authored by Nagy and Abdalkareem [46] was featured in the 2022 MC. As the title indicates, the study aims to predict when test code refactoring co-occurs with source code refactoring. As all the participants in the 2022 Challenge were given the assignment of mining the SmartSHARK data set, this study used this data to study 77 open-source Java projects. They could study 60,465 commits and their character using the tool RefactoringMiner. The two research questions addressed in the study

included:

- **Q1:** How often do source and test code refactoring co-occur, and what are the refactoring types applied to test code?
- **Q2:** Can we predict when refactoring test code should be co-occurring with source code refactoring?

The authors could reveal that the majority of refactoring commits only refactor source code, representing 73.9%. In contrast, 8.2% of the commits only refactors test code, leading to 17.9% of co-occurring refactoring. The refactoring types most applied to test code in the co-occurring refactoring commits were Change Variable Type, Move Class, and Rename Method. For the second research question, they trained random forest classifiers to predict when a test code refactoring would co-occur with a source code refactoring. To make the predictions, they provided several features to their classifier which they considered to provide useful indications concerning when a test code refactoring should co-occur with a source code refactoring. These features were all extracted from the source code of ten selected projects and can be studied in Table 5.11. The selected projects utilized in their study were chosen based on the total amount of commits in each project:

"Since our analysis aimed to investigate the co-occurrence of test and source code refactoring opportunities, we needed to study projects with sufficient development history."

The results showed that they could accurately predict co-occurrence with the AUC values between 0.67–0.92, which is the Area under the ROC (Receiver operating characteristic) Curve.

5.5.1 DAG outlining: Source code refactoring influence on test code refactoring

The authors used descriptive statistics to answer the first research question, while the second was addressed using a random forest classifier for prediction. The DAG constructed is therefore based on the variables identified from Q2. The treatment and outcome variables were first identified. Given the aim of predicting the co-occurrence of test code refactoring and source code refactoring, the influence a source code refactoring would have on a test code refactoring occurring appeared to be the causal relationship of interest. From this, the treatment variable and outcome variable were defined as:

- **Treatment: Source Code Refactoring (SR).** Binary variable showcasing whether or not a refactoring of source code was conducted.
- **Outcome: Test Code Refactoring (TR).** Binary variable showcasing whether or not a refactoring of test code was conducted.

A set of features was provided to the predictive model created in the research to make the predictions. The authors presented a clear table with all features utilized. The table from the paper is presented in Table 5.11, including the feature names and

Table 5.11: Features utilized in the predictive model, along with descriptions from paper [46] and the shortenings used in the DAGs and code. The features were only extracted from changes done in source code, either in source code refactoring commits or co-occurring refactoring commits.

Short.	**Short. (code)**	**Feature**	**Description**
R	NmbRef	# of refactorings	Number of source code refactorings
L_L	L_Locations	# of left side locations	Number of locations touched by refactorings from the parent commit.
R_L	R_Locations	# of right side locations	Number of locations touched by refactorings from the child commit.
L_LOC	L_LOC	LOC left side	Lines of code touched in parent commit by refactorings.
R_LOC	R_LOC	LOC right side	Lines of code touched in child commit by refactorings.
Fs	NmbFiles	# of files	Number of files touched by refactorings.
AFs	AvgNmbFiles	Average number of files	Average number of files touched across refactorings in a commit.
RT_U	NmbRefType_UQ	# of unique refactoring types	Number of unique refactoring types applied in a commit.
CE_U	NmbCodeElem_UQ	# of unique code elements	Number of unique code elements identified by each refactor.
PR	PrevRef	# of previous refactorings	Number of refactorings each file has had previously with the average taken of all files.
RA	RefAge	Refactoring age	Number of days since the last refactoring for each file, with the averge taken as its value.
DR	DevRefExp	Developer refactoring experience	Number of refactorings applied by the developer previously to this commit.
DC	DevRefComExp	Developer refactoring commit experience	Number of refactoring commits made by the developer previously to this commit.
RT_C	*N.A.*	Refactoring type count	A set of features: One feature for each refactoring type in the commit, along with its number of occurrences as its value.
CE_C	*N.A.*	Code element type count	A set of features: One feature for each code element type in the commit, along with its number of occurrences as its value.

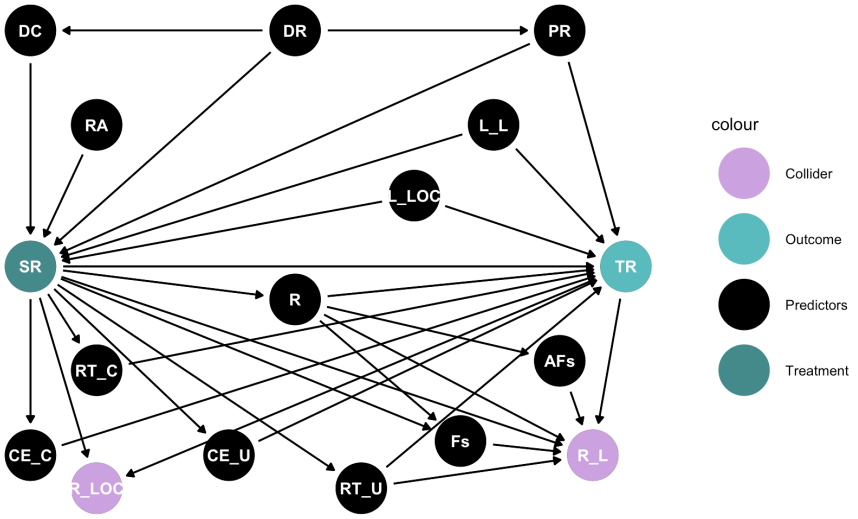


Figure 5.15: DAG showcasing the direct effect source code refactoring has on test code refactoring. All predictors utilized in the paper's analysis are therefore included. SR = Source Code Refactoring, TR = Test Code Refactoring, and the other shortenings for variables can be found in Table 5.11.

descriptions of each feature. The authors' rationale behind the choice of features is the following:

"In this study, we proposed using different features that we believe can give a good indication of whether test code should be refactored when source code is."

A DAG constructed with the treatment variable, outcome variable, and all features, is demonstrated in Figure 5.15. The motivations behind each arrow, apart from the arrows to the collider variables, are specified in Appendix B.6.

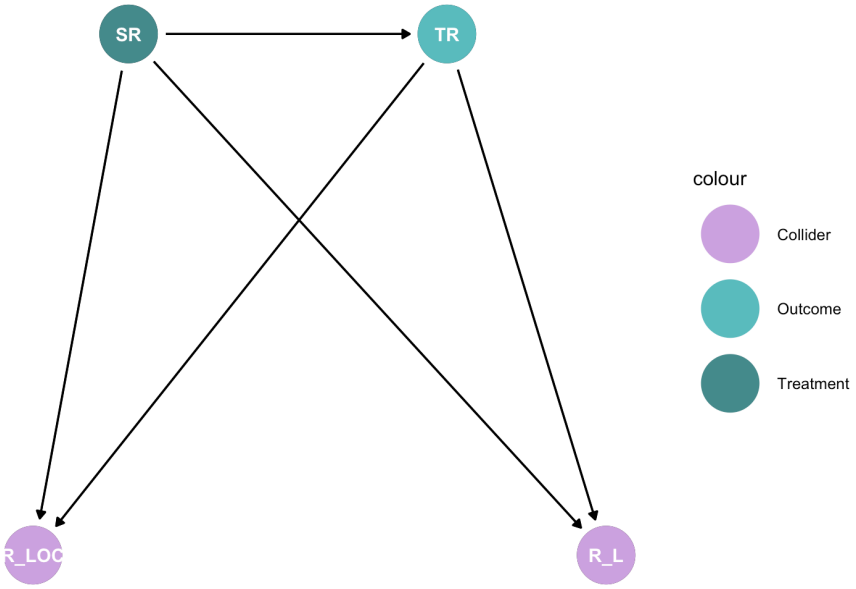


Figure 5.16: DAG showcasing the total effect source code refactoring has on test code refactoring. SR = Source Code Refactoring, TR = Test Code Refactoring, R_LOC = LOC right side, R_L = # of right side locations

As can be studied in Figure 5.15, two collider variables were identified in this scenario which are marked in purple in the DAG. The collider variables represent the features Lines of code touched in the child commit, R_LOC, and Number of locations touched by refactorings in the child commit, R_L. In addition to the DAG including all features, a more compact DAG was created including only the treatment, outcome, and these collider variables. The DAG is presented in Figure 5.16.

5.5.2 Verification of Colliders

One simple and general argument was considered when motivating the arrows from SR and TR to the two colliders:

What you do in this commit will influence what happens next.

The arrows from SR to both R_LOC and R_L can easily be motivated. More lines of code could likely be targeted in future commits if extensive changes have been made in a previous source code refactoring. An example could be if the last modification included adding functionality in the source code, and upcoming commits incorporating that functionality. Alternatively, refactoring code to make it more modular, organized, and efficient, could reduce the number of lines needing modification in the child commit. The arrows from TR to R_LOC and R_L relate to whether changes in the test code would influence upcoming changes in the source code. There could be several aspects motivating why this would be the case:

- Refactoring test code may uncover bugs, influencing upcoming source code refactorings and likely increasing the number of lines or locations in the child

commit.

- Extracting methods from test cases can reduce upcoming source code changes, affecting the number of lines or locations touched based on the reusability of the method.
- Test-Driven Development (TDD) and refactorings done in this process, can directly influence the number of lines or locations touched in the child commit, depending on the changes in test cases and their structural changes.
- Removing unnecessary or obsolete test cases/methods can impact the removal of dead or unused code in the source, which influences the number of lines of code that would be reduced.
- The type of test refactored (e.g., integration or unit) can affect the source code refactoring in the child commit. Changes in integration tests may expose necessary changes, leading to more locations being touched in the source code compared to modifications in unit tests.

The reasoning above could be applied to both colliders R_LOC, and R_Locations. Depending on whether the test code refactoring demands a more structural or extensive change in the source code, both collider variables could be influenced. For example, depending on the character of a bug revealed in a test code refactoring, the number of LOC or locations modified in the upcoming source code refactoring could be influenced. Appendix B.6 provides more thorough motivations for the colliders.

There could potentially be a relationship between R_LOC and R_Location, since they both are connected to the size of refactoring. If lines of code are being refactored, the number of locations might increase and vice versa. However, this relationship is not guaranteed, so no causal arrows were drawn between them. Even if such an arrow was included, these two features would still be considered colliders. The collider structure would have been broken if an arrow existed that pointed back at SR or TR. This was considered not to be likely since that would indicate that a future commit would have affected the current commit.

5.5.3 Construction Process

Constructing DAGs for the paper Co-Occurrence of Refactoring was a time-consuming process. The paper contained many features for conducting the prediction and no extensive elaborations were presented by the authors regarding why these variables were chosen, as stated in the citation in Section 5.5.1. Elucidating the relationships between all 15 features was intricate. However, the authors clearly presented the variables included in the predictive model. No manual investigation was therefore required in this regard.

5.6 Simulation Results

This section presents the results obtained from the simulations conducted. Section 5.6.1 presents the results from the theoretical SE scenario. Section 5.6.2 discusses the results obtained from the simulations based on the research paper Co-Occurence of Refactoring. The code utilized in this Section can be found in the replication package [11]. Appendix D presents what feature names correspond to the original feature names in the replication package of the studied paper. The variable names used in this study were defined to more clearly capture the descriptions presented of each feature in the paper.

5.6.1 Theoretical Scenario

This section explores the theoretical example presented in Section 5.2, where the definitions used to simulate data for the three variables can be found. As previously elaborated, the focus of interest in this example is the effect a developer's experience, DE, has on lines of code, LOC, produced in a day. In this example, the number of bugs, B, is the collider.

Table 5.12 shows the definitions utilized in Simstudy to create the synthetic data. The three variables are assumed to be positive counts, indicating that the Poisson distribution was suitable for all of them. However, LOC was approximated to follow a Normal distribution due to an assumed mean around 75 lines. The DE was set to have a mean of 5 years, and B was approximated to 0.5. Effect sizes were set to -0.95, 0.08, and -0.4. The effect of DE on LOC was set to be negative due to the assumption that less LOC will be produced by a developer with more experience. Similar reasoning was made regarding the effect of DE on B, where fewer bugs would likely be introduced with more experience. Variance for LOC was set to 15 and B was put on a log-link, see Section 4.3.1 for further explanation. Figure 5.17- 5.19 illustrate histograms of simulated data for all variables, with 2000 samples.

Table 5.12: The definitions used to simulate data for the theoretical example.

