Mitigating omitted variable bias in empirical software engineering

Empirical Software Engineering (2026) 31:123
https://doi.org/10.1007/s10664-026-10851-1

Mitigating omitted variable bias in empirical software engineering

Carlo A. Furia[1] · **Richard Torkar**[2,3,4]

Received: 22 September 2025 / Accepted: 26 March 2026

Abstract
Omitted variable bias occurs when a statistical model leaves out variables that are relevant determinants of the studied effects. This results in the model attributing the missing variables' effect to some of the included variables—hence over- or under-estimating the latter's true effect. Omitted variable bias presents a significant threat to the validity of empirical research, particularly in non-experimental studies such as those common in empirical software engineering. This paper illustrates the impact of omitted variable bias on two illustrative examples in the software engineering domain, and uses them to present methods to investigate the possible presence of omitted variable bias, to estimate its impact, and to mitigate its drawbacks. The analysis techniques we present are based on causal structural models of the variables of interest, which provide a practical, intuitive summary of the key relations among variables. This paper demonstrates a sequence of analysis steps that inform the design and execution of similar empirical studies in software engineering. An important observation is that it pays off to invest effort investigating omitted variable bias *before* actually executing an empirical study, because this effort can lead to a more solid study design, and to a reduction in its threats to validity.

Keywords Empirical software engineering · Confounders · Omitted variable bias · Sensitivity analysis · Simulation

Communicated by: Klaas-Jan Stol.

✉ Carlo A. Furia
furiac@usi.ch

✉ Richard Torkar
Richard.Torkar@cse.gu.se

[1] Software Institute, USI Universitá della Svizzera italiana, Via G. Buffi 13 CH-6900, Lugano, Switzerland

[2] University of Gothenburg, Chalmersplaten 4 SE-41296, Gothenburg, Sweden

[3] Chalmers University of Technology and University of Gothenburg, Gothenburg, Sweden

[4] Stellenbosch Institute for Advanced Study (STIAS), Stellenbosch, South Africa

Published online: 21 April 2026

1 Introduction

In this day and era of big data analytics, where massive datasets are commonly available with scalable machine learning techniques to boot, it is tempting to believe that "more data" is a cure-all. Unfortunately, there are scenarios where training with more data cannot improve the quality of a statistical analysis—in fact, it may even *worsen* it.

Consider a common task of any empirical discipline: estimating the (average) *effect* that changing a variable X has on another, dependent variable Y. In experimental study design parlance, X is the *treatment*[1] or *exposure*, and Y is the *outcome* or *response*. Here is a concrete example in the domain of software engineering, which we will look into more closely in Section 3: estimating how using a different programming language (treatment X) affects the quality (outcome Y) of a program implemented in that language. If we have observational data about X and Y, we may simply fit a statistical model—anything from a simple linear regression to fancier devices—on these data, and interpret the fitted model's parameters connecting X and Y as an estimate of this effect. In this process, a well-known snag is *omitted variable* bias: if there exists another *unmeasured* variable Z that affects both X and Y, our estimate of the effect $X \rightarrow Y$ will spuriously include the combined effect of X and Z instead of the effect of X alone. Continuing our empirical software engineering example, Z may be a programmer's intrinsic skills, which are all but sure to also affect the quality Y of the written programs. In such a scenario, additional data about X and Y (i.e., sampling more datapoints) is not going to help; in fact, it may just entrench our reliance on the biased estimate by reducing its variance and giving the false impression of reliability or "significance".

Omitted variable bias is a widespread risk of any statistical analysis of observational data, regardless of whether one employs frequentist (Holt et al. 2014; Gren et al. 2017; Torkar et al. 2022; Levén et al. 2024; Penzenstadler et al. 2022; Furia et al. 2022), or other kinds of machine learning models (Navarro et al. 2021; Mehrabi et al. 2021). In fact, it is always possible that *some* relevant variable was not measured, because it was unknown, inaccessible, or impractical, time-consuming, or expensive to measure with reasonable accuracy. This paper's main contribution is presenting several *mitigation strategies* to cope with omitted variable bias, and demonstrating them in scenarios and examples that are relevant for software engineering empirical data analysis.

Of course, the ideal approach to avoid omitted variable bias is running a fully *controlled experiment*, where the treatment X is assigned randomly, and the corresponding values of Y are recorded. Controlled experiments are the gold standard in science precisely because they protect from omitted variable bias even when we don't even know which other unmeasured variables may bias our estimate—since randomly assigning the treatment effectively removes its dependencies on any (unmeasured) confounder. The obvious reason why controlled experiments are not more common is that they are generally very expensive to run. In a field like empirical software engineering, proper controlled experiments are prohibitively challenging to design and run on the time scale of real-world software development—as

[1] The term *treatment* comes from the practice of randomized controlled clinical trials, where it strictly denotes a binary variable (treatment vs. control group) that is randomly assigned. In statistical parlance, however, the term is routinely applied more generally to indicate any variable whose effect on an outcome is being analyzed. Under this general meaning, a treatment variable may not be randomly assigned (Holland 1986) (for example, in an observational study) or it may be continuous (Hirano and Imbens 2004) instead of binary. In this paper, we use "treatment" with this relaxed, more general meaning—as a synonym of "exposure".

opposed to "toy" programming tasks. In contrast, there is abundant observational data from software repositories that span large systems developed over years by many developers; but discovering genuine causal effects using these purely observational data must contend with omitted variable bias. Therefore, this paper focuses on mitigation strategies for omitted variable bias when analyzing observational data.

Omitted variable bias is a common and relevant problem—especially for empirical software data. However, it is by no means the only pitfall of analyzing empirical data: as we further discuss in Section 2, there are plenty of other challenges such as the included variable bias (Furia et al. 2023), "precisely inaccurate" analyses that hide biases (McFarland and McFarland 2015), and unrepresentative population samples (Bradley et al. 2021). While each challenge requires different measures, they all likely involve trade-offs between costs and benefits—similarly to the present paper's outlook on dealing with omitted variable bias.

As we discuss more broadly in Section 2.1, there has been a growing interest in adopting robust, modern statistical practices in empirical software engineering research—especially those based on causal analysis. This paper contributes to this line of research, focusing on the practical and widespread problem of omitted variable bias.

1.1 Contributions

This paper makes the following contributions:

- Demonstrates the significance of omitted variable bias when analyzing empirical software engineering (observational) data;
- Presents statistical methods to detect, quantify, and mitigate omitted variable bias when analyzing empirical data;
- Provides simple guidelines for empirical software engineering researchers to apply those mitigation strategies in practical settings;
- For reproducibility, all data and analysis scripts are available online: REPLICATION PACKAGE: https://figshare.com/s/fe607d8eb7c4cedbac75 Furia and Torkar (2025)

The main aims of this paper are *methodological*: the illustrative examples of Section 3 and Section 4 are based on real data, but are not meant to lead to novel findings about their domains. Instead, they demonstrate plausible scenarios where confounding may occur, and explore different mitigation strategies.

1.2 Scope and Limitations

This paper focuses on a specific, yet widespread, challenge in empirical software engineering: estimating and mitigating omitted variable bias when analyzing observational data. The techniques we illustrate use structural causal models (DAGs, or directed acyclic graphs) to model the possible underlying causal relations between variables; adjustment sets to determine which variables should be included in a statistical model to make its inferences consistent with the causal relations; and sensitivity analysis to systematically explore the robustness and validity of the inferred results relative to a range of causal assumptions. The methods we present are applicable both *a posteriori*—to analyze non-experimental data—

and *a priori*—to inform the design of observational or quasi-experimental studies and to proactively outline the boundary of their validity.

These methods provide an accessible and practical framework for dealing with a common source of confounding in observational studies. However, they do not cover all confounding problems and are not suitable for every kind of empirical study design:

- We focus on recovering *causal* effects, as opposed to optimizing predictive performance or model succinctness in a purely statistical setting; in related work, we demonstrated scenarios where these goals lead to different results (Furia et al. 2023).
- The methods we present in the paper are well-suited for *cross-sectional* studies, where data from a population is collected at a specific point in time. These studies are very common in empirical software engineering. For comparison, Section 2.5 outlines a few approaches that extend confounding analysis to time-dependent data, which arise in longitudinal or panel studies.
- Omitted variable bias is only one of many different threats to the validity of empirical studies. Different kinds of confounding may require different statistical approaches (McElreath 2020); more generally, other sources of bias (selection bias, measurement bias, ...) may jeopardize a study's design validity at different levels (Wohlin et al. 2012).
- The DAG-based methods we consider in this paper have been gaining traction, as they support rigorous, yet intuitive foundations for causal modeling, and they are sufficiently general to accommodate a wide range of applications. However, other causal analysis frameworks exist—such as difference-in-differences, instrumental variables, regression discontinuity designs, matching methods, and synthetic control—that may be more suitable in certain contexts. Without going into details, a key difference of these approaches to causality is that they usually address confounding through preprocessing, design, or data structure (for example, relying on timing or instruments) rather than by modeling and adjusting within a statistical model as in DAG-based approaches (Stuart 2010). A more detailed presentation of these alternatives is outside this paper's scope, and is available elsewhere (Imbens and Wooldridge 2009; Athey and Imbens 2017).
- The DAG-based methods we demonstrate in the paper rely on several technical assumptions including: the absence of causal loops (acyclicity of the DAG), probabilistic relations between variables (as opposed to purely deterministic relations, which can only be modeled indirectly (Berrie et al. 2025)), and enough evidence to inform a plausible causal structural model. Section 2 presents the underlying methods in detail, compares them to other, related approaches, and outlines their relevance for empirical software engineering research.

1.3 Organization

Section 2 first presents relevant related work on the topics of mitigating biases in statistical analysis and, more generally, best statistical practices for empirical software engineering; then, it introduces the key notations and concepts that the rest of the paper relies on. The rest of the paper demonstrates how to deal with omitted variable bias in two illustrative examples: Section 3 uses a comparatively simpler model to estimate the effect of programming languages on code quality (also mentioned in the introduction), which serves as a relatively uncomplicated scenario. Then, Section 4 uses a more complex model to analyze the

relation between team size (how many developers work on a project) and effort (how long it takes to complete the project). Although the two illustrative examples differ in complexity, they are both based on realistic scenarios and models that we investigated in our previous work (Furia et al. 2022; Feldt et al. 2025). Section 5 serves as a high-level summary of the whole article, presenting *guidelines*, in the form of a sequence of analysis steps that generate fundamental information to support the design of a study that mitigates omitted variable bias. Finally, Section 6 concludes the paper with a short summary of the main contributions.

2 Related Work and Background

This section starts (in Section 2.1) with a brief overview of the origins of causal analysis techniques for observational data, and how they have been adopted in empirical sciences (including software engineering). Then, Section 2.2 and Section 2.3 introduce the key concepts (causal relations, DAGs, confounders) of causal analysis that we will develop in the rest of the paper. Finally, Section 2.4 positions the paper's contributions by relating them to other forms of confounding and causal inference bias.

2.1 Background

One of science's ultimate goals is understanding the processes that underlie observed phenomena. This means discovering *cause/effect* relations between variables, as opposed to mere statistical *associations*. While the roots of causal analysis date back to Neyman's potential outcomes framework (Splawa-Neyman et al. 1990), a comprehensive understanding of causality has emerged only later, in the late part of the 20th century. The key milestones in the development of a robust understanding of causality include: *i*) Rubin (1974) built upon Neyman's pioneering work, introducing a framework for causality in nonrandomized (observational) studies. *ii*) Angrist et al. (1996) introduced instrumental variables[2] based on the Neyman-Rubin framework. *iii*) Working at about the same time as Rubin, Pearl started focusing on structural (graph) models (Pearl 1982), which culminated in his celebrated techniques for rigorously analyzing causal effects and confounding (Pearl 2009).

Among Pearl's work, directed acyclic graphs (DAGs) have become widely used to model the structural dependencies between observed (and unobserved) variables. As we will demonstrate already with the simple example of Section 2.2, DAGs are, first of all, a practical notation to specify causal relations.[3] They also support techniques to *estimate* the strength of the causal relation among some nodes in the graph. Usually this is done by constructing a (linear) statistical model among variables, selected according to the DAG's structure. (We will demonstrate this shortly in Section 2.3.) Nowadays, causal analysis based on DAGs is routinely used in disciplines with a strong empirical component such as

[2] In a nutshell, an instrumental variable is a variable that acts like a natural experiment on the treatment.

[3] An alternative approach to modeling causal relations is structural equation modeling (SEM). Semantically, Pearl's causal DAG framework and SEM with latent variables are closely related. Specifically, Pearl's original work built on SEM, and modern presentations of the causal DAG framework explicitly connect it to SEM (Kenneth et al. 2013). Methodologically, however, the two approaches differ in their focus: causal DAGs (especially as we use them in the paper) specialize in modeling structural causal relations among (observed or latent) variables; SEM, instead, tends to mix measurement aspects (e.g., the relation between latent constructs and indicators) and structural ones (e.g., the relation between different constructs).

medicine (Williams et al. 2018; Peter et al. 2021), epidemiology (Greenland et al. 1999), economics (Imbens 2020), and biology (Laubach et al. 2021).

A key question when working with DAGs is how to build a realistic DAG in the first place. In fields where the underlying fundamental mechanisms are well understood, a DAG can be built based on expert knowledge and previous work. Another approach is *causal discovery* (also called structural learning), which tries to identify causal relations from data (Spirtes and Zhang 2016). While causal discovery algorithms have made significant progress in the last decade (Zanga et al. 2022)—in part on the wave of the recent machine learning boom—they remain *heuristic* approaches that work correctly only under precise assumptions about the possible interactions. This limitation is intrinsic, as one of the fundamental results of Pearl's framework is that causal relations cannot be inferred from data alone (at least not without limiting assumptions). Regardless of whether they come from expert knowledge, are inferred heuristically from data, or simply encode some (plausible) hypotheses, DAGs remain a practical tool to precisely denote, validate, and reason about the causal relations in a system.

While causal analysis techniques are not widely used in software engineering empirical research, they are gradually gaining traction. Siebert (2022)'s recent survey reports 31 studies in empirical software engineering that targeted some kind of causal analysis technique. All of the reviewed papers were published in the last 15 years, which confirms that causal analysis is not yet an established practice but is slowly gaining popularity. The majority (17 out of 31) of papers reviewed by Siebert (2022) are about fault localization and refer to George et al. (2010)—the first contribution that tried to apply a causal view instead of the traditional, purely correlational analysis that is commonplace in fault localization techniques (Wong et al. 2016; Zou et al. 2021; Rezaalipour and Furia 2024). Testing is another area targeted by several of the studies reviewed by Siebert (2022); these include applications to mutation testing (Lee et al. 2021), simulation testing (Andrew et al. 2022), and A/B testing (Liu et al. 2022). The other papers reviewed by Siebert (2022) target various topics such as performance analysis (in one case still linked to fault localization (Scholz and Torkar 2021)).

In the last few years, some more empirical software engineering research was published, targeting varied topics such as: *i*) modeling rules of human knowledge and how they are made available to artificial intelligence systems (Hans-Martin Heyn and Knauss 2022); *ii*) studying the impact of social media posts on the popularity of open-source projects (Fang et al. 2022); *iii*) analyzing dependencies in configurable software systems (Halpern 2015); *iv*) studying the impact of programming languages on coding competitions (Furia et al. 2023).

More broadly, these techniques belong to a broad and growing body of work on modernizing statistical practices (McElreath 2020). This work has been the response to the realization that some "traditional" and widely used statistical methods have limitations in terms of interpretability and real-world significance (Ronald et al. 2016; Amrehin et al. 2019; McShane et al. 2019). While improving statistical practices is only a piece of the puzzle, inadequate methods have certainly concurred to the issues with replicability experienced in empirical research areas as diverse as medicine (John and Ioannidis 2005), psychology (Open Science Collaboration 2015), and economics (Camerer et al. 2016). Since empirical software engineering research shares challenges and practices with these domains, adopting more robust data analysis practices will benefit the quality and impact of our research field (González-

Barahona and Robles 2012; Krishnamurthi and Vitek 2015; Madeyski and Kitchenham 2017; Menzies and Shepperd 2019; Jørgensen et al. 2016).

In conclusion, there is growing interest in understanding the concepts of causal analysis and applying them to analyzing software engineering data. The present paper further supports this trend by demonstrating how the framework of causal analysis can help mitigate the pervasive issue of omitted variable bias.

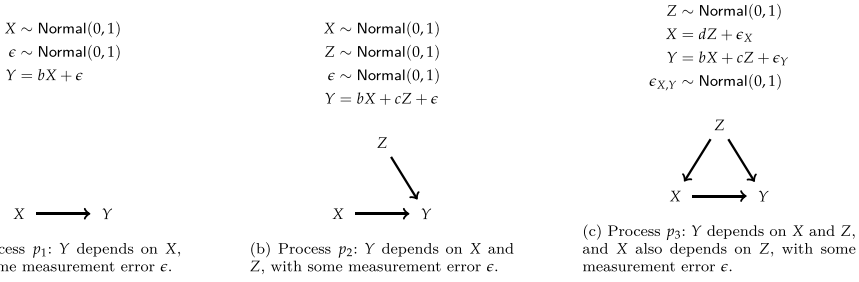
2.2 Causal Dependencies and DAGs

Let's go back to the example of two observed variables X and Y, which we briefly introduced in Section 1. Imagine that the process that determines the values of X and Y is perfectly known. In turn, we consider each of the three processes described by the equations in Fig. 1a. For simplicity, all our examples use *linear* dependencies and normal distributions, but the same line of thought is applicable to more complex, non-linear dependencies.

Process p_1 in Fig. 1a produces values of X that are drawn randomly from a normal distribution with zero mean and unit standard deviation; and values of Y that are linearly proportional to those of X. Even in such an ideal scenario, any empirical *measure* of X and Y will include some measurement error ϵ, which process p_1 models as another normal random variable that, together with X, determines the value of Y. We stress that we interpret Fig. 1a's equations as capturing the causal dependencies among variables: X and ϵ are drawn randomly (and independent of each other), and their random values determine ("cause") the value of Y in each draw. Correspondingly, the DAG (directed acyclic graph (Pearl 2009)) in Fig. 1a captures this causal relation between X and Y in a qualitative way: an edge connects X to Y to denote that the values of X and Y are related; furthermore, the edge is directed from X to Y to denote that changing X directly affects Y, that is, it causes Y to change.

Process p_2 in Fig. 1b involves a third variable Z. Just like variable X, Z's values are drawn randomly from a normal distribution with zero mean and unit standard deviation. Then, the values of Y are a linear combination of X and Z—still with a term ϵ to account for measurement errors. Variables X and Z are independent of each other; the DAG in Fig. 1b clearly shows this independence, since it does not include any edge between X and Z.

Process p_3 in Fig. 1c still involves the three variable X, Y, and Z. Now, Z is the only variable that is drawn independently; in contrast, X depends linearly on Z, and Y depends linearly on a combination of X and Z. As usual, the DAG in Fig. 1c visualizes these relations



(a) Process p_1: Y depends on X, with some measurement error ϵ.

(b) Process p_2: Y depends on X and Z, with some measurement error ϵ.

(c) Process p_3: Y depends on X and Z, and X also depends on Z, with some measurement error ϵ.

Fig. 1 Three possible processes where variable Y depends on variables X and Z, and the corresponding DAGs capturing these structural relations

among variables, showing, in particular, that there is both a direct relation between X and Y (edge $X \to Y$) and an indirect relation through Z (path $\underbrace{X \leftarrow Z \rightarrow Y}$).

2.3 Inference and Confounders

Consider an empirical study whose goal is to *estimate*, from a sample of the data, the parameters of a statistical model that captures the relations among observed variables. Let's shift perspective on our illustrative examples: now, we are given a data sample D_1, D_2, D_3 respectively produced by each process p_1, p_2, p_3. Each data sample consists of many triples (x_i, y_i, z_i) of concrete values taken by X, Y, and Z[4] in the ith observation—that is, the process's ith draw. Now, let's assume that we do not know the equations governing the generation process, but we want to quantitatively estimate the relation between X and Y from the observed sample.

Given that we are dealing with linear relations and normal distributions, we will use a *regression model* to fit the data. A key choice is whether to include variable Z in the model: regression model m_1, shown in Fig. 2a, ignores Z, whereas model m_2, shown in Fig. 2b, includes Z as a predictor. In both cases, after fitting a model on the data, the value of parameter β will be the model's estimate of the corresponding b in Fig. 1—in other words, β estimates the causal "effect" of the predictor X on the outcome Y. Table 1 shows the result of this exercise, when Fig. 1's processes use concrete values $b = 0.4$, $c = 0.7$, and $d = 0.2$ for their parameters.

Process p_1 The case of process p_1 is unproblematic: since the process does not involve any variable other than X and Y, regression model m_1 accurately infers the value of parameter $\beta \simeq 0.40 = b$, reflecting the true dependency between X and Y. The fitted regression model also accurately infers the standard deviation $\sigma = 1.00$ of the error term ϵ. Obviously, model m_2 is inapplicable to analyze data produced by p_1, since this includes no variable Z.

$$Y_i \sim \mathsf{Normal}(\mu_i, \sigma)$$
$$\mu_i = \alpha + \beta X_i$$

(a) Model m_1: Y is conditioned on X only.

$$Y_i \sim \mathsf{Normal}(\mu_i, \sigma)$$
$$\mu_i = \alpha + \beta X_i + \gamma Z_i$$

(b) Model m_2: Y is conditioned on both X and Z.

Fig. 2 Two linear regression models that capture the dependence between X, Y, and Z

Table 1 Estimating the parameters of Fig. 1's processes p_1, p_2, and p_3 with Fig. 2's regression models m_1 and m_2

	PROCESS		
MODEL	p_1	p_2	p_3
m_1	✔	✔	✖
m_2		✔	✔

(a) Whether each MODEL m estimates correctly (✔) or incorrectly (✖) the effect b of X on Y in PROCESS p.

	β			γ			σ		
MODEL	p_1	p_2	p_3	p_1	p_2	p_3	p_1	p_2	p_3
m_1	0.40	0.40	0.54				1.00	1.22	1.22
m_2		0.40	0.40		0.70	0.70		1.00	1.00

(b) For each process p_1, p_2, p_3 the values of parameters β, γ, σ in MODEL m_1 or m_2 fitted on data generated by the process. In these experiments, the parameters of Fig. 1's processes are set to $b = 0.4$, $c = 0.7$, $d = 0.2$.

[4] Process p_1's data sample only consists of pairs, since this process does not include a variable Z.

Process p_2 The case of process p_2 is more interesting, since we may analyze its data using either model m_1 or model m_2. As we expect, regression model m_2 accurately infers the value of parameter $\beta = b = 0.40$, again reflecting the true dependency between *X* and *Y*. Somewhat less obviously, the simpler model m_1 still infers the same correct value of parameter $\beta = b = 0.40$, even if *Y* also depends on *Z* in generation process p_2. With model m_1, the effect of *Z* on *Y* has spilled into the estimate of the standard deviation σ of the error term, which is in fact equal to 1.22, greater than the "true" error standard deviation 1.0.[5]

Process p_3 Regression model m_1 cannot accurately account for the more intricate data dependencies of process p_3. In fact, it overestimates the effect $\beta = 0.54 > 0.40 = b$ of *X* on *Y*; now, this effect also includes the "spurious" correlation introduced by *Z*, which simultaneously affects *X* and *Y*—as Fig. 1c's DAG clearly shows. Variable *Z* is called a *confounder*, since it mixes up the true effect of *X* on *Y* (path $X \rightarrow Y$ in the DAG) with an indirect, spurious correlation (path $X \leftarrow Z \rightarrow Y$ in the DAG) that does not correspond to an actual data dependency but is just a figment of using an inadequate statistical model. Model m_2 makes up for m_1's shortcomings by including *Z* among its predictors; this is enough to cancel *Z*'s confounding effect on the link between *X* and *Y*. Indeed, Table 1 shows that all parameters—crucially, the effect $\beta = 0.40 = b$ of *X* on *Y*—are correctly estimated.