	Distr.	**Formula**	**Variance**	**Link**
DE	Poisson	5	-	-
LOC	Approx. Normal	70 - **0.95***DE	15	-
B	Poisson	-5 + 0.08*LOC - 0.4*DE	-	Log

[11] `https://github.com/amalev18/colliders-msr`

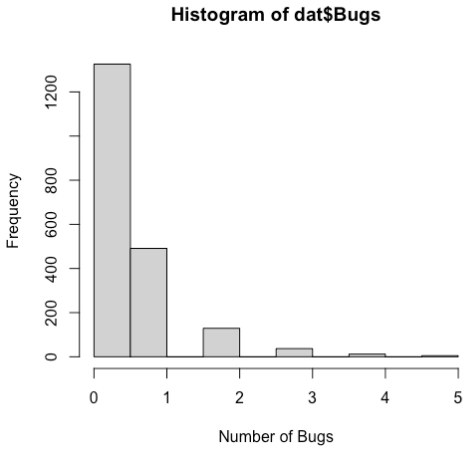


Figure 5.17: Generated data for B.

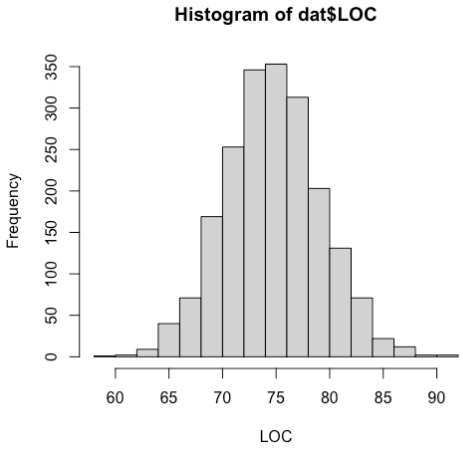


Figure 5.18: Generated data for LOC.

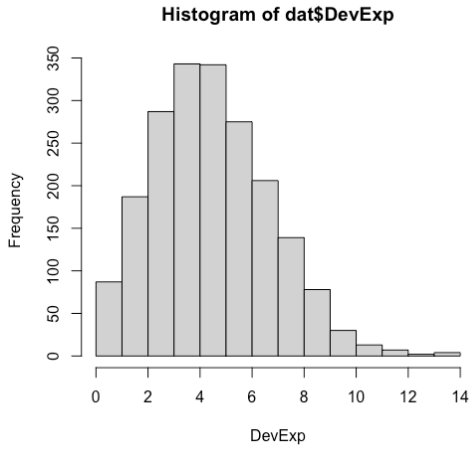


Figure 5.19: Generated data for DE.

After the generation of simulated data, two models were created, one with the inclusion of the collider, B, and one without. The **no collider** model is defined to the left, and the **collider** model is defined to the right:

$$
\begin{aligned}
\text{LOC} &\sim \text{Normal}(\mu, \sigma) \\
\mu &= \alpha + \beta_{DE} \cdot \text{DE} \\
\alpha &\sim \text{Normal}(70, 5) \\
\beta_{DE} &\sim \text{Normal}(0, 0.5) \\
\sigma &\sim \text{Exponential}(1)
\end{aligned}
$$

$$
\begin{aligned}
\text{LOC} &\sim \text{Normal}(\mu, \sigma) \\
\mu &= \alpha + \beta_{DE} \cdot \text{DE} + \beta_B \cdot \text{B} \\
\alpha &\sim \text{Normal}(70, 5) \\
\beta_{DE} &\sim \text{Normal}(0, 0.5) \\
\beta_B &\sim \text{Normal}(0, 0.5) \\
\sigma &\sim \text{Exponential}(1)
\end{aligned}
$$

For both models, priors were set to represent reasonable assumptions about the parameters involved. The prior for the intercept, α, assumes an average of approximately 70. The priors for the slopes β were centered around 0 with a standard deviation of 0.5. The standard deviation σ was set with the prior knowledge of it

being positive, and with a mean of 1.

Fitting the models with the simulated data resulted in the estimates in Table 5.13. The full output from this result can be found in Appendix C. The effect size for β_{DE} on LOC was the effect size of interest in this simulation, set to -0.95 in the simulated data. This was the effect size expected to be estimated by the models after the fitting. The model without a collider estimated the effect size to be -0.98, while the collider model gave the estimate -0.86. The model excluding the collider came closer to finding the set true effect in the simulated data than the model including the collider.

Table 5.13: Resulting estimates for β_{DE} retrieved when fitting the model in the simulation.

True Effect	**No Collider**	**Collider**
0.95	0.92	1.06

5.6.2 Co-Occurrence of Refactoring

The results from the simulations of the paper Co-Occurrence of Refactoring by Nagy and Abdalkareem [46] will be presented in this section. The DAGs constructed in Section 5.5.1 were utilized as the foundation for these simulations. As described in Section 5.5, the research aims to predict when test code refactoring is co-occurring with source code refactorings. The identified treatment is source code refactoring, SR, and the outcome is test code refactoring, TR. Two collider variables were identified in the DAG construction process: LOC right side, R_LOC, and the number of right side locations, R_Locations. In the upcoming paragraphs, the results from the generation of the simulated data will be described, followed by the models using the simulated data (with and without colliders) for both direct and total effect.

Simulation of Data

Table 5.14 demonstrates the distribution chosen for each feature utilized in the paper with the corresponding mean found in the original data. Accurate ranges capturing this real mean were successfully captured in the simulated data. Small deviations are present, due to randomization when simulating new data between runs. However, these deviations were small when present and still considered to capture an accurate mean value. All features, except AvgNmbFiles and RefAge, were count variables and therefore considered to follow the most simple count distribution Poisson. Four count variables had a large mean and were instead approximated with the Normal distribution. The two features AvgNmbFiles and RefAge were decimal numerals and simulated using the Normal distribution (no approximation).

Table 5.14: The original mean of each feature and the distribution chosen for the simulated data. The description of each feature can be found in Table 5.11.

Feature	Mean	Distribution
NmbRef	15	Poisson
L_Locations	29	Poisson
R_Locations	33	Poisson
L_LOC	640	Approx. Normal
R_LOC	619	Approx. Normal
NmbFiles	5.6	Poisson
AvgNmbFiles	1.2	Normal
NmbRefType_UQ	4.0	Poisson
NmbCodeElem_UQ	4.8	Poisson
PrevRef	31	Poisson
RefAge	114	Normal
DevRefExp	1543	Approx. Normal
DevRefComExp	100	Approx. Normal

The treatment variable, SR, and the outcome variable, TR, were further generated as binary values, showcasing if a refactoring of source code or test code had been done (1) or not (0). The set distributions for these variables and the probabilities of occurrence in the original data can be studied in Table 5.15. The occurrence probability was set to 0.92 for SR, and 0.26 for TR, which were extracted from the results in Q1 from the paper. These probabilities were also captured in the simulated data, with again some deviations.

Table 5.15: The simulated treatment and outcome variable, with the set distributions and probabilities of success for each variable.

Variable	Probability of Occurrence	Distr.
SR	0.92	Binary
TR	0.26	Binary

Throughout the process, two features—Refactoring Type count and Code Element Type count—were excluded from the simulation. These features consisted of lists of refactoring and code element types, in total approximately 100 different types. Their exclusion was based partly on the impracticality of integrating them into the simulation, and partly their negligible impact on the results since the majority of entries were zeros. An exception was additionally made regarding the LOC variables. Given the description of the variable in the paper and an initial investigation of the real data, it was classified as a count variable. However, it was noticed that it included negative values. In the code base of the paper, the LOC variables were named as differences. For example, the variable covering the amount of LOC touched in the child commit was named rightLocationDiff. That is, by LOC touched the authors also appeared to include the removal of lines, and measure it as a difference rather than a typical count. These negative numbers caused issues when simulating

the data, due to the need to logarithmize these variables (see Section 4.3.1 for explanations concerning the need for this). Given the small proportion of negative numbers, 0.4%, and the minimal impact on the mean, the rows containing negative numbers were removed.

Model: Total and Direct Effect
Both the total effect and the direct effect were examined when simulating the real research scenario, with and without colliders. Since the outcome variable TR was binary and not normally distributed, GLMs were utilized when constructing the models. All α priors were set to Normal(-1,1), and all β priors were set to Normal(0,0.5). The following models demonstrate the **no collider** model created on the left, and the **collider** model on the right, both showcasing the **total effect**:

$$\begin{aligned} \text{TR} &\sim \text{Bernoulli}(p), \\ \text{logit}(p) &= \alpha + \beta_{SR} \cdot \text{SR}, \\ \alpha &\sim \text{Normal}(-1, 1), \\ \beta_{SR} &\sim \text{Normal}(0, 0.5) \end{aligned}$$

$$\begin{aligned} \text{TR} &\sim \text{Bernoulli}(p) \\ \text{logit}(p) &= \alpha + \beta_{SR} \cdot \text{SR} + \beta_{R\_LOC} \cdot \text{R\_LOC} + \\ &\quad \beta_{R\_Locations} \cdot \text{R\_Locations} \\ \alpha &\sim \text{Normal}(-1, 1) \\ \forall\beta &\sim \text{Normal}(0, 0.5) \end{aligned}$$

The **no collider** model, showcasing the **direct effect**, included all features except the colliders, and was defined as:

$$\begin{aligned} \text{TR} &\sim \text{Bernoulli}(p), \\ \text{logit}(p) = \alpha &+ \beta_{RefAge} \cdot \text{RefAge} + \beta_{DevRefExp} \cdot \text{DevRefExp} + \beta_{L\_Locations} \cdot \text{L\_Locations} + \\ &\beta_{L\_LOC} \cdot \text{L\_LOC} + \beta_{DevRefComExp} \cdot \text{DevRefComExp} + \beta_{PrefRef} \cdot \text{PrefRef} + \\ &\beta_{SR} \cdot \text{SR} + \beta_{NmbFiles} \cdot \text{NmbFiles} + \beta_{NmbRef} \cdot \text{NmbRef} + \\ &\beta_{NmbRefType\_UQ} \cdot \text{NmbRefType\_UQ} + \beta_{NmbCodeElem\_UQ} \cdot \text{NmbRefType\_UQ} + \\ &\beta_{AvgNmbFiles} \cdot \text{AvgNmbFiles}, \\ \alpha &\sim \text{Normal}(-1, 1), \\ \forall\beta &\sim \text{Normal}(0, 0.5) \end{aligned}$$

The **collider** model, showcasing the **direct effect** and including all features, was defined as:

$$
\begin{aligned}
\text{TR} &\sim \text{Bernoulli}(p), \\
\text{logit}(p) = \alpha &+ \beta_{RefAge} \cdot \text{RefAge} + \beta_{DevRefExp} \cdot \text{DevRefExp} + \beta_{L\_Locations} \cdot \text{L\_Locations} + \\
&\beta_{L\_LOC} \cdot \text{L\_LOC} + \beta_{DevRefComExp} \cdot \text{DevRefComExp} + \beta_{PrefRef} \cdot \text{PrefRef} + \\
&\beta_{SR} \cdot \text{SR} + \beta_{NmbFiles} \cdot \text{NmbFiles} + \beta_{NmbRef} \cdot \text{NmbRef} + \\
&\beta_{NmbRefType\_UQ} \cdot \text{NmbRefType\_UQ} + \beta_{NmbCodeElem\_UQ} \cdot \text{NmbRefType\_UQ} + \\
&\beta_{AvgNmbFiles} \cdot \text{AvgNmbFiles} + \beta_{R\_LOC} \cdot \text{R\_LOC} + \beta_{R\_Locations} \cdot \text{R\_Locations} \\
\alpha &\sim \text{Normal}(-1, 1), \\
\forall\beta &\sim \text{Normal}(0, 0.5)
\end{aligned}
$$

All the count variables were standardized before being utilized in the models. Even though standardization might result in some loss of information and precision, it was chosen to be performed. The standardization facilitated the sampling given the characteristics of the data in the investigated research. The full results from running the models are detailed in Appendix C. The resulting estimates of the β parameter of interest, β_{SR}, from different models can be analyzed, and compared to the set true effect in the simulated data, in Table 5.16. These results are based on the results obtained when utilizing 8000 samples. The results vary slightly between runs with this sample size, which however is normal due to the randomization when generating data. In addition, it was found that the effect sizes shifted when the sample size increased for the total effect model including the collider variables. It did not approach the true effect size but became more distant from it with an increased amount of samples. The total effect model without the colliders remained within the same interval close to the true set effect size. No significant differences on the outcome scale could be determined when conducting the posterior predictive checks. This was the case for both the total effect and direct effect.

Table 5.16: Resulting estimates retrieved when fitting the models in the simulations.

β_{SR} / **Model**	**True Effect**	**No Collider**	**Collider**
Total effect	0.8	0.82	−6.51
Direct effect	0.5	0.52	0.13

5.7 Analysis Using Real Data from Subject Paper

This section presents the results when utilizing the original data from the paper Co-occurrence of Refactoring. It was identified that the file *commit_all_features.csv* in the original replication package contained the data for all features used in the study's predictions. However, it was impossible to directly use this CSV-file in the model created for this study. According to the authors, the data in their research was extracted solely from changes in the source code. As a result, no data points

were included in the data set indicating the presence of test code refactoring without source code refactoring. To run the models in this study, a variation is required of the three cases: (1) only SR, (2) only TR, and (3) co-occurrence of SR and TR. Without this variance in the treatment, SR, it would not be possible to obtain valuable measurements of the impact of the treatment variable. Moreover, it would not reflect all possible outcomes since a portion of the data would be ignored.

The full original raw data was attempted to be obtained to utilize all three cases in this study. This posed challenges, due to the repository's large size and lack of documentation or comments in the code. However, all raw data including test refactoring only data points was successfully obtained by making a small modification to a feature extraction file. The modification can be found in the replication package of this thesis[12].

Consequently, it was possible to analyze the total effect using the real data, as the necessary features were available. However, obtaining the full dataset for all features would have required replicating the entire study from the original paper, which was beyond the scope of this thesis. Therefore, only an analysis of the total effect of SR on TR using the original data from the paper was possible. The models used were based upon the ones created in the simulation but with adjusted priors set to Normal$(0, 0.05)$ $\forall\beta$ priors:

$$\begin{aligned} \text{TR} &\sim \text{Bernoulli}(p), \\ \text{logit}(p) &= \alpha + \beta_{SR} \cdot \text{SR}, \\ \alpha &\sim \text{Normal}(-1, 1), \\ \beta_{SR} &\sim \text{Normal}(0, 0.05) \end{aligned}$$

$$\begin{aligned} \text{TR} &\sim \text{Bernoulli}(p) \\ \text{logit}(p) &= \alpha + \beta_{SR} \cdot \text{SR} + \beta_{R\_LOC} \cdot \text{R\_LOC} + \\ &\quad \beta_{R\_Locations} \cdot \text{R\_Locations} \\ \alpha &\sim \text{Normal}(-1, 1) \\ \forall\beta &\sim \text{Normal}(0, 0.05) \end{aligned}$$

Table 5.17 presents the resulting estimates of the β_{SR} parameter when including and excluding the collider variables in the statistical model investigating the total effect, using various sample sizes. The full output from the results of 8000 samples can be found in Appendix C. The true effect size was unknown since the run included real data from the research paper. It was found that the effect size differed slightly when collider variables were included, but this difference was not significant. Additionally, it was observed that the effect size for both models decreased as the sample size increased. This pattern was similar to what was found for the collider model in total effect simulation, suggesting that something might be wrong as this should not be the case.

[12] `https://github.com/amalev18/colliders-msr.git`

Table 5.17: Resulting estimates retrieved for β_{SR} when fitting the models using the original data from the studied paper and the utilized sample size.

β_{SR} / **Sample Size**	**No Collider**	**Collider**
1000	−0.14	−0.14
5000	−0.59	−0.60
8000	−0.87	−0.89
10000	−1.06	−1.07

After analyzing the β estimates, a posterior predictive check was conducted. Figure 5.20 illustrates a plot of the posterior distributions for the two models. Both models show similar distributions: most data points are zero and a smaller share are ones, indicating that it is less likely for a TR to occur than a SR. The dark blue dots represent the replicated data, while the light blue bars represent the observed data. Both models seem to capture the observed data distributions well. The inclusion of collider variables appears to have no practical implications and does not significantly impact the distribution of data

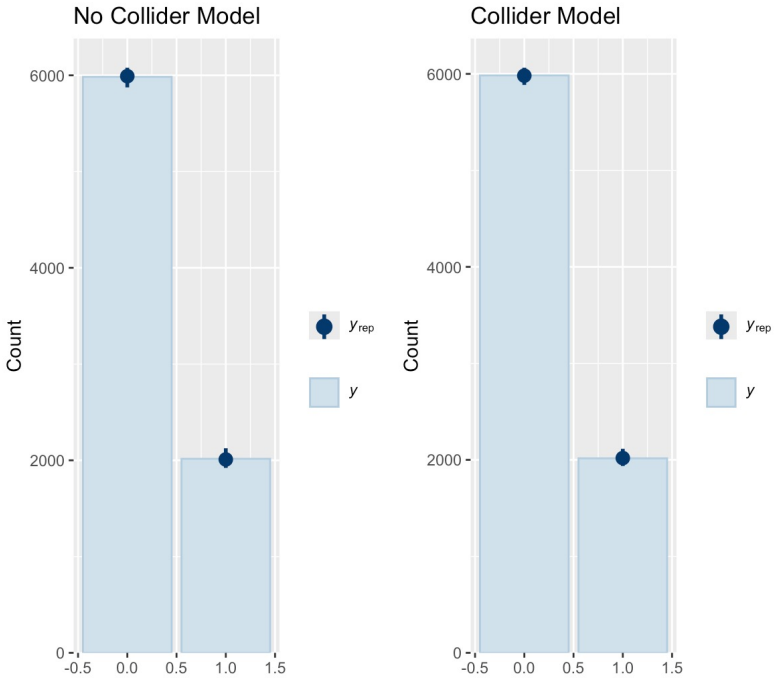


Figure 5.20: Posterior predictive check using original data for total effect.

6

Discussion

This study aimed to investigate the practical implications of accidentally including collider variables in research within the mining software repository domain. The methodology used to obtain the results also provided insights into the benefits and challenges of adding a causal perspective to the studies in this field. Sections 6.1 and 6.2 discuss the insights gained for each research question. Section 6.3 presents a proposed set of guidelines for MSR researchers. Section 6.4 discusses potential validity threats to the study. Section 6.5 recommends improvements for future work.

6.1 RQ1: Benefits and Challenges of Applying a Causal Approach to Studies in the MSR domain

Perceived Current Transparency and Reproducibility
When selecting suitable research papers, the two tracks MC and DPA from the MSR conference were investigated, revealing clear differences. Research papers in the MC track were generally more transparent and reproducible, often accompanied by a replication package. The variables utilized in their analyses were also clearly presented. Additionally, more papers in the MC track explained why the variables were chosen. Specifically, 90% of the MC track papers explicitly stated the variables included in the data analysis, compared to 16.7% in the DPA track. Conversely, DPA papers more commonly described the general data collection process, including outlining the data sources. This was done in 91.7% of the cases.

Replication packages were provided for 83.3% of the papers in the MC track during the years investigated (2022 to 2023), while only 58.8% of the more recent DPA papers (2018 to 2023) included replication packages. However, a positive trend was identified for DPA papers: when looking at papers from 2018 to 2023, 58.8% provided a replication package, compared to 33.3% from 2015 to 2017. These differences could be due to the characteristics of the respective tracks. The MC track is a competition where researchers are given the same dataset to conduct their research. The authors further determine the scope and aim of their investigation and specify which variables they have chosen to use in their analysis. This likely explains why the variables and motivations behind why they were chosen are stated more clearly

in the MC papers. On the other hand, the DPA track highlights papers that have conducted excellent research with great impacts and advancements in the field. As previously stated, these papers more commonly presented the data collection process rather than detailing which variables were used for data analysis.

Challenges Encountered
Identifying variables during the sampling process was intricate, especially when examining papers from the DPA track, due to the lack of clarity regarding which variables were included in the data analysis. In such cases, it was necessary to examine the replication packages to identify the variables utilized. The repositories provided were typically large, lacking comments and vague descriptions of navigation in the repository, making it cumbersome to familiarize oneself with each repository. This added to the perception of lower transparency for the DPA papers.

During this study, DAGs were constructed based on three selected papers: one paper from those that received a DPA and two papers from the MC track. The problem of missing documentation of included variables also caused challenges when constructing DAGs. The degree to which authors elaborated on assumptions regarding variables varied widely. In the final paper chosen for simulations and analysis, only one sentence elaborated on why particular variables were included. This particular sentence[1] provided the sole rationale for choosing the features based on the authors' perception of good indicators for prediction. In other cases, such as in Atoms of Confusion from the DPA track, many assumptions were identified within the text. However, these assumptions were nested throughout the text, requiring a thorough reading to understand the authors' perceptions. The main challenge when constructing DAGs was the lack of clearly stated assumptions behind variables and their relationships. This could be a question of whether or not the assumptions of the authors were correctly interpreted and captured during the construction of DAGs, posing a threat to the validity of the study's results. Further discussion on this issue will be addressed in Section 6.4.

Possible Benefits of Adding Causality Aspects
The initial parts of the thesis, as anticipated from previous research, highlighted a lack of a causal perspective in studies conducted within the mining software repository domain. During the sampling process, no papers were identified in either conference track that addressed causality. In some cases, papers included keywords related to causality, but these were disclaimers rather than inclusion of causality. The lack of transparency in the research was perceived as a significant obstacle, a point strengthened by previous studies on transparency in the MSR domain [78, 77]. Adding causality into the methodology of such research could increase transparency by clarifying the underlying assumptions authors make when conducting their research, thus eliminating the need for interpretation. DAGs could graphically achieve this, making it clear to readers which variables were utilized in the analysis and their relationships. As seen in this study, identifying and interpreting assumptions and

[1] "In this study, we proposed using different features that we believe can give a good indication of whether test code should be refactored when source code is."[46]

variables is time-consuming and could affect the validity of research building on top of it. Therefore, incorporating causality could enable more robust advancements in the MSR domain by reducing ambiguity in terms of underlying assumptions. Confirming this statement would require further research.

Additionally, a statistical analysis was conducted based on a research paper to investigate the possible benefits of adding causality to research methodologies. In the paper On the Co-Occurrence of Refactoring of Test and Source Code [46], collider variables were identified. The collider variables found were R_LOC and R_Locations, referring to lines of code and locations touched by refactorings in the child commit. The total effect of source code refactoring (SR) on test code refactoring (TR) was investigated using the real dataset, but the direct effect was not due to challenges in acquiring the full raw data. Small differences in estimates were observed when the collider variables were included compared to when they were excluded, but no significant differences were seen in the outcome variable. The validity of these results is questionable, as the effect sizes changed with increased sample size, which should not be the case according to McElreath [40], where Bayesian estimates remain stable regardless of sample size. One aspect discussed further in Section 6.4 is the lack of knowledge regarding how the real data from the paper was generated. The data was not collected for the purpose formulated in this study, which could have impacted the outcome. For example, the authors excluded all raw data points where only the outcome variable was present in their dataset (TR being 1 while SR was 0), meaning no variance was present for the treatment variable. Consequently, no clear evidence can be presented in terms of benefits concerning practical implications. Future work is required to investigate the behavior where the effect size changes with increased sample size.

Key Findings of RQ1

- The current lack of transparency in the field posed challenges in building DAGs, as variables and assumptions were not clearly stated.
- Incorporating causality could improve transparency, potentially leading to more robust advancements in the long run.
- Although the investigation explored the practical implications of excluding collider variables, no such conclusions could be drawn. Future research is needed.

6.2 RQ2: Recommendations to Mitigate the Potential Challenges of Conditioning on Colliders in MSR-Related Studies

Impacts of Colliders in Simulations

The impact of colliders and ways to mitigate them were investigated through simulations. The initial simulation demonstrated a hypothetical scenario that could occur within SE. Results from this theoretical example showed that the estimated

effect size was closer to the true effect size when the collider variable was excluded, indicating that excluding the collider variable affected the parameter estimates, even though the difference was not substantial. These results indicate the need to identify and mitigate collider variables before conducting an analysis. Although the difference was not large, excluding the collider led to an estimate closer to the true effect size.

The simulations of the real research scenario revealed a larger difference in the resulting estimates. For investigating the direct effect, the true effect size of the treatment variable on the outcome variable was set to 0.5. The model excluding the colliders provided an estimate of 0.52, while the model including the colliders provided an estimate of 0.13. However, when examining the practical impacts of these estimates, no significant difference was identified. This could be due to backdoor paths being opened when conditioning on colliders, but the effects transported through these backdoor paths were negligible. Other research scenarios are needed to confirm if practical implications could be identified in other cases. The difference in estimates found in this study highlights that collider variables worsened the sought-after causal effect of the treatment on the outcome, as seen in the theoretical simulation.

Another aspect that could have influenced the simulation results is the data generation process for the synthetic data. The data was generated by capturing the mean of each variable from the real data while maintaining reasonable effect sizes for all variables. The simulation was sensitive to changes, partly because the variables in the dataset had very different magnitudes. This made simulating data similar to the ones utilized in the studied paper challenging, potentially leading to proportions of effect sizes that were not entirely reasonable and affecting the simulation results. Unlike the real data, the estimates from the synthetic data did not continuously differ with increased sample size. One possible explanation for this is the data generation was under the control of the authors of this study. As a result, the simulated data could be seen as cleaner compared to real data, which could have further affected the results.

Regarding the simulation of the total effect, the effect size was again close to the true one in the scenario where colliders were excluded. However, as found in the analysis using the real dataset, the effect size for the model including the collider variables changed with increased sample size, becoming more distant from the true effect. Due to the unexpected behavior, it is impossible to draw robust conclusions from these results.

The results from the simulations of the real research scenario did not reveal as significant practical differences as anticipated when colliders were included. Some unexpected behavior was also identified. The research methodology used to investigate colliders involved immediately simulating the entire DAG. The larger DAG seen in Figure 5.15 highlights the complexity of the research scenario. For future research, the methodology could be improved to facilitate a better understanding of the scenario under study and the impact of colliders. Smaller sub-DAGs could be simulated before the large DAG, including parts of the DAGs per simulation, to gain a fuller picture of the collider effects. The current methodology did not

allow for such incremental construction, thus eliminating the possibility of gaining progressive and more robust insights.