Using a model such as m_1 to analyze data from a process whose dependencies include a confounder (like process p_3) is an instance of *omitted variable bias*. The rest of the paper describes more realistic illustrative examples where omitted variable bias may occur, and presents various mitigation strategies to counter the bias and recover a precise estimate of the causal effect linking treatment *X* and outcome *Y*.

From the toy examples of this section, we can start to glean how omitted variable bias is commonplace in realistic settings. Even when our empirical data are rich and include many different variables, there is always a chance that we are missing some *other* variables that, like *Z*, confounds the effect of interest. Even if we are aware of possible confounders, measuring them to include them in our model (like model m_2 does) may be expensive, impractical, or impossible. For example, the confounder may lack a good operationalization, or it may be inaccessible because its values were not recorded and the process is not repeatable. These observations motivate the main contributions of the paper, which demonstrate how to identify and mitigate the wicked effects of confounders in a variety of practical scenarios.

Correction and sensitivity analysis Broadly speaking, the rest of the paper illustrates two complementary approaches to deal with omitted variable bias when analyzing observational data. First, we will present techniques that can *correct* for possible omitted variable bias. Based on structural models of causality (see Fig. 1's DAGs), such techniques suggest how to model the data in a way that filters out confounding effects and recovers more precisely the strength of actual causal relationships.

As we will see in our examples, correcting for possible confounders is not always possible or practical. On the one hand, some confounding variables may simply be inaccessible;

[5] Equivalently, we can rewrite Fig. 1b's generative equations as $Y = bX + E$, where $E \sim \text{Normal}(0, \sqrt{1+c^2})$; if $c = 0.7$, $\sqrt{1+c^2} \simeq 1.22$, which is exactly the inferred value of σ in m_1 fitted on data from p_2.

on the other hand, the model of causality that underlies the application of adjustment techniques may itself be imprecise or uncertain. In such cases, a *sensitivity analysis* can mitigate uncertainty and clarify the boundary of validity of our findings. In a nutshell, a sensitivity analysis is a sort of "what if" analysis that quantifies the robustness of the results under different assumptions about the underlying process.[6] A sensitivity analysis can corroborate results and expose limits of validity, but is not expected to give any kind of definitive, binary answer.

2.4 Other Forms of Confounding

The term "confounding" is used to denote different, related concepts in the statistical analysis of empirical data (Greenland and Morgenstern 2001). In this paper, we use "confounding" to denote bias in the estimate of a *causal effect*—as we demonstrated in a nutshell in the previous sections. This notion of confounding is customary in modern causal analysis (Pearl 2009), and in related approaches to mitigate confounding, such as instrumental variables (Angrist et al. 1996).

An early usage of "confounding" in statistics was to denote *noncollapsibility* (Greenland and Morgenstern 2001; Undy Yule 1903). In a nutshell, the effect of X over Y is *collapsible over* Z if the marginal association between X and Y (obtained by averaging over Z) is the same as their conditional association (obtained by conditioning on Z); noncollapsibility holds when the two associations differ. While noncollapsibility is a purely statistical notion, the notion of confounding is rooted in causal relations, which depend not only on data but also on the data generation process (Pearl 2009); therefore, the two concepts overlap but are distinct. In particular, noncollapsibility may signal causal confounding (for example, in a scenario like Fig. 1c); but it may also arise from nonlinearities without confounding. For example, consider a scenario with the same causal structure $X \rightarrow Y \leftarrow Z$ as in Fig. 1b but where Y has a non-linear logistic relation with the linear combination of X and Z. Since the causal structure is the same as in Fig. 1b there is no confounding; however, the effect of X on Y depends on whether we condition on Z, and hence there is noncollapsibility.

In classical frequentist statistics, the term "confounding" denotes an experimental design that makes two effects indistinguishable from data (Fisher 1935). This usage of the term refers to statistical identifiability, and does not have, in general, any relation to causal effects. For example, a so-called *fractional* factorial design omits some combinations of factor levels, which may introduce confounding in this sense.

In fields such as psychometrics, confounding is often characterized as a measurement problem or, more generally, an issue of *experimental design* (Cook and Campbell 1979). In the present paper, in contrast, we take the main point of view of analyzing observational data, having little or no control on the process that generated the data.

While all these senses of the term "confounding" are related, the causal meaning that we follow in this paper captures a key challenge the analysis of observational data, and provides the clearest characterization of the omitted variable bias.

[6] It is important to notice that, despite their superficial similarities, the practice of sensitivity analysis and the malpractice of data dredging (also known as "*p*-hacking") have opposite goals: "In *p*-hacking, many justifiable analyses are tried, and the one that attains statistical significance is reported. In sensitivity analysis, many justifiable analyses are tried, and all of them are described." (McElreath 2020, p. 319).

2.5 Time-dependent Data

As we explained in Section 1.2, the techniques presented in this paper are not directly applicable to analyze confounding in *time-dependent* data, such as those that are collected in longitudinal studies. This section is a brief detour into approaches that take time into account.

As a concrete example, consider studying the impact of code reviews on the quality of a software project. A longitudinal study would consider a series of interventions $R_1, R_2, \ldots, R_n$, where each R_k denotes whether release k of a project was ($R_k = 1$) or wasn't ($R_k = 0$) reviewed before being merged into the main branch. The study's goal is to determine the cumulative effect of the interventions R_k on the project quality Q at release n. Covariates, possibly confounders, may also be time dependent, such as for the *experience* E_k of the developer reviewing release k.

Even though it is common for the observational data collected in software engineering studies (in particular by mining software repositories) to have a temporal dimension, a study may choose to abstract away the temporal dependencies and perform a cross-sectional analysis—possibly as a stepping stone towards a full-fledged longitudinal analysis. In the aforementioned domain of code reviews, a cross-sectional study would simply consider each data point as a "snapshot" of the relations $R_k \to Q_k$ at release (time) k. Kosuke and Kim (2019) elucidate the impact of abstracting away time in longitudinal data, and under what assumptions it does not introduce confounding.

On the other hand, there have been several proposals in the literature of techniques to model and correct for confounding in time-dependent data. Marginal structural models (MSMs) (Robins 2000) and structural nested models (Vansteelandt and Joffe 2014) are especially interesting from our perspective, since they are built atop the same causal DAGs that underlie this paper's technique. For instance, MSMs build DAGs that represent the relation between variables at different time steps. In the code review domain, for example, we could posit a causal relation $E_k \to E_{k+1}$ if a reviewer may opt in to be assigned to the next code review.

3 Confounding in Programming Languages and Code Quality

Our first illustrative example follows closely the fundamental structure of Section 2.3's prototypical example, while recasting it in a more realistic setting. Our goal is estimating the effect of using different programming languages (predictor variable $\mathrm{Language}$) on the quality of the produced code written in that language (outcome variable $\mathrm{Quality}$).

As we discussed in depth in related work (Furia et al. 2022),[7] a programmer's ability (expressed by variable Skill) is likely to confound the causal effect of $\mathrm{Language}$ on $\mathrm{Quality}$.[8] Namely, a more skilled programmer is likely to produce higher-quality code

[7] In this section, we analyze a subset of the same data collected by others (Ray et al. 2014; Emery et al. 2019) that we also analyzed in our previous work (Furia et al. 2022). This section's and our previous work's analysis (Furia et al. 2022) are, however, orthogonal: in this section, we analyze possible confounding effects based on causal assumptions; in contrast, Furia et al. (2022)'s analysis is purely statistical and targets Bayesian statistical modeling practices.

[8] It's plausible several other confounders of this causal relation exist (Furia et al. 2022); for clarity of presentation, we only consider Skills, as if it captures the effects of other possible confounders of the same relation.

(Skill → Quality); and programmers with different abilities may prefer to work with certain programming languages over others (Skill → Language). Figure 3 summarizes these relations by means of a DAG, which is isomorphic[9] to Fig. 1c's abstract DAG.

As discussed in Section 2.3, this DAG structure entails that, if we want to estimate the true effect of Language on Quality for observational data—where we cannot control the effect of Skill on Language, that is, we cannot randomize which language each programmer will use—we need to also *condition* on the confounder Skill. In practice, this may be impossible because precisely measuring Skill is not easy: for example, if the data comes from a source code repository, the identity of the programmers may be unknown; even if we have access to a programmer's identity, it may be practically cumbersome to reliably assess their programming skills.

The rest of this section demonstrates how we can mitigate the effects of this unobserved variable bias, in a way that we can still get something out of our observational data about languages and code quality—even in such a tainted scenario.

3.1 Data: Programming Languages and Code Quality

To get a plausible ground truth about the relative effects of skills and languages on code quality, we analyze a subset of the data collected for a large-scale repository study (Ray et al. 2014), as made available in Emery et al. (2019)'s reanalysis. This dataset summarizes the commit history of hundreds of projects in various languages.

To keep things simple—and to avoid bias that may come from underrepresented or misclassified languages (Emery et al. 2019)—we only retain data about projects: *i*) written in Python or Java (two widely used, yet fairly different languages); *ii*) with at least 100 commits on record; *iii*) that are not multi-language (that is, each project is entirely written in Java, or entirely written in Python). These criteria select 105 projects: 45 written in Java and 60 written in Python. For each of these projects, we are interested in two key variables:

Language: a binary, ordinal variable (values: Java or Python), which denotes the project's language;

Quality: a continuous variable (ranging over [0, 1]), which measures the project's quality as the complement $1 - \text{Bugs}/\text{Commits}$ of the fraction of all project commits that are flagged as introducing a bug.[10]

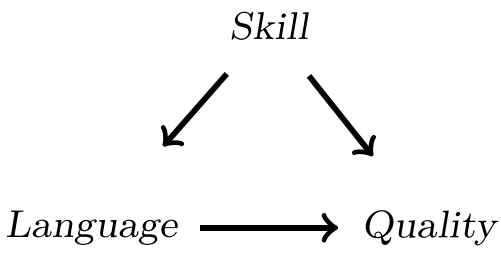


Fig. 3 The effect of Language on code Quality is confounded by the programmer's Skill

[9] Figures 3 and 1c are isomorphic as graphs. However, the variables associated with corresponding nodes may have different characteristics—most notably, X is continuous whereas Language is categorical.

[10] Naturally, this is a very crude, simplistic measure of quality. Again, this example's goal is not to discover new empirical knowledge about the fault proneness of different programming languages, but to illustrate how to apply technique to account for possible omitted variable bias.

This dataset also includes information about which developer produced each commit. We use it to derive a crude estimate of the *skills* of developers active on a project as follows. First, we only consider the 952 developers who produced at least 10 commits each in the selected projects. The skill of each developer *d* among these "frequent committers" is the complement $1 - \text{Bugs}_d/\text{Commits}_d$ of the fraction of all commits authored by *d* that introduced a bug. Finally, variable Skill summarizes the skills associated with each project:
Skill: a continuous variable (ranging over [0, 1]), which is the mean skill of all developers among the "frequent committers" who contributed to the project.
We stress that the goal of this data selection process is *not* supporting any general claims about the actual effects of a programming language on a project's quality. It simply gives a rough idea of the magnitude of these effects in a real world scenario, so that we can appreciate that confounding is a plausible occurrence—hence, a practical concern.

3.2 Quantifying Confounding

In this exercise, the goal is estimating the effect of choosing Java or Python as a programming language (variable Language) on the quality of the developed project (variable Quality). If we had access also to variable Skill, we could single out the Language → Quality effect by fitting Fig. 4a's regression model m_3, which uses Language and Skill as predictors. On the data described in Section 3.1, this produces an estimate of the effect Language → Quality of $\ell = -0.012$, which indicates that using Python is very weakly associated with a modest reduction of quality (with a lot of uncertainty, as shown in Fig. 4d).

In practice, it may not be possible to reliably measure the confounder Skill. In this case, we can only fit Fig. 4b's model m_4, which gives us a different estimate $\ell = -0.052$ of the effect Language → Quality of Language on Quality. If we compare this with the previous estimate based on the unbiased model m_3, we notice that the confounding of Skill *inflates* the effect Language → Quality as measured in this data. In reality, we would not have access to the unbiased estimate; how to assess how much confidence we can put in an estimate that comes from a possibly confounded model? There are three main ways of proceeding (Lin et al. 1998):

- If we can muster an estimate the confounding effect Skill → Quality, we can use it to compute how strong the effect Skill → Language would have to be to *tip* the estimate of the Language → Quality effect (i.e., flip it from negative to positive). Section 3.2.2 discusses this scenario.
- Conversely, if we can estimate the confounding effect Skill → Language, we can compute how strong the effect Skill → Quality would have to be to tip the estimate of the Language → Quality effect. Section 3.2.3 discusses this scenario.
- Finally, assuming that there are several different confounders that simultaneously affect Language and Quality, we can compute how many confounders with a certain effect would it take to tip the estimate of the Language → Quality effect. Section 3.2.4 discusses this scenario.

Assessing when tipping may occur under plausible scenarios can inform us about the practical impact of the confounder on our data analysis: if tipping requires a strong confounding effect, which is unlikely in practice, we can probably tolerate the bias and still use our estimate, albeit imperfect, of the effect of Language on Quality. Conversely, if even a weak

$$Quality_i \sim \text{Normal}(\mu_i, \sigma)$$
$$\mu_i = \alpha + \ell \cdot Language_i + s \cdot Skill_i$$

(a) Model m_3 with two predictors *Language* and *Skill*.

$$Quality_i \sim \text{Normal}(\mu_i, \sigma)$$
$$\mu_i = \alpha + \ell \cdot Language_i$$

(b) Model m_4 with one predictor *Language*.

$$Quality_i \sim \text{Normal}(\mu_i, \sigma)$$
$$\mu_i = \alpha + s \cdot Skill_i$$

(c) Model m_5 with one predictor *Skill*.

	MODEL	ESTIMATE	95% PROB.		60% PROB.	
m_3	uncounfounded	-0.012	-0.035	0.011	-0.021	-0.002
m_4	confounded	-0.052	-0.094	-0.010	-0.070	-0.034

(d) Estimates of regression coefficient ℓ in models m_3 and m_4, with 95% and 60% probability intervals.

Fig. 4 Different models to estimate the dependence between Quality and Language

confounder is likely to tip, we would conclude that we should not rely on the observational data analysis, and try to collect different kinds of data (or to use the existing data to answer different kinds of questions). Such an analysis is useful also in more open-ended scenarios, for example when we don't really know what (other) factors may affect Language and Quality but we have reasons to believe that *some* unmeasured variables exist. Conversely, tipping analyses are also applicable when testing a hypothesis: in such cases, the tipping value could indicate the boundary between "significant" and "not significant" effect.

To perform these so-called tipping point sensitivity analyses we will use the R package `tipr` (Lucy and Mcgowan 2022), which implements state-of-the-art analysis techniques (Lin et al. 1998) based on causal models similar to those outlined in Section 2.2.

3.2.1 Scaled-mean Difference

In a tipping-point sensitivity analysis (Lucy and Mcgowan 2022), any unmeasured confounder Z is modeled as a standardized, normally distributed random variable. For a binary treatment variable X with two nominal values X_0, X_1, this means that the impact of the confounder Z on the treatment X is captured by normal distributions with unit variance and mean μ_0 (when $X = X_0$) and μ_1 (when $X = X_1$).

Accordingly, we use the *scaled-mean difference* (SCD) to quantifying the impact of a confounder (Skill in our example) on the treatment (Language in our example) in a sensitivity analysis (Lucy and Mcgowan 2022).[11] In our case, the $\text{SCD}(Skill \rightarrow Language)$ of a programmer's skills in different language is the difference between the mean Skill of a Python programmer and the mean Skill of a Java programmer, expressed as a multiple of Skill's standard deviation:

$$SCD(\text{Skill} \rightarrow \text{Language}) \quad = \quad \frac{\mathbb{E}[\text{Skill} \mid \text{Python}]}{\sigma[\text{Skill} \mid \text{Python}]} - \frac{\mathbb{E}[\text{Skill} \mid \text{Java}]}{\sigma[\text{Skill} \mid \text{Java}]} \tag{1}$$

In (1), $\text{Skill} \mid x$ denotes the values of Skill in all datapoints where $\text{Language} = x$, whereas $\mathbb{E}[D]$ denotes the mean and $\sigma[D]$ the standard deviation of some data D.

Since it's based on a standardized scale, the SCD is easy to interpret uniformly, across different settings and domains. For instance, an SCD of ± 3 denotes a huge effect: three standard deviations of difference indicate that the distribution of skills of Python and Java

[11] The scaled-mean difference $\mu_1/\sigma_1 - \mu_2/\sigma_2$ between two groups is similar to, but different from, the standardized mean difference $(\mu_1 - \mu_2)/\sigma$—a commonly used effect size. Precisely, the scaled-mean difference is simply the difference of standardized means in the treatment and control groups.

programmers barely overlap; clearly, such a massive difference in skills is unlikely to happen in practice. In our dataset, *SCD Skill Language* is -1.545, which denotes quite a sizeable, albeit not huge, effect size.

3.2.2 Confounding SCD

In this scenario, we have measured the (possibly confounded) effect $\text{Language} \rightarrow \text{Quality}$, and we have an idea (for instance, from other studies) of a plausible value for the confounding effect $\text{Skill} \rightarrow \text{Quality}$. From this data, we calculate the SCDSkillLanguage that would lead to a confounding such that our estimate of the effect $\text{Language} \rightarrow \text{Quality}$ has the opposite sign of the "true" effect.

In our running example, the estimate of $\ell = -0.052$ using model m_4 is the measured effect $\text{Language} \rightarrow \text{Quality}$; whereas the estimate of $s = 0.835$ using Fig. 4c's model m_5 gives us a plausible value for the confounding effect $\text{Skill} \rightarrow \text{Quality}$. As shown in Table 2, a modest SCDSkillLanguage of -0.062 would be sufficient to flip the sign of the measured effect. Such an SCD is fairly modest, and likely to happen in practice; in fact, we have seen that the SCD measured in the data is much larger. In all, we cannot have much confidence that the estimate of the effect $\text{Language} \rightarrow \text{Quality}$ is valid.

3.2.3 Confounding Effect

In this scenario, we have measured the (possibly confounded) effect $\text{Language} \rightarrow \text{Quality}$, and we have an idea (for instance, from other studies) of a plausible value for the confounding SCD $\text{Skill} \rightarrow \text{Language}$. From this data, we calculate the effect $\text{Skill} \rightarrow \text{Quality}$ that would lead to a confounding such that our estimate of the effect $\text{Language} \rightarrow \text{Quality}$ has the opposite sign of the "true" effect.

In our running example, the estimate of $\ell = -0.052$ using model m_4 is, once again, the measured effect $\text{Language} \rightarrow \text{Quality}$; whereas the $SCD(Skill \rightarrow Language) = -1.545$ measured on the data according to (1) gives us a plausible value for the confounding effect $\text{Skill} \rightarrow \text{Language}$. As shown in Table 2, a modest effect $\text{Skill} \rightarrow \text{Quality}$ of 0.034 would be sufficient to flip the sign of the measured effect. Such a confounding effect is fairly modest, and likely to happen in practice; in fact, we have seen that the estimate of this effect $s = 0.835$ using Fig. 4c's model m_5 is much larger. Also in this scenario, we cannot have much confidence that the estimate of the effect $\text{Language} \rightarrow \text{Quality}$ is valid.

Table 2 Three scenarios where we calculate the effect sufficient to produce TIPPING the MEASURED effect $\text{Language} \rightarrow \text{Quality}$, based on an ESTIMATE of the confounder Skill on either the outcome (first row) or the treatment (second row), or of several unknown confounders C (bottom rows)

MEASURED		ESTIMATE		TIPPING	
effect $\text{Language} \rightarrow \text{Quality}$	-0.052	effect $\text{Skill} \rightarrow \text{Quality}$	0.835	SCD $\text{Skill} \rightarrow \text{Language}$	-0.062
effect $\text{Language} \rightarrow \text{Quality}$	-0.052	SCD $\text{Skill} \rightarrow \text{Language}$	-1.545	effect $\text{Skill} \rightarrow \text{Quality}$	0.034
effect $\text{Language} \rightarrow \text{Quality}$	-0.052	effect $\text{C} \rightarrow \text{Quality}$	0.170	number of confounders C	2
		SCD $\text{C} \rightarrow \text{Language}$	-0.150		

3.2.4 Number of Confounders

In this scenario, we have measured the (possibly confounded) effect $\mathrm{Language} \rightarrow \mathrm{Quality}$, and we are considering several different confounders. We have an idea of a plausible value for both the SCD $\mathrm{C} \rightarrow \mathrm{Language}$ and the effect $\mathrm{C} \rightarrow \mathrm{Quality}$ for any such unknown confounders C. From this data, we calculate how many such variables C would produce an overall confounding such that our estimate of the effect $\mathrm{Language} \rightarrow \mathrm{Quality}$ has the opposite sign of the "true" effect.

As usual, the estimate of $\ell = -0.052$ using model m_4 serves as the measured effect $\mathrm{Language} \rightarrow \mathrm{Quality}$. Then, we can speculate that a generic confounder C has an $SCDCLanguage = -0.15$ and an effect $\mathrm{C} \rightarrow \mathrm{Quality} = 0.17$. These values are respectively 1/10 and 1/5 of the corresponding values for Skill as measured in the dataset; intuitively, they represent confounders with a much more tamed power compared to Skill. Nevertheless, just two such generic confounders would be sufficient to flip the sign of the measured effect. Since it is definitely plausible that there are a couple of confounders with moderate effect, we reach again the same conclusion that we cannot have much confidence that the estimate of the effect $\mathrm{Language} \rightarrow \mathrm{Quality}$ is valid.

3.3 Sensitivity Analysis

Let's generalize the tipping analysis described in this section beyond (Ray et al. 2014)'s data. First of all, consider a range of possible measured effects $\mathrm{Language} \rightarrow \mathrm{Quality}$, including both positive and negative values. For each of them, Fig. 5 plots a line of a different color that marks the combination of values of confounding effect $\mathrm{Skill} \rightarrow \mathrm{Quality}$ (horizontal axis) and SCD $\mathrm{Skill} \rightarrow \mathrm{Language}$ (vertical axis) that would tip the measured effect.

Figure 5 indicates that the two confounders, $\beta_{\mathrm{Skill} \rightarrow \mathrm{Quality}}$ and $SCD_{\mathrm{Skill} \rightarrow \mathrm{Language}}$ are inversely proportional. Intuitively, this is because both confounding factors concur to introduce tipping, but each affects a different variable; thus, the more pronounced one of them is, the less it is required of the other. Another observation is that the various colored lines in Fig. 5 become flatter as the absolute value of the measured effect $\beta_{\mathrm{Language} \rightarrow \mathrm{Quality}}$ decreases. In fact, if the measured effect is small, even moderate-magnitude confounders may introduce a strong bias; whereas stronger effects are swayed only by correspondingly strong confounders.

Such a sensitivity analysis can provide a useful guide not only to analyze empirical data, but to *plan* new experiments to improve the validity of existing findings. For example, it would be interesting to collect reliable data about the relations $\mathrm{Skill} \rightarrow \mathrm{Quality}$ (what's the impact of a developer's skills on the quality of code they produce?) and $\mathrm{Skill} \rightarrow \mathrm{Language}$ (do developers with different skill have marked preferences for which language to use?). Furthermore, a sensitivity analysis would enhance the value of an empirical study for other researchers, since it would better, and quantitatively, identify the study's envelope of validity, and it would make the study's assumptions more transparent.