Characteristics of Studied Research Paper
Despite that no practical implications was identified in this study, neither in simulations nor during the analysis utilizing real data, the fact remains that the studied paper did accidentally condition on colliders, assuming the constructed DAG accurately represents the paper. A vast amount of literature demonstrates how including collider variables can bias the final result of a study (see for example [40]). The paper was chosen for this study because colliders were identified, but it also presented other characteristics that made it suitable. Notably, it resembles the concept of a causal salad, where numerous features were more or less randomly included in a machine learning model to draw conclusions. The authors utilized 15 features in their predictive model, chosen solely based on their perception of which features would be useful. The DAG created for this paper, presented in Figure 5.15, clearly shows the complexity of the relationships among the utilized variables. According to causal inference theory, it is crucial to include the appropriate variables to obtain an unbiased causal effect [40]. Without considering these relationships, there is a notable risk of including incorrect variables, as demonstrated in the studied paper.

To avoid conditioning on collider variables, researchers should be aware of the variables they incorporate in their studies and the reasons for their inclusion. Using causal models, like the DAGs utilized in this study, clarifies how variables relate to each other. This will reveal which variables that are colliders and should be excluded from the analyses. Even if the study's aim is purely predictive, understanding causality helps explain why a phenomenon occurs. In the long run, incorporating causality will lead to more robust predictions [57]. Additional research scenarios would be needed to understand the impacts of such measures in real research.

Key Findings of RQ2

- Excluding colliders led to more accurate estimates of the sought-after effect size of a treatment variable in both the simulated theoretical SE scenario and one simulation of the real research scenario.
- DAGs can be utilized as a tool to mitigate the risk of accidentally conditioning on colliders. In the studied paper, using DAGs could have prevented its similarities to the causal salad.

6.3 Guidelines for MSR Researchers

Based on this study, a set of guidelines is proposed for MSR researchers to ensure awareness of causality and to avoid conditioning on colliders. These steps should be performed before conducting a statistical analysis to select appropriate variables:

1. Compile a list of all variables intended for use in the analysis. Justify the inclusion of each variable.

2. Begin the construction of a DAG by clearly defining and motivating the selection of treatment and outcome variables of interest.
3. Sequentially incorporate one variable at a time into the DAG. Draw directional arrows to connect the newly added variable to those already included. Document the underlying assumptions or evidence supporting each arrow's direction. Repeat this process until all variables from the list are incorporated into the DAG.
4. Engage other researchers to review the DAG critically. Assess the presence of arrows and their direction.
5. Identify any colliders present within the DAG. Exclude these variables from further analysis.
6. Continue with the statistical analysis using the validated set of variables.
7. Clearly outline this entire process in the research paper to ensure transparency for readers.

6.4 Threats to Validity

This section addresses the threats to the validity of the study, guided by the framework provided by Menzies and Shepperd [41], which describes twelve common issues in software analytics. Additionally, it considers issues of generalizability in sampling methodologies within SE, as discussed by Baltes and Ralph [5].

One identified issue by Menzies and Shepperd [41] is the oversight of related work. While the literature review for this study included recent and relevant research, it did not follow a specific strategy like systematic review or snowball technique. This was primarily due to time constraints and the focus on providing background rather than forming the foundation for the study's results. While systematic reviews are not always feasible, this approach may be considered a limitation. Instead, gaps were identified through guidance from supervisors and exploration of suitable sources related to causal inference across various research domains until saturation was reached.

The study employed a purposive sampling strategy by selecting papers from the MSR conference, which introduces a potential bias since it relies on the author's judgment. This approach may also be seen as using convenience data rather than ensuring relevance [41]. The decision to focus on Mining Challenge (MC) papers could be perceived as convenient due to their high transparency and reproducibility, though they are still considered as representative of the field.

Evaluating results from mining software repository research may introduce sampling biases [5]. Each repository could exhibit biases and may not be representative of the broader software industry, including both private and public repositories. Additionally, focusing solely on papers that received DPA or participated in the MC at the MSR Conference might introduce bias. The selection process for these papers lacks transparency from the MSR committee, raising uncertainty about their

representativeness. However, since this study relies on a specific paper, a discussion about generalizability is not essential.

Constructing DAGs relies on interpreting authors' assumptions from the studies investigated, introducing a risk of misinterpretation. Consequently, DAGs may never perfectly capture true causal relationships, as defining these relationships is often challenging, and human judgment and assumptions may be inaccurate. Nevertheless, DAGs remain valuable for analysis as they provide a structured method to represent relationships and identify the need to condition on certain variables.

Considering the impact of heteroscedasticity and outliers, such as heavy tails [41], the simulated data aimed to mirror characteristics from the real data of the studied research paper or theoretical example. The simulated data captured mean values based on real data characteristics, though not all variable information was fully replicated. Ground distributions, like Poisson for counts, were utilized to capture the means rather than variances. Some aspects, such as the full range of LOC, were not fully accounted for, and data was standardized before running the models. In the final model for direct effects, two features were excluded for practicality, potentially disregarding outcome effects. These decisions aimed to improve simulation efficiency, despite potential information loss. Conversely, the real data collection process in this study was beyond the authors' control, including data quality. The original dataset was modified for simulation purposes while still maintaining the original study's statistical properties.

A significant threat to validity is the potential of the study being underpowered [41], implying insufficient effect sizes might have been detected. While the simulated data showed considerable results for effect sizes, the analysis of real data did not reveal any significant differences in estimates. The synthetic data, generated by the authors, aimed to explore the impact of conditioning on colliders on final results. Effect sizes for colliders, treatment, and outcome variables were prioritized to be set sufficiently large. Given the complexity of variables relationships in the research scenario, some effects may have been underestimated. The relationship between all variable effect sizes may not have been fully accurate, but the focus was to capture the mean, reasonable ranges of variance, and effect size from the real dataset.

Other noteworthy threats include uncertainties in the results from analyzing the original data, particularly observed variations in the β estimates across different sample sizes. Such variations were unexpected but evident. Therefore, results presented in Section 5.7 require further validation to ensure their reliability.

6.5 Future Work

There are several opportunities for improvement and expansion based on the research conducted in this study. First and foremost, a thorough re-analysis of this study's simulations should be conducted to identify any potential errors. Given the variability observed in β-estimates across different sample sizes, it is recommended to further investigate methods that could mitigate this variability and improve the model's reliability. This iterative process should continue until the model demon-

strates robustness. Once the robustness is confirmed, conclusions can be drawn regarding practical implications for real data from the study Co-occurrence of Refactoring.

Alternatively, the methodology employed in this study could be replicated and applied across other research papers. This involves systematically applying the proposed methodology in this study to MSR research or different domains in SE. Such an approach would help clarify the prevalence and significance of collider bias across various contexts, validate the methodology's effectiveness, and improve its generalizability and robustness.

Lastly, there is an opportunity to evaluate the transparency in how researchers express their assumptions when incorporating causality. This evaluation would assess the clarity of these assumptions and examine their impact on the interpretation of causal inferences in SE research.

7

Conclusion

This study investigated the practical impacts of applying a causal approach to research in the mining software repository domain. Initially, a perceived lack of transparency and, to some extent, reproducibility was observed in the research papers reviewed from the MSR conference. This became apparent when constructing DAGs based on other researchers' work, as variables, selection processes, and underlying assumptions were not always clearly described. Incorporating causality could potentially have increased transparency by clearly illustrating the relationships among variables using DAGs. Without such clarity, future research building upon these findings may have been compromised. Determining the practical implications of excluding collider variables from the studied research scenario could not been proved. Future investigation is encouraged to explore the unexpected behavior observed during the analysis of the original dataset from the studied paper.

Based on the simulations conducted, more accurate estimates of the sought after causal effect were obtained when excluding collider variables. This was evident in both the theoretical SE scenario and the simulation of the direct effect using real research data. Despite these differences in estimates, no practical implications were observed on the final outcomes. Using DAGs is recommended as a strategy to mitigate the risk of accidentally including colliders in research, which complements the previous mentioned benefit. The studied research paper appeared to lack a thorough analysis of variables included in their analyses, showing similarities to what could be described as a causal salad. Based on the findings, it is recommended to incorporate a causal perspective into research within this domain and utilize DAGs to identify potential colliders. This approach will facilitate the selection of variables for future analyses.

Bibliography

[1] Carlos D. A. de Almeida, Diego N. Feijó, and Lincoln S. Rocha. "Studying the Impact of Continuous Delivery Adoption on Bug-Fixing Time in Apache's Open-Source Projects". In: *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*. 2022, pp. 132–136. DOI: 10.1145/3524842.3528049.

[2] Eman Abdullah AlOmar et al. "An Exploratory Study on Refactoring Documentation in Issues Handling". In: *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*. 2022, pp. 107–111. DOI: 10.1145/3524842.3528525.

[3] Suchinta Arif et al. "Causal drivers of climatemediated coral reef regime shifts". In: *Ecosphere* 13 (Mar. 2022). DOI: 10.1002/ecs2.3956.

[4] Amirreza Bagheri and Péter Hegeds. "Is Refactoring Always a Good Egg? Exploring the Interconnection Between Bugs and Refactorings". In: *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*. 2022, pp. 117–121. DOI: 10.1145/3524842.3528034.

[5] Sebastian Baltes and Paul Ralph. "Sampling in software engineering research: a critical review and guidelines". In: *Empirical Softw. Engg.* 27.4 (July 2022). ISSN: 1382-3256. DOI: 10.1007/s10664-021-10072-8. URL: https://doi.org/10.1007/s10664-021-10072-8.

[6] V. R. Basili and G. Caldiera. "The goal question metric paradigm". In: *Encyclopedia of Software Engineering* 2 (1994), pp. 528–532. URL: https://www.cs.umd.edu/~basili/publications/technical/T89.pdf.

[7] Kenneth J. Berry and Janis E. Johnston. "Two-Sample Tests". In: *Statistical Methods: Connections, Equivalencies, and Relationships*. Cham: Springer Nature Switzerland, 2023, pp. 91–169. ISBN: 978-3-031-41896-9. DOI: 10.1007/978-3-031-41896-9_4. URL: https://doi.org/10.1007/978-3-031-41896-9_4.

[8] Felivel Camilo, Andrew Meneely, and Meiyappan Nagappan. "Do Bugs Foreshadow Vulnerabilities? A Study of the Chromium Project". In: *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. 2015, pp. 269–279. DOI: 10.1109/MSR.2015.32.

[9] Patrick Chadbourne and Nasir U. Eisty. "Applications of Causality and Causal Inference in Software Engineering". In: (2023), pp. 47–52. DOI: 10.1109/SERA57763.2023.10197835.

[10] Arifa I. Champa et al. "Insights into Female Contributions in Open-Source Projects". In: *2023 IEEE/ACM 20th International Conference on Mining Soft-*

ware Repositories (MSR). 2023, pp. 357–361. DOI: 10.1109/MSR59073.2023.00055.

[11] Yang Chen et al. "A Machine Learning Approach for Vulnerability Curation". In: *2020 IEEE/ACM 17th International Conference on Mining Software Repositories (MSR)*. 2020, pp. 32–42. DOI: 10.1145/3379597.3387461.

[12] Morakot Choetkiertikul et al. "Characterization and Prediction of Issue-Related Risks in Software Projects". In: *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. 2015, pp. 280–291. DOI: 10.1109/MSR.2015.33.

[13] Carlos Cinelli, Andrew Forney, and Judea Pearl. "A Crash Course in Good and Bad Controls". In: *SSRN Electronic Journal* (Jan. 2020). DOI: 10.2139/ssrn.3689437.

[14] Daniel Alencar da Costa et al. "The Impact of Switching to a Rapid Release Cycle on the Integration Delay of Addressed Issues - An Empirical Study of the Mozilla Firefox Project". In: *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*. 2016, pp. 374–385.

[15] Francisco Gomes de Oliveira Neto et al. "Evolution of statistical analysis in empirical software engineering research: Current state and steps forward". In: *Journal of Systems and Software* 156 (2019), pp. 246–267. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2019.07.002. URL: https://www.sciencedirect.com/science/article/pii/S0164121219301451.

[16] Jean C. Digitale et al. "Key concepts in clinical epidemiology: collider-conditioning bias". In: *Journal of Clinical Epidemiology* 161 (2023), pp. 152–156. ISSN: 0895-4356. DOI: https://doi.org/10.1016/j.jclinepi.2023.07.004. URL: https://www.sciencedirect.com/science/article/pii/S0895435623001762.

[17] Amir Feder et al. "Causal Inference in Natural Language Processing: Estimation, Prediction, Interpretation and Beyond". In: *Transactions of the Association for Computational Linguistics* 10 (2022). Ed. by Brian Roark and Ani Nenkova, pp. 1138–1158. DOI: 10.1162/tacl_a_00511. URL: https://aclanthology.org/2022.tacl-1.66.