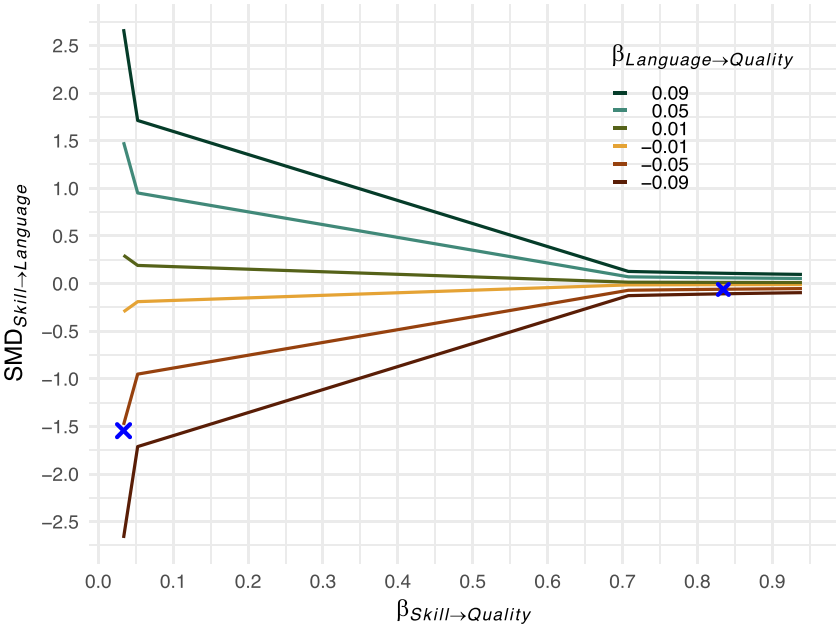


Fig. 5 A graphical summary of tipping analyses of confounder effect of Skill on the Language → Quality effect. For different measured values of effect ($\beta_{\text{Language}\to\text{Quality}}$), the plot shows the values of the effect Skill → Quality ($\beta_{\text{Skill}\to\text{Quality}}$) and of the SCD Skill → Language (labeled $\text{smd}_{\text{Skill}\to\text{Language}}$) that would tip the main effect. The two blue cross marks correspond to the two scenarios discussed in Section 3.2.2 and Section 3.2.3

4 Confounding in Teamwork Effort

Our second illustrative example is based on Feldt et al. (2025)'s review of research in the factors that affect the productivity of software development teams. It demonstrates how to practically assess the impact of potential unknown confounders in a more complex, realistic scenario. As in Section 3.2's illustrative example, our aim is not to (directly) contribute to the knowledge about developer productivity; in fact, our analysis will at times adopt simplifying assumptions that are not (entirely) realistic. In contrast, the goal is to demonstrate credible confounding patterns, and to illustrate how to navigate around them in realistic conditions.

4.1 Data: Teamwork, Effort, and Other Covariates

Feldt et al. (2025) propose to use causal DAGs to summarize and combine the key findings of systematic literature reviews. In one of their case studies, they review several primary studies about productivity in software development. The DAG shown in Fig. 6 is one of the DAGs that is obtained by applying (Feldt et al. 2025)'s approach; it summarizes the main relevant relations between variables that have been observed in some of the reviewed literature on the topic.[12]

[12] This does not mean that this is the ultimate summary of research in the area of software development productivity. For our purposes, all that matters is that it displays a rich collection of *plausible* relations, so that our omitted variable bias analysis is grounded in a realistic scenario.

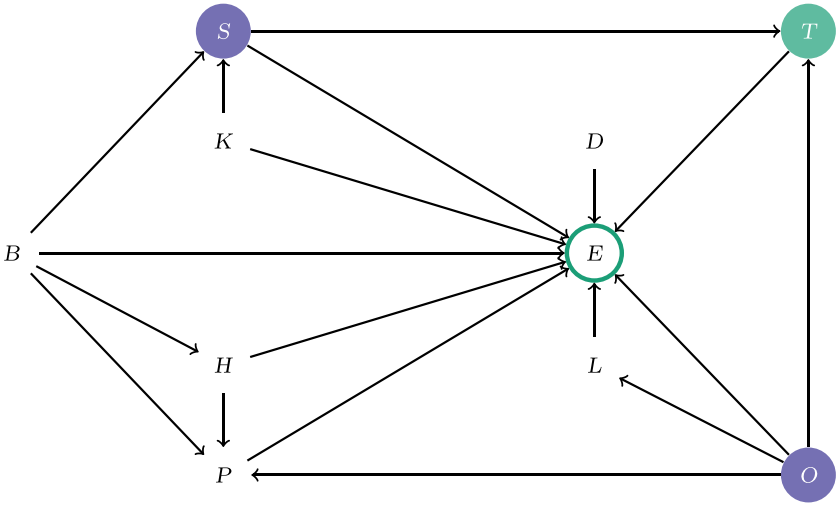


Fig. 6 A DAG summarizing relations among variables that characterize a software project, based on Feldt et al. (2025)'s literature review. The variables are, from top to bottom and left to right: software Size, T eam size, software Kind, Domain, Business area, Effort, Hardware, Location, Programming language, and Organization type

Figure 6's DAG consists of 18 relations (arrows) among 10 variables:

B the company's **b**usiness area (accounting, sales, human resources, ...)
D the project's **d**omain (healthcare, finance, entertainment, ...)
E the overall **e**ffort spent by the software developers (hours worked, planned years, ...)
H the **h**ardware that runs the software (server, client, mobile, embedded, ...)
K the **k**ind of software (application, library, system, web, ...)
L the company's **l**ocation (North America, Europe, Asia, ...)
O the company's **o**rganization type (private, public, non-profit, ...)
P the project's **p**rogramming language
S the software **s**ize (lines of code, function points, ...)
T the size of the **t**eam of programmers on the project

In the rest of this section, we put ourselves in the shoes of a researcher who is designing a new study to determine the strength of the causal relation between T (team size) and E (effort). The key questions that need to be addressed to design such a study are:

1. What variables, other than the treatment T and the outcome E, ought to be measured?
2. Given the variables that could be effectively measured, what possible remaining confounders of the causal effect of treatment T on outcome E may remain?

Since Fig. 6 summarizes several primary studies in the domain of software development productivity, we can conveniently use it as the basis of further studies in the same domain. Even if we were targeting a domain with little prior research, we could still build a DAG that captures whatever is known based on the state of the art in this domain. A DAG is just a convenient notation to summarize knowledge about the structural relations among

variables; if previous work is scarce, the DAG will be simplistic or incomplete but will still serve as a useful guide. At a minimum, we can always fall back to building a minimal DAG such as in Fig. 1c, which just captures the relation of interest $X \rightarrow Y$ and a generic confounder Z.

4.2 Adjustment Sets

As in the previous illustrative example, to estimate the effect of a treatment (T in our example) on an outcome (E in our example), we fit on the data a linear regression model m_A:

$$\begin{aligned} \mathrm{E}_i &\sim \mathrm{Normal}(\mu_i, \sigma) \\ \mu_i &= \alpha + \beta_t \cdot \mathrm{T} + \sum_{v \in A} \beta_v \cdot v_i \end{aligned} \tag{2}$$

The estimate of coefficient β_t in the fitted model m_A measures the effect $\mathrm{T} \rightarrow \mathrm{E}$.

Thus, addressing question *i*) above (*What variables ought to be measured?*) is tantamount to deciding which variables (covariates)[13] should be included in set A in model m_A (2). In particular, A should include all *confounders*, so that the effect $\mathrm{T} \rightarrow \mathrm{E}$ can be estimated without bias. Set A represents what is called an *adjustment set*: given a DAG and two of its variables—representing the outcome (E in our case) and the treatment (T in our case)—the adjustment set A is the set of additional variables in the DAG that we have to include in (2)'s model to ensure that β estimates the unbiased, uncounfounded, genuine causal effect of T on E. In other words, the adjustment set includes all predictors that we should include if we want to avoid introducing omitted variable bias.

The adjustment set (or rather adjustment sets, since a DAG may admit multiple, alternative adjustment sets) can be computed directly on the DAG based solely on its structure—assuming, of course, that the DAG correctly captures real-world causal relations.[14] In our example, the adjustment set is $A = \{\mathrm{O}, \mathrm{S}\}$; therefore including the two predictors O and S, in addition to T, ensures that β captures the net causal relation between treatment and outcome.

Remember that the adjustment set is a variable selection strategy whose goal is correcting for possible confounding effects of causal relations. As explained in Section 1.2, this is the specific focus of this paper; naturally, different kinds of research questions may require different variable selection criteria.[15] This does not necessarily imply that adjusting for confounding bias is irreconcilable with other data analysis goals. In fact, we will see that a DAG often admits different equivalent adjustment sets, which often provides more flexibility in how this technique is applied.

[13] In common statistical jargon, a covariate is any predictor variable other than the treatment.

[14] In previous work (Furia et al. 2023), we illustrated how adjustment sets are computed, and what's the intuition behind them. For brevity, we do not repeat the explanation in this paper, but simply *use* the adjustment sets computed from a DAG in our analysis.

[15] In particular, if the goal is fitting a statistical (regression) model with an optimal trade off between predictive accuracy and model conciseness, information-theoretic model selection criteria provide a serviceable approach (Dvorzak and Wagner 2016; Watanabe 2010; Vehtari et al. 2017).

4.3 Unmeasured Confounders

Even though Fig. 6's DAG is based on several empirical studies, it still is completely plausible that it does not include all factors that contribute to the observed relation between treatment and outcome. In fact, in every complex, real-world process, it is exceedingly likely that there are unmeasured variables that might still have a sizeable impact on the variables of interest.

To address this issue of unmeasured (unknown) additional confounders, we can extend the previous adjustment set analysis. For every pair of nodes $X \to Y$ in Fig. 6's DAG, we introduce an unmeasured confounder $X \leftarrow Z_{X,Y} \to Y$ that affects X and Y simultaneously. Then, we recompute the adjustment set of the DAG extended with such additional node. Table 3 shows the results of this analysis.

Adding a confounder to 16 out of 18 edges in Fig. 6's DAG does not change the adjustment set, which remains $\{O, S\}$—as when considering the original DAG without unknown confounders. In other words, including O and S in model m_A (2) conveniently also voids the effect of other possible confounders. In contrast, a confounder $Z_{S,T}$ affecting edge $S \to T$ admits two adjustment sets, one that includes $Z_{S,T}$ itself, and one that does not. Clearly, we prefer the latter adjustment set $\{B, K, O, S\}$: since it does not include unknown variable $Z_{S,T}$, it ensures that any possible confounding effect of $Z_{S,T}$ is corrected for indirectly, even if $Z_{S,T}$ cannot be measured—in fact, without even knowing what this variable represents.

Unfortunately, the last edge $T \to E$ cannot be handled so easily. If there were an additional variable $Z_{T,E}$ simultaneously affecting T and E, the only way to correct its confounding effect would be to measure $Z_{T,E}$ and include it as a predictor in (2)'s model. By definition, this is impossible because we don't know what $Z_{T,E}$ is, or we have an idea but it is impractical or impossible to measure it.

Table 3 Adjustment sets for Fig. 6's DAG extended with a confounder $Z_{X,Y}$ affecting each edge $X \to Y$

CONFOUNDED EDGE		ADJUSTMENT SETS
1	$B \to E$	$\{O, S\}$
2	$B \to H$	$\{O, S\}$
3	$B \to P$	$\{O, S\}$
4	$B \to S$	$\{O, S\}$
5	$D \to E$	$\{O, S\}$
6	$H \to E$	$\{O, S\}$
7	$H \to P$	$\{O, S\}$
8	$K \to E$	$\{O, S\}$
9	$K \to S$	$\{O, S\}$
10	$L \to E$	$\{O, S\}$
11	$O \to E$	$\{O, S\}$
12	$O \to L$	$\{O, S\}$
13	$O \to P$	$\{O, S\}$
14	$O \to T$	$\{O, S\}$
15	$P \to E$	$\{O, S\}$
16	$S \to E$	$\{O, S\}$
17	$S \to T$	$\{B, K, O, S\}\{O, S, Z_{S,T}\}$
18	$T \to E$	$\{O, S, Z_{T,E}\}$

4.4 Sensitivity Analysis

Figure 7 summarizes the analysis results so far: in order to get an unbiased estimate of the $T \rightarrow E$ causal relation in Fig. 6's DAG, we should include B, K, O, S as additional predictors, which safeguards against unmeasured confounders affecting other relations in the DAG. However, there remains a possible confounder $Z_{T,E}$—which we'll just call Z from now on—that cannot be controlled for indirectly by means of other known variables.