[18] Andrew Forney and Scott Mueller. In: *Journal of Causal Inference* 10.1 (2022), pp. 141–173. DOI: doi:10.1515/jci-2021-0048. URL: https://doi.org/10.1515/jci-2021-0048.

[19] Carlo A. Furia, Richard Torkar, and Robert Feldt. "Towards Causal Analysis of Empirical Software Engineering Data: The Impact of Programming Languages on Coding Competitions". In: *ACM Transactions on Software Engineering and Methodology* 33.1 (Nov. 2023), pp. 1–35. ISSN: 1557-7392. DOI: 10.1145/3611667. URL: http://dx.doi.org/10.1145/3611667.

[20] Mohammad Gharehyazie, Baishakhi Ray, and Vladimir Filkov. "Some from Here, Some from There: Cross-Project Code Reuse in GitHub". In: *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. 2017, pp. 291–301. DOI: 10.1109/MSR.2017.15.

[21] Nicolas E. Gold and Jens Krinke. "Ethical Mining - A Case Study on MSR Mining Challenges". In: *2020 IEEE/ACM 17th International Conference on Mining Software Repositories (MSR)*. 2020, pp. 265–276. DOI: 10.1145/3379597.3387462.

[22] Keith Goldfeld and Jacob Wujciak-Jens. "simstudy: Illuminating research methods through data generation". In: *Journal of Open Source Software* 5.54 (2020), p. 2763. DOI: 10.21105/joss.02763. URL: https://doi.org/10.21105/joss.02763.

[23] Dan Gopstein et al. "Prevalence of confusing code in software projects: atoms of confusion in the wild". In: *Proceedings of the 15th International Conference on Mining Software Repositories*. MSR '18. Gothenburg, Sweden: Association for Computing Machinery, 2018, pp. 281–291. ISBN: 9781450357166. DOI: 10.1145/3196398.3196432. URL: https://doi.org/10.1145/3196398.3196432.

[24] Konstantin Grotov et al. *A Large-Scale Comparison of Python Code in Jupyter Notebooks and Scripts*. 2022. arXiv: 2203.16718 [cs.SE].

[25] Christopher Harris, Abigail Terry, and Giselle P Truman. "Spreading Love and Margarine: An Examination of the Butter-Splitter Correlation in Maine". In: *The Journal of Culinary Chemistry and Sociological Studies* 267.25 (2024).

[26] Johannes Härtel and Ralf Lämmel. "Operationalizing Threats to MSR Studies by Simulation-Based Testing". In: *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*. 2022, pp. 86–97. DOI: 10.1145/3524842.3527960.

[27] Ahmed E. Hassan. "The road ahead for Mining Software Repositories". In: *2008 Frontiers of Software Maintenance*. 2008, pp. 48–57. DOI: 10.1109/FOSM.2008.4659248.

[28] Hadi Hemmati et al. "The MSR Cookbook: Mining a decade of research". In: *2013 10th Working Conference on Mining Software Repositories (MSR)*. 2013, pp. 343–352. DOI: 10.1109/MSR.2013.6624048.

[29] Miguel Hernán. "The C-Word: Scientific Euphemisms Do Not Improve Causal Inference From Observational Data". In: *American Journal of Public Health* 108 (Mar. 2018), e1–e4. DOI: 10.2105/AJPH.2018.304337.

[30] Miguel A. Hernán, Sonia Hernández-Díaz, and James M. Robins. "A Structural Approach to Selection Bias". In: *Epidemiology* 15.5 (2004), pp. 615–625. ISSN: 10443983. URL: http://www.jstor.org/stable/20485961 (visited on 01/26/2024).

[31] Miguel A. Hernán, John Hsu, and Brian Healy. "A Second Chance to Get Causal Inference Right: A Classification of Data Science Tasks". In: *CHANCE* 32 (1 Jan. 2019), pp. 42–49. ISSN: 0933-2480. DOI: 10.1080/09332480.2019.1579578. URL: https://www.tandfonline.com/doi/full/10.1080/09332480.2019.1579578.

[32] Miguel A. Hernán and Susana Monge. "Selection Bias Due to Conditioning on a Collider". In: *BMJ (Clinical Research Ed.)* 381 (2023), p. 1135. DOI: 10.1136/bmj.p1135.

[33] Andre Hora. "What Code Is Deliberately Excluded from Test Coverage and Why?" In: *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. 2021, pp. 392–402. DOI: 10.1109/MSR52588.2021.00051.

[34] Anisha Islam et al. "Evolution of the Practice of Software Testing in Java Projects". In: *2023 IEEE/ACM 20th International Conference on Mining Soft-*

ware Repositories (MSR). 2023, pp. 367–371. DOI: 10.1109/MSR59073.2023.00057.

[35] Fatemeh Khoshnoud et al. "Which bugs are missed in code reviews: An empirical study on SmartSHARK dataset". In: *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*. 2022, pp. 01–05. DOI: 10.1145/3524842.3527997.

[36] Oz Kilic, Nathaniel Bowness, and Olga Baysal. "Keep the Ball Rolling: Analyzing Release Cadence in GitHub Projects". In: *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. 2023, pp. 372–376. DOI: 10.1109/MSR59073.2023.00058.

[37] Zoe Kotti and Diomidis Spinellis. "Standing on Shoulders or Feet? The Usage of the MSR Data Papers". In: *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. 2019, pp. 565–576. DOI: 10.1109/MSR.2019.00085.

[38] Demetrios N. Kyriacou, Philip Greenland, and Mohammad A. Mansournia. "Using Causal Diagrams for Biomedical Research". In: *Annals of Emergency Medicine* 81 (5 May 2023), pp. 606–613. ISSN: 01960644. DOI: 10.1016/j.annemergmed.2022.08.014. URL: https://linkinghub.elsevier.com/retrieve/pii/S0196064422005789.

[39] Yalin Liu, Jinfeng Lin, and Jane Cleland-Huang. *Traceability Support for Multi-Lingual Software Projects*. 2020. arXiv: 2006.16940 [cs.SE].

[40] Richard McElreath. *Statistical rethinking : a Bayesian course with examples in R and Stan (2nd ed.)* CRC Press/Taylor & Francis Group Boca Raton, 2020.

[41] Tim Menzies and Martin Shepperd. "Bad smells in software analytics papers". In: *Information and Software Technology* 112 (2019), pp. 35–47. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2019.04.005. URL: https://www.sciencedirect.com/science/article/pii/S095058491930076X.

[42] Nandita Mitra, Jason Roy, and Dylan Small. "The Future of Causal Inference". In: *American Journal of Epidemiology* 191 (10 Sept. 2022), pp. 1671–1676. ISSN: 0002-9262. DOI: 10.1093/aje/kwac108. URL: https://academic.oup.com/aje/article/191/10/1671/6618833.

[43] Ambarish Moharil et al. "Between JIRA and GitHub: ASFBot and its Influence on Human Comments in Issue Trackers". In: *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*. 2022, pp. 112–116. DOI: 10.1145/3524842.3528528.

[44] Hamid Mohayeji et al. "Investigating the Resolution of Vulnerable Dependencies with Dependabot Security Updates". In: *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. 2023, pp. 234–246. DOI: 10.1109/MSR59073.2023.00042.

[45] *MSR 2024 - 21st IEEE/ACM International Conference on Mining Software Repositories*. https://conf.researchr.org/home/msr-2024. Accessed: February 8, 2024. 2024.

[46] Nicholas Alexandre Nagy and Rabe Abdalkareem. "On the Co-Occurrence of Refactoring of Test and Source Code". In: *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*. 2022, pp. 122–126. DOI: 10.1145/3524842.3528529.

[47] Feifei Niu et al. “The ABLoTS Approach for Bug Localization: is it replicable and generalizable?” In: *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. 2023, pp. 576–587. DOI: 10.1109/MSR59073.2023.00083.

[48] Ana Rita Nogueira et al. “Methods and tools for causal discovery and causal inference”. In: *WIREs Data Mining and Knowledge Discovery* 12 (2 Mar. 2022). ISSN: 1942-4787. DOI: 10.1002/widm.1449. URL: https://wires.onlinelibrary.wiley.com/doi/10.1002/widm.1449.

[49] Humphrey O. Obie et al. *On the Violation of Honesty in Mobile Apps: Automated Detection and Categories*. 2022. arXiv: 2203.07547 [cs.SE].

[50] Luca Pascarella and Alberto Bacchelli. “Classifying Code Comments in Java Open-Source Software Systems”. In: *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. 2017, pp. 227–237. DOI: 10.1109/MSR.2017.63.

[51] Neil Pearce and Debbie A Lawlor. “Causal inferenceso much more than statistics”. In: *International Journal of Epidemiology* 45 (6 Dec. 2016), pp. 1895–1903. ISSN: 0300-5771. DOI: 10.1093/ije/dyw328. URL: https://academic.oup.com/ije/article/45/6/1895/2999350.

[52] J. Pearl, M. Glymour, and N.P. Jewell. *Causal Inference in Statistics: A Primer*. Wiley, 2016. ISBN: 9781119186847. URL: https://books.google.se/books?id=L3G-CgAAQBAJ.

[53] Judea Pearl. *Causality*. 2nd ed. Cambridge University Press, 2009. DOI: 10.1017/CBO9780511803161.

[54] Judea Pearl. “The seven tools of causal inference, with reflections on machine learning”. In: *Commun. ACM* 62.3 (Feb. 2019), pp. 54–60. ISSN: 0001-0782. DOI: 10.1145/3241036. URL: https://doi.org/10.1145/3241036.

[55] Judea Pearl. “Theoretical Impediments to Machine Learning With Seven Sparks from the Causal Revolution”. In: *CoRR* abs/1801.04016 (2018). arXiv: 1801.04016. URL: http://arxiv.org/abs/1801.04016.

[56] Anthony Peruma et al. “Refactoring Debt: Myth or Reality? An Exploratory Study on the Relationship Between Technical Debt and Refactoring”. In: *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*. 2022, pp. 127–131. DOI: 10.1145/3524842.3528510.

[57] Niklas Pfister, Peter Bühlmann, and Jonas Peters. “Invariant Causal Prediction for Sequential Data”. In: *Journal of the American Statistical Association* 114.527 (2019), pp. 1264–1276. DOI: 10.1080/01621459.2018.1491403. eprint: https://doi.org/10.1080/01621459.2018.1491403. URL: https://doi.org/10.1080/01621459.2018.1491403.

[58] João Felipe Pimentel et al. “A Large-Scale Study About Quality and Reproducibility of Jupyter Notebooks”. In: *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. 2019, pp. 507–517. DOI: 10.1109/MSR.2019.00077.

[59] Piotr Przymus et al. “The Secret Life of CVEs”. In: *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. 2023, pp. 362–366. DOI: 10.1109/MSR59073.2023.00056.

[60] R. Rajesh. "An introduction to grey influence analysis (GINA): Applications to causal modelling in marketing and supply chain research". In: *Expert Systems with Applications* 212 (2023), p. 118816. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.118816. URL: https://www.sciencedirect.com/science/article/pii/S0957417422018346.

[61] Donald Rubin. "Estimating causal effects of treatments in randomized and nonrandomized studies". In: *Journal of Educational Psychology* 66 (Oct. 1974). DOI: 10.1037/h0037350.

[62] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering". In: 14.2 (Apr. 2009), pp. 131–164. ISSN: 1382-3256. DOI: 10.1007/s10664-008-9102-8. URL: https://doi.org/10.1007/s10664-008-9102-8.

[63] Simone Scalabrino et al. "Data-Driven Solutions to Detect API Compatibility Issues in Android: An Empirical Study". In: *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. 2019, pp. 288–298. DOI: 10.1109/MSR.2019.00055.

[64] Eric B. Schneider. "Collider bias in economic history research". In: *Explorations in Economic History* 78 (2020), p. 101356. ISSN: 0014-4983. DOI: https://doi.org/10.1016/j.eeh.2020.101356. URL: https://www.sciencedirect.com/science/article/pii/S0014498320300516.

[65] The Royal Swedish Academy of Science. "Answering causal questions using observational data". In: (Oct. 2021). URL: https://www.nobelprize.org/uploads/2021/10/advanced-economicsciencesprize2021.pdf.

[66] The Royal Swedish Academy of Science. "Understanding development and poverty allevation". In: (Oct. 2019). URL: https://www.nobelprize.org/uploads/2019/10/advanced-economicsciencesprize2019.pdf.

[67] Ian Shrier and Robert W. Platt. "Reducing Bias Through Directed Acyclic Graphs". In: *BMC Medical Research Methodology* 8 (2008), p. 70. DOI: 10.1186/1471-2288-8-70.

[68] Julien Siebert. "Applications of statistical causal inference in software engineering". In: *Information and Software Technology* 159 (2023), p. 107198. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2023.107198. URL: https://www.sciencedirect.com/science/article/pii/S0950584923000526.

[69] Oleg Sofrygin, Mark J. van der Laan, and Romain Neugebauer. "simcausal R Package: Conducting Transparent and Reproducible Simulation Studies of Causal Effect Estimation with Complex Longitudinal Data". In: *Journal of Statistical Software* 81.2 (2017), pp. 1–47. DOI: 10.18637/jss.v081.i02.

[70] Dani Stanbouly et al. "What is Collider Bias and Why Should We Care?" In: *Journal of Oral and Maxillofacial Surgery* 80.9 (2022), pp. 1463–1465. ISSN: 0278-2391. DOI: https://doi.org/10.1016/j.joms.2022.04.019. URL: https://www.sciencedirect.com/science/article/pii/S0278239122003494.

[71] Klaas-Jan Stol and Brian Fitzgerald. "The ABC of Software Engineering Research". In: *ACM Trans. Softw. Eng. Methodol.* 27.3 (Sept. 2018). ISSN: 1049-331X. DOI: 10.1145/3241743. URL: https://doi.org/10.1145/3241743.

[72] Weijie Sun et al. “An Empirical Study to Investigate Collaboration Among Developers in Open Source Software (OSS)”. In: *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. 2023, pp. 352–356. DOI: 10.1109/MSR59073.2023.00054.

[73] Alexey Svyatkovskiy et al. *Fast and Memory-Efficient Neural Code Completion*. 2021. arXiv: 2004.13651 [cs.SE].

[74] Johannes Textor et al. “Robust causal inference using directed acyclic graphs: the R package ’dagitty’”. In: *International Journal of Epidemiology* 45.6 (2016), pp. 1887–1894.

[75] Alexander Trautsch and Steffen Herbold. “The SmartSHARK Repository Mining Data”. In: *CoRR* abs/2102.11540 (2021). arXiv: 2102.11540. URL: https://arxiv.org/abs/2102.11540.

[76] Fabian Trautsch et al. “Adressing Problems with External Validity of Repository Mining Studies Through a Smart Data Platform”. In: *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*. 2016, pp. 97–108.

[77] Adam Tutko, Austin Z. Henley, and Audris Mockus. “How are Software Repositories Mined? A Systematic Literature Review of Workflows, Methodologies, Reproducibility, and Tools”. In: *ArXiv* abs/2204.08108 (2022). URL: https://api.semanticscholar.org/CorpusID:248228041.

[78] M. Vidoni. “A systematic process for Mining Software Repositories: Results from a systematic literature review”. In: *Information and Software Technology* 144 (2022), p. 106791. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2021.106791. URL: https://www.sciencedirect.com/science/article/pii/S0950584921002317.

[79] Matthew J. Vowels, Necati Cihan Camgoz, and Richard Bowden. “Dya Like DAGs? A Survey on Structure Learning and Causal Discovery”. In: *ACM Comput. Surv.* 55.4 (Nov. 2022). ISSN: 0360-0300. DOI: 10.1145/3527154. URL: https://doi.org/10.1145/3527154.

[80] Chao Wang, Zhenpeng Chen, and Minghui Zhou. “AutoML from Software Engineering Perspective: Landscapes and Challenges”. In: *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. 2023, pp. 39–51.

[81] Jinqiu Yang et al. “Towards Extracting Web API Specifications from Documentation”. In: *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)*. 2018, pp. 454–464.

A

Color Coded Spreadsheets for Sampling Process

This appendix contains two color coded spreadsheets, one for DPA and one for MC. In the tables, green indicates a fulfilled criterion, red indicates unfulfilled, and yellow indicates uncertainty about fulfillness. The shortenings used in the head rows have the following meanings:

- Y = Year
- T = Title
- RP = Replication Package
- ND = No DAGs
- SM = Statistical Model
- V = Variables
- SCD = Selection Criteria Data
- SCV = Selection Criteria Variables

A.1 Distinguised Paper Awards

Table A.1:

Y	T	RP	ND	SM	V	SCD	SCV
2023	AutoML from Software Engineering Perspective [80]	Y	Y	N	-	Y	N
2023	Investigating Vulnerable Dependencies with Dependabot [44]	Y	Y	Y	Y	Y	N
2023	The ABLoTS Approach for Bug Localization [47]	N	Y	-	-	-	-

Continued on next page

Table A.1: (Continued)

2022	A Large-Scale Comparison of Python Code in Jupyter Notebooks and Scripts [24]	Y	Y	N	-	Y	N
2022	On the Violation of Honesty in Mobile Apps: Automated Detection and Categories [49]	Y	Y	Y	Y	Y	Y
2022	Operationalizing Threats to MSR Studies by Simulation-Based Testing [26]	Y	Y	Y	Y	N	N
2021	What code is deliberately exclude from test coverage and why? [33]	N	Y	-	-	-	-
2021	Fast memory-efficient neural code completion [73]	N	Y	-	-	-	-
2021	Escaping the Time Pit: Pitfalls and Guidelines for using Time-based Git Data [73]	Y	Y	N	-	Y	N
2020	Traceability Support for Multi-Lingual Software Projects [39]	N	Y	-	-	-	-
2020	Ethical Mining - A Case Study on MSR Mining Challenges [21]	N	Y	-	-	-	-
2020	A Machine Learning Approach for Vulnerability Curation [11]	N	Y	-	-	-	-
2019	Standing on Shoulders or Feet? The Usage of the MSR Data Papers [37]	Y	Y	N	-	Y	N
2019	Data-Driven Solutions to Detect API Compatibility Issues in Android: An Empirical Study [63]	Y	Y	N	-	Y	N
2019	A Large-scale Study about Quality and Reproducibility of Jupyter Notebooks [58]	Y	Y	N	-	Y	N
2018	Prevalence of Confusing Code in Software Projects - Atoms of Confusion in the Wild [23]	Y	Y	Y	Y	Y	N
2018	Towards Extracting Web API Specifications from Documentation [81]	N	Y	-	-	-	-

Continued on next page

Table A.1: (Continued)

2017	Classifying code comments in Java open-source software systems [50]	Y	Y	Y	Y	Y	Y
2017	Some From Here, Some From There: Cross-Project Code Reuse in GitHub [20]	N	Y	-	-	-	-
2016	Adressing problems with external validity of repository mining studies through a smart data platform [76]	Y	Y	Y	Y/N	Y	N
2016	Studying the Impact of Switching to a Rapid Release Cycle on Integration Delay of Addressed Issues - An Empirical Study of the Mozilla Firefox Project [14]	N	Y	-	-	-	-
2015	Do Bugs Foreshadow Vulnerabilities? A Study of the Chromium Project [8]	N	Y	-	-	-	-
2015	Characterization and prediction of issue-related risks in software projects [12]	N	Y	-	-	-	-

A.2 Mining Challenge

Table A.2:

Y	T	RP	ND	SM	V	SCV
2023	An Empirical Study to Investigate Collaboration Among Developers in Open Source Software (OSS) [72]	Y	Y	Y	Y	Y
2023	Evolution of the Practice of Software Testing in Java Projects [34]	Y	Y	Y	N	Y
2023	Insights into Female Contributions in Open-Source Projects [10]	N	Y	-	-	-
2023	Keep the Ball Rolling: Analyzing Release Cadence in GitHub Projects [36]	Y	Y	Y	N	Y
2023	The Secret Life of CVEs [59]	Y	Y	Y	N	N
2022	An Exploratory Study on Refactoring Documentation in Issues Handling [2]	Y	Y	N	-	Y
2022	Between JIRA and GitHub: ASFBot and its Influence on Human Comments in Issue Trackers [43]	Y	Y	Y	Y	Y
2022	Is Refactoring Always a Good Egg? Exploring the Interconnection Between Bugs and Refactorings [4]	Y	Y	N	-	Y
2022	On the Co-Occurrence of Refactoring of Test and Source Code [46]	Y	Y	Y	Y	Y
2022	Refactoring Debt: Myth or Reality? An Exploratory Study on the Relationship Between Technical Debt and Refactoring [56]	Y	Y	Y	Y	Y
2022	Studying the Impact of Continuous Delivery Adoption on Bug-Fixing Time in Apache's Open-Source Projects [1]	N	Y	-	-	-
2022	Which bugs are missed in code reviews: An empirical study on SmartSHARK dataset [35]	Y	Y	N	-	Y

B

Motivation for Arrows

B.1 DAG outlining Q3: Project age influence on rate of atoms

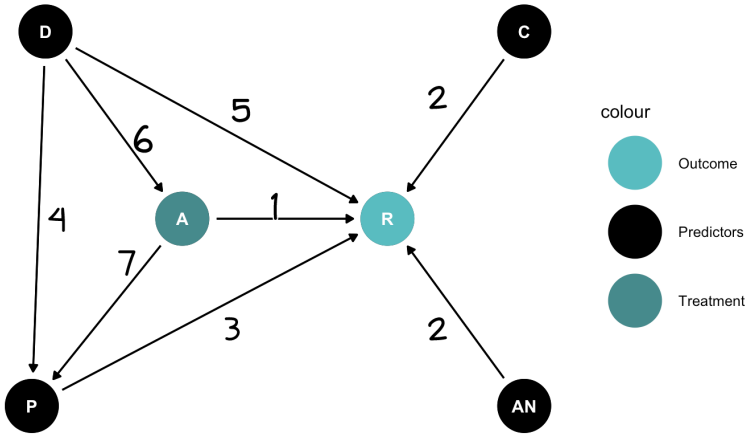


Figure B.1: DAG showcasing the system under study in Atoms of Confusion Q3. P = project ID, D = domain, A = date, R = rate, c = count, AN = all.nodes.

1. **date** → **rate**: As the research question under study was outlined such as "Does project age influence the rate of atoms?", it make sense to state project age as the exposure and atom rate as the outcome. In order to avoid confusion, the project age is defined as the project start date when doing the calculations and comparing the differences between projects. This can be seen in 5.6, where the Y-axis is "Average Atom Rate" and the X-axis is "Project Start Date". In the code (atom-finder/src/analysis/code_age.R line 59), it is clear since the date is used as the input for exposure and rate as the input for treatment. *rlm.model <- MASS::rlm(rate ~date, data=project.age.mean.atoms)*

2. **count** → **rate** & **all.nodes** → **rate**: The atom rate is a derived value from count and all.nodes, which is shown in the picture with the dubble arrows and surrounded by a box, as described by Gopstein et al. [23]. This calculation can be shown from the code in the repository (atom-finder/src/analysis/code_age.R line 28): *code.age.all.atoms[, rate := count / all.nodes]*.

3. **project ID → rate** : From the previous findings in this paper, the author could state "Similar projects tend to have similar atom usage rates, while unrelated projects use atoms differently"[23]. They argue that the usage of atoms between similar projects can depend on preferences of coding style, which lead to an encourage of certain atom types. Additionally, they can draw parallels in usage of atoms between projects with less similarity in nature, but with common authors and levels of neutrality.

4. **domain → project ID**: Related to the findings above, the authors also state that "... there is similarity between projects of the same domain" [23]. They can state that projects of the same domain seem to have a similar usage of atoms such as "... Type Conversion is very popular in compilers, but rarely used in version control systems" [23].

5. **domain → rate** : The authors could draw connections from the application domain directly to the atom rate. This was done when comparing the numbers of atoms in each project, were and the projects were colored by domain. Four out of seven domains were ordered next to each other, which could not be random, and state that ... application domain has a relationship to the usage of atoms of confusion [23].

6. **domain → date**: The research question itself discuss the potential in changes of usage in atoms of confusion over time, and point out that the surroundings can have an impact on how the coding is performed. The authors describe "The software industry has evolved dramatically in the past 30+ years since the oldest project in our corpus was born" [23]. Where they describe that some atoms were encouraged to use due to its efficiency.

7. **date → project ID**: Continuing from the earlier discussion regarding impact from surroundings, the authors elaborate on what could influence coding style within a project. They highlight "... that style is strongly influenced by the generation in which the project was started" [23], implying that the project's start date directly affects its overall structure and development.

B.2 DAG outlining Q4: Commit type influence on atom removal rate

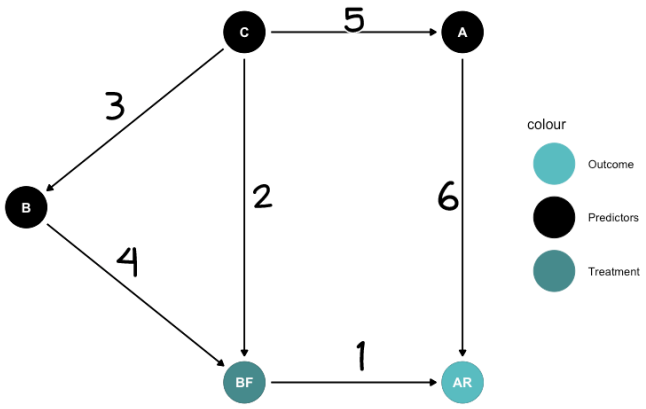


Figure B.2: An incomplete DAG of the system under study in Atoms of Confusion Q4. BF = bug-fix commit (TT + TF), AR = atom.removal, B = n.bugs, C= source.change, A = change in atoms (increase/ decrease).