Whether Z's confounding is negligible or consequential ultimately depends on its strength relative to the strength of the other causal relations. Roughly, if $E \rightarrow T$ and the other relations are very strong, whereas $Z \rightarrow E$ and $Z \rightarrow T$ are very weak, our estimate of the $E \rightarrow T$ relation has a good chance of remaining reliable even if we cannot measure the unknown Z. Our next analysis step is thus a sensitivity analysis based on *simulation*: using some plausible, informed estimates for the various effects in the DAG, we'll try to recover the effect $E \rightarrow T$ without measuring Z, and we'll see how far off this estimate is from the ground truth.

4.4.1 Simulation Parameters

For our simulation, we use the following generative model, which mirrors the structure of Fig. 7's DAG:

$$
\begin{aligned}
B &\sim \text{Binomial}(1, 0.5) & K &\sim \text{Binomial}(1, 0.5) \\
O &\sim \text{Binomial}(1, 0.5) & Z &\sim \text{Normal}(0, 1) \\
S &\sim \text{Normal}(b_s B + k_s K, 1) & T &\sim \text{Normal}(o_t O + s_t S + \gamma_t Z, 1) \\
E &\sim \text{Normal}(b_e B + k_e K + o_e O + s_e S + t_e T + \gamma_e Z, 1)
\end{aligned}
\tag{3}
$$

In (3), every categorical variable is binary and follows a Bernoulli distribution with 0.5 probability of drawing a 1; the other variables are real-valued and follow a normal distribution with unit variance and mean that is given by a linear combination of the variables

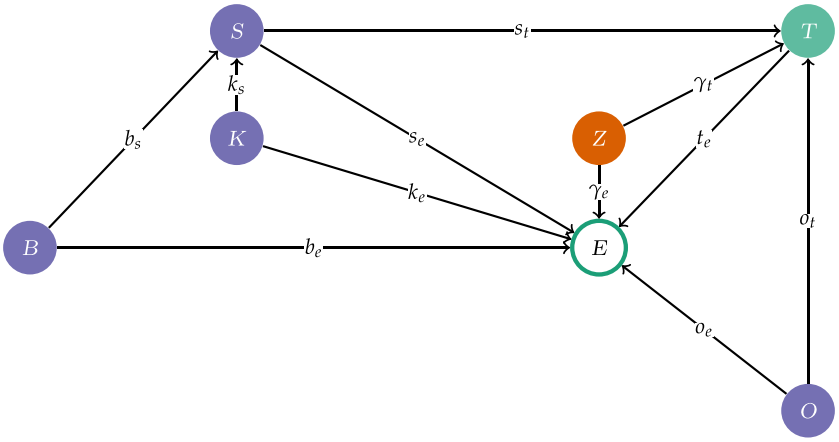


Fig. 7 Figure 6's DAG simplified to include only treatment T, outcome E, as well as all variables in the adjustment set that also accounts for unknown/unmeasured confounders on other edges. The DAG also shows an unmeasured confounder Z that may still exist: we cannot adjust it away with any other variables

that directly affect it according to Fig. 7's model. This is admittedly a strongly simplified generative model, but it has the advantage that we can choose standardized effect sizes for its parameters, instead of having to rely on difficult-to-obtain estimates on a natural scale. Furthermore, (3) could be generalized (e.g., to include categorical variables with more than two possible values), or specialized according to domain-specific characteristics (e.g., to use truncated distributions that capture hard bounds of the value of some variables).

The generative model in (3) has 11 parameters, one for each edge in Fig. 7's DAG. Each parameter x_y denotes the effect of X on Y, corresponding to edge $X \to Y$. The exceptions are edges $Z \to E$ and $Z \to T$, whose effects we denote as γ_e and γ_t to single them out (as they are the part of the model that is completely unmeasured). In a concrete case study, the simulation would use parameters that reflect the system that is actually being observed, where the data comes from. In our case, we do not have a specific data collection process in mind. Alternatively, one could simply try out all parameter combinations that are remotely plausible. In our case, such an exhaustive analysis would be practically infeasible; for example, if each parameter can take 6 possible values, we would end up with 6^{11} parameter combinations—that is over 362 millions!

Instead, we go back to Feldt et al. (2025) and use the statistics from some of the reviewed primary studies to get a ballpark estimate of the relevant effects. This trades off some generality for a manageable simulation time. Table 4 shows the range of parameter values that we used for our simulations.

Confounder: First of all, we want to simulate all plausible confounding scenarios of Z; therefore, we consider all combinations of small (0.1), medium (0.3),

Table 4 Range of values for the parameters of the generative model in (3) used in the simulation. For each parameter, the table also reports the JUSTIFICATION for the choice of VALUES, usually as a reference to a primary study that measured such an effect or a comparable one

PARAMETER		VALUES	JUSTIFICATION
b_e	$B \to E$	0.3	Tsunoda and Ono (2014) Range of values for the parameters of the generative model
b_s	$B \to S$	0.3	Same as b_e
k_e	$K \to E$	0.1	Wang et al., (2008) Tab. XIII, % variance explained/project type
k_s	$K \to S$	0.1	Same as k_e
o_e	$O \to E$	0.5	Tsunoda and Ono, (2014) Tab. IV, ω^2/maintenance (all)
o_t	$O \to T$	0.5	Same as o_e
s_e	$S \to E$	-0.1	Tsunoda and Ono (2014) Tab. XIII, ρ/maintenance (all)
s_t	$S \to T$	-0.1	Same as s_e
t_e	$T \to E$	0.1, 0.3, 0.5	all plausible positive effect sizes
γ_e	$Z \to E$	-0.5, -0.3, -0.1, 0.1, 0.3, 0.5	all plausible effect sizes
γ_t	$Z \to T$	-0.5, -0.3, -0.1, 0.1, 0.3, 0.5	all plausible effect sizes
n	sample size	5, 10, 50	
n_{sim}	repetitions	200	

and large (0.5) standardized effect sizes[16]—both positive and negative—for each parameter γ_e and γ_t.

Main effect: We also consider all possible effect sizes for parameter t_e, which represents the treatment effect that we are trying to estimate; however, we only consider *positive* effect sizes since, according to the studies reviewed in Feldt et al. (2025), it is implausible that large teams produce an overall lesser effort than small teams.

Indirect effects: As for the effects x_e corresponding to the edges $X \to E$, for every other variable X in the adjustment set, we picked a small, medium, or large, positive or negative effect size based on some of the studies reviewed in Feldt et al. (2025). Since those studies focused on effort (or related variables) as outcome, we could not find any hard data about the magnitude of the effects of these other variables X on *other* covariates. Simplistically, we assume that x_y is the same as x_e, that is variable X has roughly the same effect on all variables it directly affects. Again, a specific case study could come up with more definite estimates; our goal is mainly to demonstrate this analysis method on somewhat plausible data.

The simulation includes implicitly two more parameters:

Sample size: the sample size n determines how many datapoints we sample from (3)'s generative model. We try three different sample sizes: 5, 10, and 50. Due to the nature of the data we are simulating, small sample sizes (i.e., 5 and 10) are especially relevant and realistic: collecting all such detailed data about many software projects would be costly (in particular, it's unlikely that such data can be reliably obtained by simply mining open-source repositories); hence, an actual empirical study would likely be limited to a smallish sample size. Nevertheless, we also include a more substantial sample size (i.e., 50) to extend the reach of our analysis.

Repetitions: For each parameter combination, we repeat the whole simulation-inference process n_{sim} times, and take the average of the obtained estimates. We go with 200 repetitions, which should be enough to smoothen out any random fluctuations in our simulations.

4.4.2 Simulation Process

For every combination of values for the parameters in Table 4, Algorithm 1 uses generative model (3) to draw random samples of the variables *B*, *K*, *O*, *Z*, *S*, *T*, *E* (consistently with the dependencies shown in Fig. 7). It then fits the following linear regression model on the samples:

$$\begin{aligned} \mathrm{E}_i &\sim \mathrm{Normal}(\mu_i, \sigma) \\ \mu_i &= \alpha + \beta_t \cdot \mathrm{T}_i + \beta_b \cdot \mathrm{B}_i + \beta_k \cdot \mathrm{K}_i + \beta_o \cdot \mathrm{O}_i + \beta_s \cdot \mathrm{S}_i \end{aligned} \tag{4}$$

[16] The values 0.1, 0.3, and 0.5 are often considered as the boundaries between negligible, small, medium, and large effect sizes for effects measured as standardized mean differences (Cohen 1988, p. 224–225) Fey et al. (2023). This is just a plausible choice of values for our illustrative example; in concrete case studies, one could pick parameter values more precisely, based on what is considered a small, medium, or large effect in the domain of interest.

Model (4) estimates the effect size β_t of T on E without conditioning on Z, which we assume cannot be measured, but otherwise includes all measurable variables in the adjustment set.

This sample/fit process is repeated n_{sim} times for each parameter combination; finally, Algorithm 1 returns the mean β_t, the 50% highest-probability density interval $\ell_{50} \dots u_{50}$, and the 95% highest-probability density interval $\ell_{95} \dots u_{95}$ of the estimates of β_t over all repetitions.[17] These statistics summarize the likely ranges of β_t estimated from the simulations.

Input: parameters $b_s, b_s, k_e, k_s, o_e, e_t, s_e, s_t, t_e, \gamma_e, \gamma_t, n, n_{\text{sim}}$
Output: estimate β_t; 50% interval $\ell_{50} \dots u_{50}$; 95% interval $\ell_{95} \dots u_{95}$
$est \leftarrow \emptyset$
// repeat n_{sim} times
for $r \leftarrow 1 \dots n_{sim}$ **do**
  $sim \leftarrow \emptyset$
  // collect n random samples from generative model (3)
  for $s \leftarrow 1 \dots n$ **do**
    $B, K, O \leftarrow$ samples from $\mathsf{Binomial}(1, 0.5)$
    $Z \leftarrow$ sample from $\mathsf{Normal}(0, 1)$
    $S \leftarrow$ sample from $\mathsf{Normal}(b_s B + k_s K, 1)$
    $T \leftarrow$ sample from $\mathsf{Normal}(o_t O + s_t S + \gamma_t Z, 1)$
    $E \leftarrow$ sample from $\mathsf{Normal}(b_e B + k_e K + o_e O + s_e S + t_e T + \gamma_e Z, 1)$
    // $sim[s, V]$ stores the value of variable V in the sth sample
    $sim[s, B], sim[s, K], sim[s, O], sim[s, Z], sim[s, S], sim[s, T], sim[s, E] \leftarrow B, K, O, Z, S, T, E$
  end
  $f \leftarrow \text{fit}((4), sim)$ // fit model (4) with data sim
  // $est[r]$ stores the fitted model's estimate of β_t in the rth repetition
  $est[r] \leftarrow \text{estimate}(\beta_t, f)$
end
// compute statistics over all n_{sim} repetitions
$\beta_t \leftarrow \text{mean}(est)$ // average
$\ell_{50}, u_{50} \leftarrow \text{HPDI}(est, 0.50)$ // 50% probability interval
$\ell_{95}, u_{95} \leftarrow \text{HPDI}(est, 0.95)$ // 95% probability interval
return $\beta_t, \ell_{50}, u_{50}, \ell_{95}, u_{95}$

4.4.3 Simulation Results

Figures 8, 9, and 10 display (a significant subset of) the simulation results. Each figure includes $16 = 4 \times 4$ subplots for each combination of the following values for γ_e and γ_t: -0.5, 0.1, 0.3, 0.5. Figure 8 refers to the experiments where the effect t_e to be estimated is small (0.1); Fig. 9 where t_e is medium (0.3); and Fig. 9 where t_e is large (0.5). Overall, the subset of experimental results displayed in the three figures is sufficient to see the main trends in the sensitivity analysis; anyway, the replication package (Furia and Torkar 2025) includes plots for all parameter combinations. The capsule summary of these results is that it is possible to retrieve a reasonably precise estimate of the effect t_e provided the confounding effect of the unmeasured Z (parameters γ_e and γ_t) is not large compared to t_e.