1. **bug-fix commit (TT+TF) → atom.removal**: The question under study "Are atoms removed more often in bug-fix commit?"
2. **source.change → bug-fix commit (TT+TF)**: Change in characters, used as the variable *source.change*, represent the definition of a change in the code (in other words; a commit). The presence of a commit is crucial for a bug fix commit to happen.
3. **source.change → n.bugs**: In a bug-fix commit, the objective is to make changes in the code to remove bugs. The change of characters in code is a requirement for the number of bugs to change.
4. **n.bugs → bug-fix commit (TT+TF)**: When the number of bugs decrease in a commit, it is called a bug-fix commit. In the computation, the number of bugs resolved in a commit represent *n.bugs* which makes *is.bug=1* if *n.bugs* > *0*.
5. **source.change → change in atoms (increase/decrease)**: The removal or introducing of atoms is made by changes in the code.
6. **change in atoms (increase/decrease)→ atom.removal**: The change of atoms is described as *increased* or *decreased*, depending on the removal or introducing of atoms in a commit. This is later used in calculating the atom removal rate, together with the presence of a bug-fix commit.

B.3 DAG outlining Q5: Atoms influence on bugs

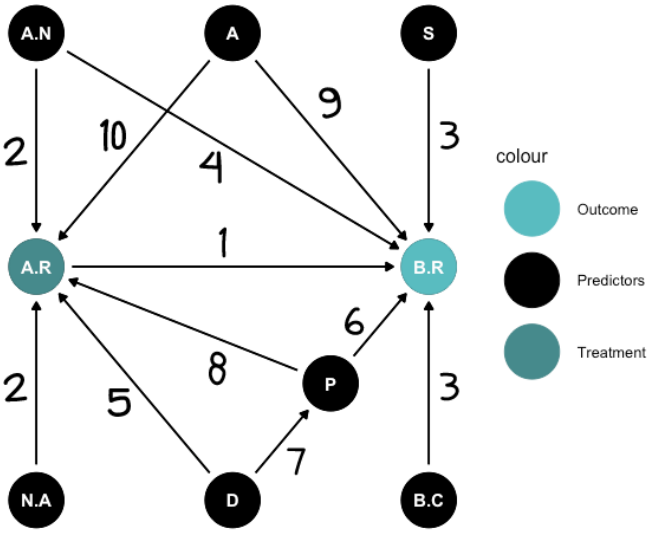


Figure B.3: A DAG showcasing the system under study in Atoms of Confusion Q5. A.R = atom.rate, A.N = all.nodes, N.A = non.atoms, D = domain, B.R = bug.rate, B.C = bug.count, S = since, P = project ID, A = atom ID

1. **atom.rate** → **bug.rate**: The question under study, investigating if projects with more atoms also have more bugs.
2. **all.nodes** → **atom.rate** & **non.atoms** → **atom.rate**: Atom rate is a derived value from following computation: *atom.rate:=(X.all.nodes-X.non.atoms)/X.all.nodes].* Where the size of the project and the amount of atoms effect the resulting atom rate.
3. **since** → **bug.rate** & **bug.count** → **bug.rate**: Bug rate is a derived value from *bug.rate := bug.count/as.numeric(as.Date("2018-01-01")-since).* Where the age of the project and the amount of bugs effect the resulting bug rate.
4. **all.nodes** → **bug.rate**: When plotting, the authors decides to use "Bugs per year per AST node (log). Meaning that the size of the project is taken into account when plotting the data point *y=365*bug.rate/X.all.nodes.*
5. **domain** → **atom.rate**: See explanation of same arrow (domain → rate) in DAG 1, Appendix B.1.
6. **project ID** → **bug.rate**: The authors describes this relation by stating Acknowledging that not every kind of project has the same attack surface (ways to become vulnerable) [23].
7. **domain** → **project ID**: The project's data is paired by domain. See explanation of same arrow (domain → project ID) in DAG 1, Appendix B.1.
8. **project ID** → **atom.rate**: See explanation of same arrow (project → rate) in DAG 1, Appendix B.1.

9. **atom ID** → **bug.rate**: The authors believes that small code patterns (atoms) contribute to misunderstanding among programmers, leading to the occurrence of bugs. In the study by Gopstein et al. [23], each atom type is associated with an effect size ranging from 0 to 1, indicating the degree to which the atom contributes to confusion compared to a transformed pair.

10. **atom ID** → **atom.rate** : From the first research question (Q1), it is proven that various atom types occur in different ranges. The authors suggest potential underlying causes, such as atoms being socially accepted, benefits that can be perceived, or certain atoms being well-known anti-patterns.

B.4 DAG outlining Q6: Atom rate influence on level of confusion

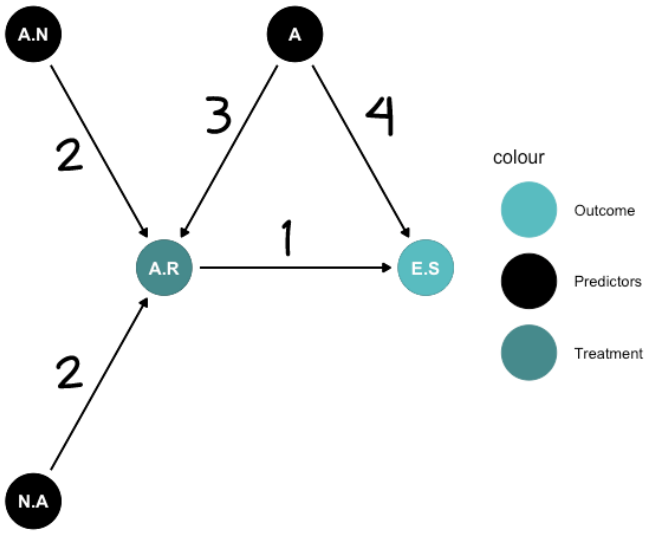


Figure B.4: A DAG showcasing the system under study in Atoms of Confusion Q6. A.R = all.atom.rates, A.N = all.nodes, N.A = non.atoms, E = effect.size, A = atom ID

1. **all.atom.rates** → **effect.size**: This relationship showcase the question under study, where Gopstein et al. [23] wanted to answer the question to; "Does prevalence correlate with amount of confusion?". They believe atoms that are less confusing occurs more often.

2. **all.nodes** → **all.atom.rates** & **non.atoms** → **all.atom.rates**: The atom rate (the atom's occurence) is a derived value from non.atoms and all.nodes.

3. **atom ID** → **all.atom.rates**: From the first research question (Q1), it is proven that various atom types occur in different ranges. The authors suggest potential underlying causes, such as atoms being socially accepted, benefits that can be perceived, or certain atoms being well-known anti-patterns.

4. **atom ID → effect.size**: Each atom type is associated with an effect size ranging from 0 to 1, indicating the degree to which the atom contributes to confusion compared to a transformed pair.

B.5 DAG outlining: The presence of the ASFBot on amount of comments

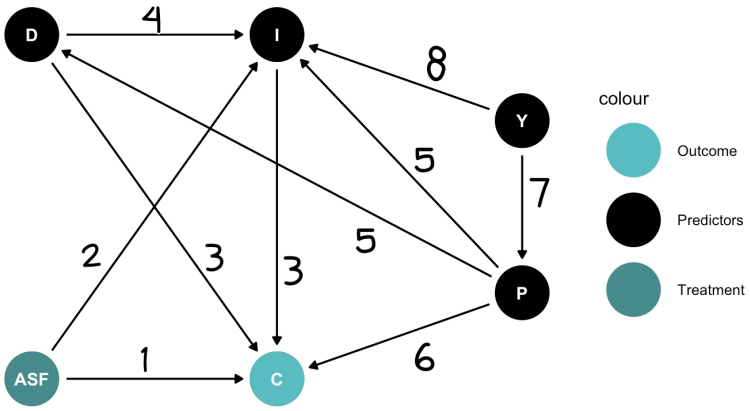


Figure B.5: DAG showcasing the research scenario including the ASFBot. ASF = ASFBot, D = Numbers of Developers, I = Number of Issues, Y = Project Age, C = Median Number of Comments, P = Project Name

1. **ASFBot → Median Number of Comments**: This arrow represents the investigation of whether the ASFBot impacts the number of comments made by humans in Apache projects.
2. **ASFBot → Number of Issues**: The authors explain how the ASFBot influences the number of issues: In this work, we find empirical support for the advantages provided by the ASF, and we find that developers can be more productive when the ASFBot is present as they no longer need to perform the repetitive task of manually updating issues in JIRA when merging a PR on GitHub [43]."
3. **Number of Issues → Median Number of Comments** & **Number of Developers → Median Number of Comments**: The authors conclude from the results that As expected, the number of monthly developers and number of monthly issues also explain the number of comments that mention PRs and fixes, i.e., more comments are produced when there are more developers and more issues to be discussed.[43]
4. **Number of Developers → Number of Issues**: More developers lead to more issues being reported, as issues are created by developers.
5. **Project Name → Number of Developers** & **Project Name → Number of Issues**: The study investigates nine different projects and suggests that variables (including number of developers and issues) should be computed

differently for each project. The paper states: "... are computed for each of these 9 projects and modeled into the mixed-effects linear regression. To account for variability originating from the fact that each project is unique, we modeled an additional variable called project_name, which represents the names of the nine projects as a random effect [43]".

6. **Project Name → Median Number of Comments**: The study investigates nine different projects and suggests that variables should be computed differently for each project. The paper states: "This means that there is a large difference between the projects [43]".
7. **Project Age → Project Name**: By domain knowledge and previous reasoning in other studies, it can be suggested that the style and structure of a project can be strongly influenced by the time at which the project was started.
8. **Project Age → Number of Issues**: Older projects may have a higher number of reported issues over time due to more technical debt and outdated code that needs attention, as well as increased complexity that makes the code harder to maintain.

B.6 DAG outlining: Source code refactoring influence on test code refactoring

No reasoning about the relationships between variables was provided in the paper by Nagy and Abdalkareem [46]. Therefore, the following motivations are based on the authors' domain knowledge and interpretation of the research.

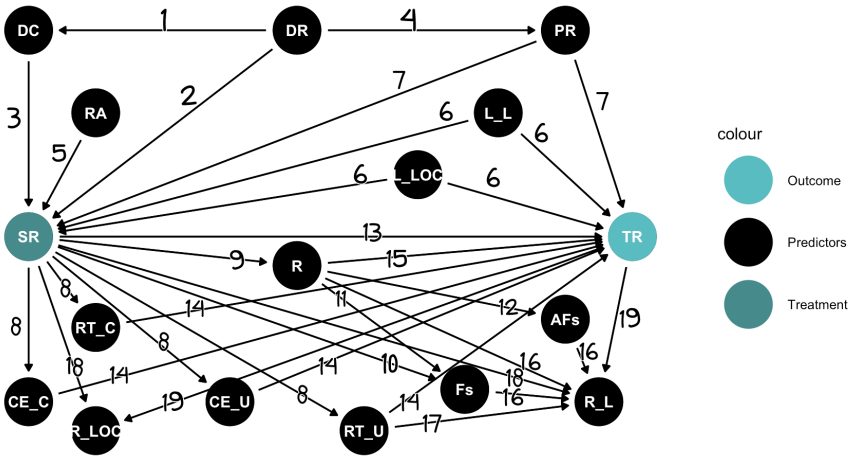


Figure B.6: DAG showcasing the direct effect source code refactoring has on test code refactoring. All predictors utilized in the papers analysis are therefore included. SR = Source Code Refactoring, TR = Test Code Refactoring, and the other shortenings for variables can be found in Table 5.11, .

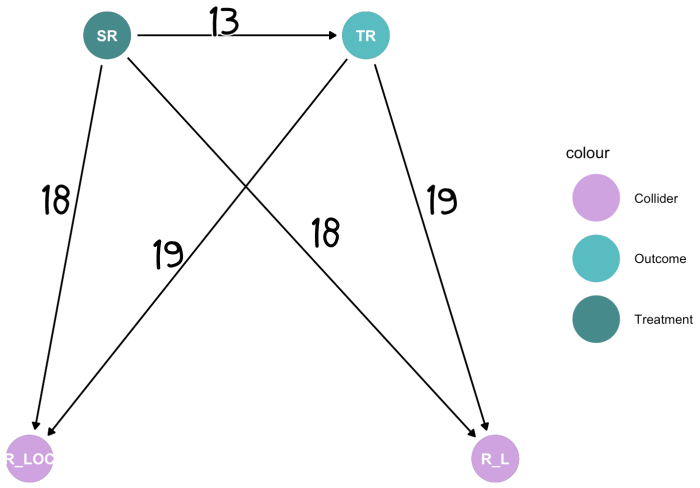


Figure B.7: DAG showcasing the total effect source code refactoring has on test code refactoring. SR = Source Code Refactoring, TR = Test Code Refactoring, R_LOC = LOC right side, R_L = # of right side locations

1. **DevRefExp** → **DevRefComExp**: There is no commit without refactorings. The more refactorings performed, the more likely there will be additional commits.
2. **DevRefExp** → **SR**: If a developer has great experience and familiarity with the codebase and has performed many refactorings in the past, they are more likely to recognize areas for improvement and proactively make changes. This increases the likelihood of additional source code refactorings. (Note: This feature is extracted from SR, not TR, so the reasoning does not apply to an arrow to TR.)
3. **DevRefComExp** → **SR**: A developer with significant refactoring commit experience is likely to prioritize best practices and codebase quality. They invest effort in maintaining the codebase, making it more possible to continue to refactor the codebase. (Note: This feature is extracted from SR, not TR, so the reasoning does not apply to an arrow to TR.)
4. **DevRefExp** → **PrevRef**: The more refactorings a developer has made previously, the higher the number of previous refactorings.
5. **RefAge** → **SR**: It is less likely for refactorings to occur as the number of days increases. This could imply either good quality or that certain files are more likely to be refactored than others due to their characteristics. Continuous refactoring of files leads to a low refactoring age and increases the likelihood of further refactorings.
6. **L_LOC** → **SR** & **L_Locations** → **SR** & **L_LOC** →, **TR** & **L_Locations** → **TR**: More extensive refactoring likely increases the possibility of introducing bugs and necessitating further refactorings in the source code. Similarly, extensive refactoring of source code may require corresponding refactorings in

the test code.

7. **PrevRef** → **SR** & **PrevRef** → **TR**: Previous refactoring work can influence the need for further refactorings in both source and test code. For example, extensive refactoring in source code may increase the likelihood of needing to refactor the connected test code.

8. **SR** → **CodeElement_C** & **SR** → **NmbCodeElem_UQ** & **SR** → **RefType_C** & **SR** → **NmbRefType_UQ**: Source code refactorings affect the count of both refactoring types and code element types. This is because any modification, addition, or deletion will impact these counts. Moreover, the more refactorings performed, the higher the likely count of both code elements and refactoring types.

9. **SR** → **NmbRef**: There is no increase in the number of refactorings without source code refactoring.

10. **SR** → **NmbFiles**: Source code refactoring directly impacts the number of files in which the refactorings are done.

11. **NmbRef** → **NmbFiles**: The number of files is based on the number of refactorings, with an increase in refactorings likely leading to more files being touched.

12. **NmbRef** → **AvgNmbFiles**: The average is calculated by dividing the number of files each refactoring touched by the total number of refactorings. Therefore, the number of refactorings is directly used in this computation. (Note: The total number of files does not indicate how many files are touched in each individual refactoring, even if the same files are being considered. Therefore, no arrow is drawn from NmbFiles to AvfNmbFiles)

13. **SR** → **TR**: This is the primary question under study.

14. **NmbRefType_UQ** → **TR** & **RefType_C** → **TR** & **NmbCodeElem_UQ** → **TR** & **CodeElement_C** → **TR**: The greater the variety of refactoring types or code elements, the higher the likelihood that corresponding tests will require refactoring due to the larger number different changes made in the source code.

15. **NmbRef** → **TR**: More refactorings mean more changes to the source code, increasing the probability that test code refactoring will be needed to reflect these changes.

16. **AvgNmbFiles** → **R_Locations** & **NmbRef** → **R_Locations** & **NmbFiles** → **R_Locations**: With multiple files touched per refactoring and several refactorings are performed, it can lead to more diverse changes in the codebase. Consequently, this might result in more locations being affected in the upcoming commit, potentially due to the need to fix bugs or address other issues.

17. **NmbRefType_UQ** → **R_Locations**: A greater variety of refactoring types

results in more diverse changes to the codebase, which most likely lead to necessary modifications across several locations in the child commit.

18. **SR** $\rightarrow$ **R_LOC**& **SR** $\rightarrow$ **R_Locations**: With extensive changes in the source code previously in a refactor, more lines could likely be targeted for further modifications in upcoming commits. For example, if the previous change included adding functionality in the source code and upcoming commits incorporate it. Alternatively, by refactoring code and making it more modular, organized, and efficient, fewer lines of code would perhaps be needed to modify in the child commit.

19. **TR** $\rightarrow$ **R_LOC** & **TR** $\rightarrow$ **R_Locations**: This relates to whether changes in tests would affect upcoming changes in the source code. There could be several aspects motivating why this would be the case, see the motivation below:

    (a) Refactoring of test code may lead to an **uncovering of bugs**, which would further influence the upcoming refactoring commit in the source code. Depending on where the refactoring is done, and what is discovered, more or less lines of code would probably be affected in the child commit. Or locations.

    (b) One common refactoring type is the **extraction of methods**. Test cases are also encouraged to be written as easily and concise as possible. If a refactoring would extract common functionality to reusable methods, they could be able to be reused in the source code additionally. If so, this could reduce upcoming source code changes where fewer lines are needed to be touched. The refactoring could also first reduce the number of lines in child commit, when common functionality is swapped to the reusable method. Depending on the extent of reusability of the method, more or less locations could be touched in the codebase.

    (c) **Test-Driven Development** (TDD), and refactorings done in this process, could directly influence the number of lines touched in the child commit. By this approach, changes in the test cases could lead to both an increased amount of lines needed to be touched in order for the tests to pass, or reduction. The test cases could also reflect structural changes, which would influence the number of locations touched by upcoming child commits.

    (d) Removing **unnecessary** or **obsolete test cases/methods** could influence the removal of certain amounts of dead- or unused code in the source code. This would influence the amount of LOC that would be reduced.

    (e) **Type of test** that is refactored. If a modification would be done to an integration test, rather than a unit test, and a run would expose necessary changes, it is more likely that more locations would be touched in the source code. This, compared to if the refactoring was done in a unit test.

C

Output of Estimates Retrieved

The output generated are all from the use of standardized data, and the numbers in the tables are on the logit scale. This due to the link function utilized in the statistical models. All of the models were run on a sample size of 8000.

1. Following results is retrieved from **theoretical example**, where the data is **simulated**, and the model does **not include colliders**:

	Estimate	**Est.Error**	**l-95% CI**	**u-95% CI**	**Rhat**	**Bulk_ ESS**	**Tail_ ESS**
α	70.10	0.11	69.90	70.31	1.00	23245	16346
β_{DE}	**0.92**	0.02	0.88	0.96	1.00	23114	15211
σ	3.90	0.03	3.84	3.96	1.00	22119	14972

2. Following results is retrieved from **theoretical example**, where the data is **simulated** from real-world scenario, and the model does **include colliders**:

	Estimate	**Est.Error**	**l-95% CI**	**u-95% CI**	**Rhat**	**Bulk_ ESS**	**Tail_ ESS**
α	68.86	0.12	68.63	69.10	1.00	16708	15628
β_{DE}	**1.06**	0.02	1.02	1.10	1.00	18071	15427
β_B	1.08	0.06	0.97	1.19	1.00	17986	15647
σ	3.82	0.03	3.76	3.88	1.00	19893	14888

3. Following results is retrieved from the **total effect**, where the data is **simulated**, and the model does **not include colliders**:

	Estimate	**Est.Error**	**l-95% CI**	**u-95% CI**	**Rhat**	**Bulk_ ESS**	**Tail_ ESS**
α	−1.75	0.10	−1.96	−1.56	1.00	16579	13579
β_{SR}	**0.82**	0.10	0.62	1.03	1.00	17296	13762

4. Following results is retrieved from the **total effect**, where the data is **simulated**, and the model does **include colliders**:

	Estimate	**Est.Error**	**l-95% CI**	**u-95% CI**	**Rhat**	**Bulk_ ESS**	**Tail_ ESS**
α	3.40	0.20	3.01	3.79	1.00	16383	15869
β_{SR}	**-6.51**	0.24	−6.99	−6.04	1.00	13738	13659
β_{R_LOC}	0.10	0.08	−0.05	0.27	1.00	17058	13660
$\beta_{R\_Locations}$	5.93	0.16	5.63	6.25	1.00	14010	14487

5. Following results is retrieved from the **direct effect**, where the data is **simulated**, and the model does **not include colliders**:

	Estimate	**Est.Error**	**l-95% CI**	**u-95% CI**	**Rhat**	**Bulk_ ESS**	**Tail_ ESS**
α	−1.42	0.10	−1.62	−1.23	1.00	44410	15234
β_{SR}	**0.52**	0.10	0.32	0.73	1.00	44296	15432
$\beta_{L\_Locations}$	0.09	0.03	0.05	0.14	1.00	46589	15463
β_{L_LOC}	−0.00	0.02	−0.05	0.05	1.00	48536	13969
$\beta_{NmbFiles}$	−0.02	0.03	−0.07	0.03	1.00	48043	14202
β_{NmbRef}	0.01	0.03	−0.05	0.06	1.00	36367	16072
$\beta_{NmbRefType\_UQ}$	0.00	0.03	−0.05	0.05	1.00	46922	14408
$\beta_{NmbCodeElem\_UQ}$	0.01	0.02	−0.04	0.05	1.00	47360	15603
$\beta_{AvgNmbFiles}$	0.01	0.03	−0.04	0.06	1.00	38214	16553
β_{RefAge}	0.01	0.03	−0.04	0.06	1.00	45858	13610
$\beta_{DevRefExp}$	0.01	0.02	−0.04	0.06	1.00	49707	14938
$\beta_{DevRefComExp}$	−0.04	0.02	−0.09	0.01	1.00	43410	14529
$\beta_{PrevRef}$	0.04	0.03	−0.01	0.09	1.00	47921	15244

6. Following results is retrieved from the **direct effect**, where the data is **simulated**, and the model does **include colliders**:

	Estimate	**Est.Error**	**l-95% CI**	**u-95% CI**	**Rhat**	**Bulk_ ESS**	**Tail_ ESS**
α	−1.34	0.12	−1.57	−1.10	1.00	35096	14809
β_{SR}	**0.13**	0.12	−0.10	0.37	1.00	37030	14314
$\beta_{L\_Locations}$	0.11	0.03	0.05	0.16	1.00	37962	13713
β_{L_LOC}	−0.02	0.03	−0.08	0.04	1.00	40552	14390
$\beta_{NmbFiles}$	0.01	0.03	−0.05	0.06	1.00	40905	14975
β_{NmbRef}	−2.25	0.07	−2.38	−2.12	1.00	17312	15934
$\beta_{NmbRefType\_UQ}$	−0.93	0.04	−1.00	−0.85	1.00	20706	16910
$\beta_{NmbCodeElem\_UQ}$	−0.02	0.03	−0.07	0.04	1.00	40383	14174
$\beta_{AvgNmbFiles}$	−0.24	0.03	−0.30	−0.18	1.00	36818	14956
β_{RefAge}	0.02	0.03	−0.04	0.08	1.00	43327	13934
$\beta_{DevRefExp}$	0.00	0.03	−0.06	0.06	1.00	39417	14401
$\beta_{DevRefComExp}$	−0.03	0.03	−0.08	0.03	1.00	44263	13907
$\beta_{PrevRef}$	0.04	0.03	−0.02	0.10	1.00	43169	14241
β_{R_LOC}	0.36	0.03	0.30	0.42	1.00	39988	15091
$\beta_{R\_Locations}$	3.05	0.08	2.89	3.22	1.00	16383	15423

7. Following results is retrieved from the **total effect**, where the data is **empirical**, and the model does **not include colliders**:

	Estimate	**Est.Error**	**l-95% CI**	**u-95% CI**	**Rhat**	**Bulk_ ESS**	**Tail_ ESS**
α	−0.29	0.05	−0.38	−0.20	1.00	17824	13864
β_{SR}	**-0.87**	0.04	−0.95	−0.78	1.00	16007	12301

8. Following results is retrieved from the **total effect**, where the data is **empirical**, and the model does **include colliders**:

	Estimate	**Est.Error**	**l-95% CI**	**u-95% CI**	**Rhat**	**Bulk_ ESS**	**Tail_ ESS**
α	−0.28	0.05	−0.37	−0.18	1.00	20884	14922
β_{SR}	**-0.89**	0.04	−0.97	−0.80	1.00	20150	15179
β_{R_LOC}	0.13	0.04	0.05	0.21	1.00	18316	15437
$\beta_{R\_Locations}$	0.29	0.03	0.23	0.36	1.00	18847	16561

D

Variable Names in Replication Packages

This appendix presents which feature names corresponds to what feature names in the original replication package of the studied paper. The variable names used in this study were defined to more clearly capture the descriptions of each feature made in the paper.

DAGs	This Study	Original CSV-file
R	NmbRef	refactoryCount
L_L	L_Locations	leftLocationCount
R_L	R_Locations	rightLocationCount
L_LOC	L_LOC	leftLocationDiff
R_LOC	R_LOC	rightLocationDiff
Fs	NmbFiles	fileTouchCount
AFs	AvgNmbFiles	fileTouchAverage
RT_U	NmbRefType_UQ	refactoryTypeCount
CE_U	NmbCodeElem_UQ	codeElementCount
PR	PrevRef	num_touched_before
RA	RefAge	age
DR	DevRefExp	dev_ref_exp
DC	DevRefComExp	dev_ref_com_exp