Let's first look at the plot grid in Fig. 9, which correspond to a "ground truth" medium effect $t_e = 0.3$ (marks ✖). When the confounding effects are small ($\gamma_e, \gamma_t = 0.1$, second row or column of the grid) or medium ($\gamma_e, \gamma_t = 0.3$, third row or column of the grid),

[17] Highest density intervals are a Bayesian analog to confidence intervals in frequentist statistics (Gelman et al. 2014; Jaynes 1976). Since highest density intervals are simply defined as intervals on a probability distribution, their interpretation is usually considered more intuitive (Hoekstra et al. 2014; Hespanhol et al. 2019).

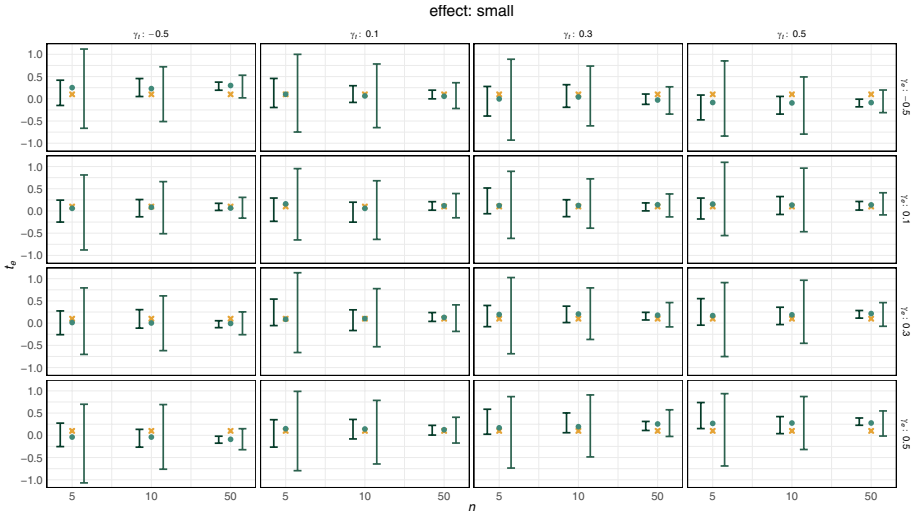


Fig. 8 A summary of the results of running Algorithm 1 for some parameter combinations in Table 4 and *small* ground truth effect $t_e = 0.1$. The plots report, for each sample size 5, 10, 50, the ground truth t_e ✖, the mean estimate ●, the 50% probability interval ▮(shifted left), and the 95% probability interval ▮ (shifted right) of β_t

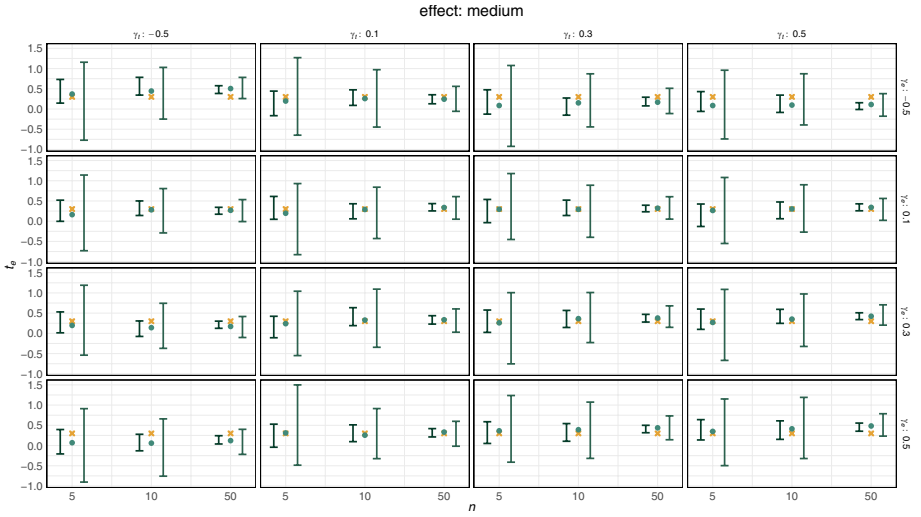


Fig. 9 A summary of the results of running Algorithm 1 for some parameter combinations in Table 4 and *medium* ground truth effect $t_e = 0.3$. The plots report, for each sample size 5, 10, 50, the ground truth t_e✖, the mean estimate ●, the 50% probability interval ▮(shifted left), and the 95% probability interval ▮ (shifted right) of β_t

the estimates (marks ●) are quite close to the actual effect, nearly overlapping it. Remarkably, this holds even for only 5 or 10 samples; however, a small sample size leads to very wide probability intervals, even for 50% probabilities (left of the estimate). Increasing the

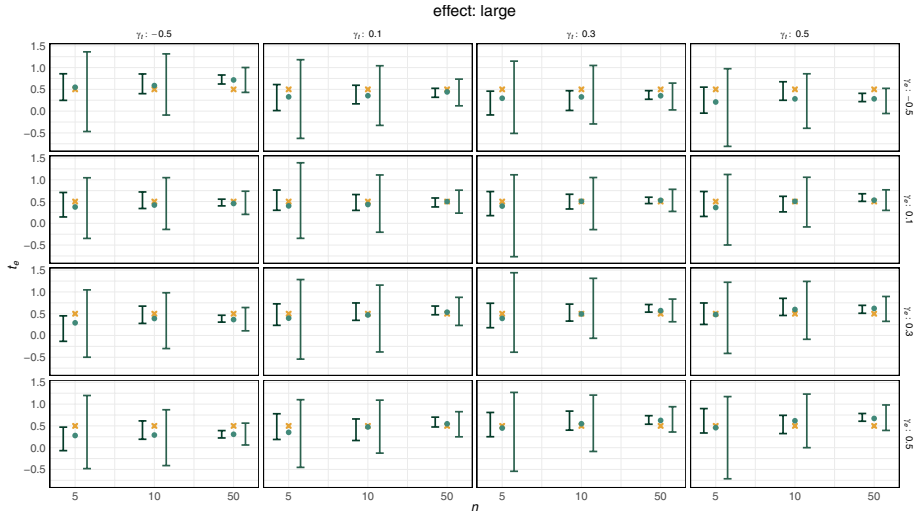


Fig. 10 A summary of the results of running Algorithm 1 for some parameter combinations in Table 4 and *large* ground truth effect $t_e = 0.5$. The plots report, for each sample size 5, 10, 50, the ground truth t_e, the mean estimate, the 50% probability interval (shifted left), and the 95% probability interval (shifted right) of β_t

sample size to 50 substantially shrinks the probability intervals; if obtaining a substantial number of datapoints is challenging in practice, these results indicates that a substantial uncertainty about the accuracy of the estimate would remain. In contrast, as we consider larger confounding effects ($\gamma_e, \gamma_t = \pm 0.5$, first and last row or column of the grid), there is a substantial gap between the estimates and the actual effect. Precisely, when γ_e and γ_t have the same sign (both positive, or both negative), Z's bias results in *overestimating* the ground truth effect; conversely, when γ_e and γ_t have opposite sign, the bias results in *underestimating* the ground truth effect.

If we now consider a "ground truth" small effect $t_e = 0.1$ (plot grids in Fig. reffig:simspsplotsspssmall) or a large effect $t_e = 0.5$ (plot grids in Fig. 10) we largely see the same trends. On the one hand, a large effect size is not strictly "harder to bias"; that is, the estimate of $t_e = 0.5$ is biased in a roughly similar way by certain confounding effects γ_e, γ_t as the estimate of $t_e = 0.1$. Intuitively, this happens because we still condition on all other variables in the adjustment set, and hence what is left is the net confounding effect of omitting Z over the estimate. On the other hand, if the "true" effect we are estimating is small compared to the biasing influence of Z, the same absolute amount of bias translates into an estimate error that is possibly much more consequential: even the 50% probability intervals clearly overlap zero; thus, in a real setting it would be hard to conclude that a definite (positive or negative) effect exists at all.

4.5 Sensitivity Analysis with E-values

The sensitivity analysis based on simulation presented in Section 4.4 is informative and flexible, but it also has clear disadvantages: it can be very time consuming, and it does usually requires a good amount of empirical data to tune the simulation parameters in a way

that is representative of the domain. As a less demanding alternative, this section presents a tipping-point analysis similar to the one we described in Section 3.2 for the other illustrative example.

4.5.1 E-values

Section 3.2's sensitivity analysis uses the SCD (scaled-mean difference) as a standardized measure of the effect of an unmeasured confounder on the treatment. The SCD (1) is defined based on a dichotomous partition of the treatment variable. In Section 3's domain, the treatment was the programming language, which is naturally dichotomous. In contrast, team size T (the treatment variable in our current domain) is an intrinsically quantitative (numeric) variable; in order to calculate an SCD of $Z \rightarrow T$, we would have to arbitrarily partition teams into small vs. large. Instead, we rely on a different kind of tipping-point analysis based on the notion of E-value (Tyler et al. 2011), which is also applicable to continuous quantitative exposure variables.[18]

The E-value[19] is "the minimum strength of association, on the risk ratio scale, that an unmeasured confounder would need to have with both the treatment and the outcome to fully explain away a specific treatment-outcome association, conditional on the measured covariates" (Tyler et al. 2011). In our scenario, an estimate β_t of the observed effect $T \rightarrow E$, obtained by fitting model (4) on empirical data, would represent the "specific treatment-outcome association, conditional on the measured covariates." The E-value can be computed from β_t, as well as ϵ_t (the standard error of β_t's estimate), an estimate of σ (parameter σ in model (4)), and a parameter δ that quantifies the arbitrarily chosen change in treatment variable T. With these parameters, an E-value of e can be interpreted as follows: if the unmeasured confounder Z were strong enough to increase, by a factor of e, the probability of raising the exposure T by δ—while simultaneously affecting the outcome E by a similar amount in standardized units—then the observed effect β_t would be entirely due to Z's confounding.

Values β_t, ϵ_t, and σ all come from fitting model (4) on empirical data; this makes the E-value a convenient way of estimating the effect of unmeasured confounders, since it is a byproduct of a standard regression analysis. Since it also depends on ϵ_t and σ, computing the E-value from a regression analysis brings the additional advantage that it takes into account the uncertainty in the estimates, as well as the other covariates the regression model conditions on. In contrast, parameter δ in the computation of the E-value can be chosen arbitrarily so that it reflects a variation "of interest" in T. Ultimately, δ still implicitly introduces a dichotomous partition of the treatment, but it does so in a way that is more apt for a continuous treatment.

4.5.2 Computing E-values

Let's get into computing the E-value in our illustrative example of team size and effort. Since we don't have real-world data directly about the variables of Fig. 7, we resort, once again, to simulation to get some plausible data in a convenient format. Unlike in Section 4.4,

[18] We rely on the R package EValue (Mathur et al. 2021), which implements a variety of state-of-the-art sensitivity analysis techniques for unmeasured confounding based on the notion of E-value (Chinn 2000; Tyler et al. 2011; Tyler and Vanderweele 2017).

[19] The "E" stands for "Evidence".

now we don't need to perform many repetitions with small sample size, since the simulated data will not be used for a sensitivity analysis but to create a synthetic dataset based on the Table 4's parameters. Thus, we simply draw ten thousand datapoints by sampling model (3) for each of the following parameter combinations:

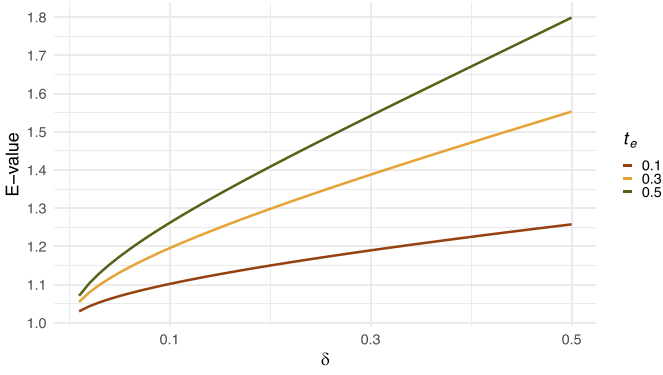
- $b_e, b_s, k_e, k_s, o_e, o_t, s_e, s_t, t_e$ are as in Table 4; namely, we consider a range of positive effect sizes for t_e, whereas we stick with realistic values for the other parameters.
- $\gamma_e = \gamma_t = 0$; in other words, we do not introduce any confounding in the simulated model. This simplifying assumption does not affect the following analysis, since it still makes sense to compute an E-value in such a scenario: the E-value quantifies the magnitude of a hypothetical confounder given an observed effect; it is not a way of detecting confounders but of reasoning about their possible strength.[20]

Then, we fit model (4) with each of the three simulated datasets $D^{0.1}, D^{0.3}, D^{0.5}$ (one for each value of $t_e = 0.1, 0.3, 0.5$). The fit with data D^s gives values β_t^s, ϵ_t^s, σ^s, respectively of the estimated effect $T \to E$, of the standard error of this estimate, and of the standard deviation of model (4)'s likelihood. With β_t^s, ϵ_t^s, σ^s, we compute the E-value for many different values of parameter δ ranging from 0.01 to 0.5.

4.5.3 Confounding Sensitivity

Figure 11a plots the E-values for the three effect sizes 0.1, 0.3, and 0.5. Overall, the E-value is proportional to δ (the parameter that captures the increase in treatment level caused by a possible confounder Z) and to t_e (the actual effect $T \to E$). This simply reflects that the bigger the effect to be explained away by a confounder, the stronger the confounder has to sway treatment and outcome at the same time.

Then, Fig. 11b lists precise E-values for certain combinations of δ and t_e. Let's look into a couple of interesting parameter combinations:



(a) Each colored line shows the E-value for each δ when the $T \to E$ effect is t_e.

	δ		
t_e	*0.1*	*0.3*	*0.5*
0.1	1.10	1.19	1.26
0.3	1.20	1.39	1.55
0.5	1.26	1.54	1.80

(b) E-values for different combinations of $T \to E$ effect t_e and parameter δ.

Fig. 11 E-value tipping-point analysis for the unknown confounder Z on the $T \to E$ relation in Fig. 7

[20] For completeness sake, the replication package includes a computation of E-values in scenarios where γ_e and γ_t are non-zero and range over the same values as in Table 4.

i) When $t_e = \delta = 0.1$, the E-value is 1.10, which can be interpreted as follows: if Z is such that, in nominal conditions, it can raise by at least 10% the probability of increasing a project's team size by 0.1—while correspondingly increasing the effort by a "similar" amount—(where the increase is subject to some standard distributional assumptions), then Z would be sufficient to explain entirely the observed effect $t_e = 0.1$. In (3)'s standardized model, 0.1 is a *small* increase; thus, it is not unrealistic that a certain unmeasured factor increases (by 10%) the probability of ending up with a moderately larger team size and effort.
ii) When $t_e = 0.5$ and $\delta = 0.1$, the E-value is 1.26. Informally, to explain away a *large* observed effect, Z should increase by 26% the probability of introducing a *small* increase in team size ($\delta = 0.1$). This is definitely a more significant impact than in the previous point *i*), but it is perhaps still plausible.
iii) When $t_e = \delta = 0.5$, the E-value is 1.80. Informally, to explain away a *large* observed effect, Z should increase by 80% the probability of introducing a *large* increase in team size. This scenario is no longer so plausible: it is a case of "extraordinary claims [requiring] extraordinary evidence" (Sagan 1986).

Comparing scenarios *ii*) and *iii*) is also interesting. Overall, they suggest that it is still possible that a confounder is responsible for a large effect; however, this is plausible only if the confounder Z can bias the estimate of the $T \to E$ effect with only a small change in the treatment T (scenario *ii*), where the $T \to E$ is more sensitive), as opposed to a large change (which is less likely to have been missed by the quantitative analysis).

As usual, with a better understanding of the domain (for example, the typical project characteristics in the company where the analyzed data was collected), the E-value analysis could produce more actionable results. For example, if one is confident that the estimates of team size and project effort that are normally produced are usually precise, it would indicate that it's unlikely that a confounder would go undetected even for small changes of treatment/outcome (e.g., $\delta = 0.1$). Conversely, if the characteristics of the projects that are being analyzed make estimates intrinsically imprecise or uncertain, an unmeasured confounder that substantially increases the probability of swaying such estimates is a plausible possibility. As one example, Morasca and Russo (2001) study of productivity (one of those surveyed in Feldt et al. (2025)) presents estimates of productivity with a large dispersion ($\sigma \in [0.35, 7.2]$); these indicate much uncertainty, thus raising the possibility of unmeasured confounders.

5 Dealing with Omitted Variable Bias

This section is a high-level summary of the techniques presented in the paper. The summary also serves as a procedural checklist, presenting the main steps of an analysis of confounding and the order in which they should be followed.

Variables The first step is surveying the variables that characterize the target of our study, their types (categorical, ordinal, numeric, ...), how costly they are to measure (e.g., they can be mined from software repositories vs. they require running a controlled experiment), and how much uncertainty we expect in their measures. We should also select which variables

are the *treatment* (a.k.a. *exposure*) and the *outcome*, whose relation is the main focus of the study. This step underlies all the following ones, as it provides a way of becoming familiar with the study's domain in an incremental fashion.

Causal DAG The variables of interest, identified in the previous step, serve as nodes of a DAG such as those in Figs. 3 and 6. Causal DAGs are the notation of choice to succinctly express structural, causal relations among variables. As we have demonstrated in the paper, there are several analyses that are based on a DAG's structure.

What kind of information we can use to build a DAG depends on the domain we are investigating, and on the maturity of the state of the art. If there are plenty of rigorous primary studies about the quantities of interest, and perhaps even systematic reviews or meta-studies, we can summarize their evidence in a DAG—similarly to what is proposed by Feldt et al. (2025). If our study's target is novel or less established, we may have to rely on domain expertise and intuition to build a DAG. Even if there is a lot of uncertainty about the precise causal structure underlying a certain domain, there are techniques to (partially) validate candidate DAGs (Furia et al. 2023). It is quite natural to also consider different possible DAGs, and to use them to perform a "what if" analysis in different scenarios. In this step of the analysis, causal DAGs are mainly a convenient notation to rigorously express our knowledge or hypotheses about the causal relations among variables, and to investigate their consequences on the overall results of our analysis.

Adjustment Sets As shown in Section 4.2, given a DAG and treatment/outcome variables, one can systematically compute an *adjustment set*: a set of covariates that should be conditioned on in a regression to ensure that the coefficient associated with the treatment variable estimates the unbiased effect of the treatment on the outcome ("controlling for" the spurious influence of any confounders).

In the best-case scenarios, one can simply use an adjustment set to prevent confounding. Unfortunately, this is not always possible. First, the adjustment set's validity is predicated on the accuracy and completeness of the DAG: if we missed some relevant variables, or misrepresented some causal relations, an adjustment set no longer guarantees an unbiased estimate. Second, even if we are confident the DAG is accurate, certain DAGs do not admit adjustment sets (because different kinds of confounding require incompatible adjustment sets (McElreath 2020)), or some variables in the adjustment set are hard or impossible to measure (a scenario that we explored in Section 4.3). In these cases, the next steps in this list can help deal with these shortcomings.

Ballpark Estimate of Parameters In order to proceed with a sensitivity analysis of unmeasured confounders, one needs to collect a rough estimate of the strength of the main relations among variables in the DAG. If a good amount of data is available from the study's domain, we can use them to come up with estimates on the natural scale. Otherwise, we can still resort to mocking an ersatz model, based on the DAG's relations, that uses standardized variables. On a standardized scale, it is easier to make "guesstimates" about plausible parameter values (e.g., small vs. large), and to explore the impact of different parameter combinations—as we did to select the parameter values in Table 4.

Sensitivity Analysis Using the parameter estimates identified in the previous step, one can perform different kinds of sensitivity analyses of unmeasured confounder. These analyses provide a quantitative estimate (usually on a normalized scale) of how much some unmeasured confounder may bias the estimate of an effect of interest. In the paper, we demonstrated two kinds of so-called *tipping-point* sensitivity analysis, which express how strong an unmeasured confounder should be to cancel out an observed treatment/slash outcome effect. Section 3.2 presented a tipping-point analysis based on an estimate of the scaled-mean difference of an unmeasured confounder on the treatment, which is applicable to dichotomous treatment variables. Section 4.5 presented a sensitivity analysis based on E-values—a probability ratio between the confounded and non-confounded scenarios—which is also applicable to continuous treatment variables.

If a sensitivity analysis indicates that possible unmeasured confounders are unlikely to exist, or to have a noticeable impact, one can proceed with the real data analysis, reassured that confounding is a remote possibility.

Simulation Analysis If the previous step's sensitivity analysis is inconclusive, in that it failed to rule out the possibility of confounding, one can perform a more precise analysis of confounding based on simulation. Section 4.4 illustrated this on our second illustrative example of teamwork productivity. A simulation analysis is flexible, because one can explore many different variants of generative and inference models. It also supports analyzing the impact of dealing with small sample sizes in a way that realistically reflects the availability of data in the study domain. These advantages come with a cost in terms of simulation time; usually, however, it still is much cheaper to perform a detailed simulation than to embark "blind" in running an empirical study without a clear understanding of the possible confounders, and of the threats to validity they may introduce.

Study Design and Execution All previous steps are ultimately a preparation for the design and actual execution of the envisioned study. Precisely, there are three main outcomes of the previous steps:

All clear: the analysis indicates that confounding is not possible (because of the DAG structure), can be prevented (using a suitable adjustment set), or is unlikely to have a sizeable impact (as shown by the sensitivity analysis). This is the best-case scenario, which bodes well for the validity of our study.
Proceed with caution: the analysis indicates that confounding is a possibility, but, depending on the effects that are in place and on the sample size that we may be able to collect, may or may not be consequential. In this case, we may still decide to go ahead with our study or, more cautiously, we may perform additional preliminary analyses (for example using detailed simulations) to gauge more precisely the quantitative relations that animate our domain.
No go: the analysis indicates that major confounding is unavoidable, and that our estimates of effects are likely to be ridden with uncertainty. In this case, it may not be worth to proceed with the study as originally intended. Instead, we may refocus our goals, and redefine our research questions, so as to move them to a scope that is more likely to be productive.

5.1 Reporting a Sensitivity Analysis

To encourage researchers to perform sensitivity analyses as part of their quantitative studies, we outline a few simple guidelines on how to *report* the results of a sensitivity analysis in a paper.
DAG: The DAG is a succinct summary of the key assumptions that underlie a sensitivity analysis. Therefore, it's always recommended to include the DAG in the paper, preferably with a brief justification for its structure—or a reference to another publication that justifies it.
Parameter estimates: The other key ingredient for a sensitivity analysis is an estimate of the main relations among variables—in particular, possible unmeasured confounders. Since there is usually a degree of uncertainty about the "real" parameter values, one should normally indicate a *range* of values for each parameter, rather than a single point estimate. Each estimate should also be accompanied by a succinct justification: a study that suggests that value, or, at least, a rationale for an educated guess. Table 4 shows an example of how the various parameter estimates could be reported in tabular form.
Tipping plot: The outcome of a sensitivity analysis can be visually summarized with a plot of the tipping points similar to Figs. 5 and 11a. As we argued throughout the paper, a sensitivity analysis explores and compares different scenarios. Therefore, its outcome is generally not dichotomous but rather a matter of degree. A plot provides a concise, visual summary of "what if" scenarios such as: "if the unmeasured confounders are within certain ranges, then their confounding effect would/would not be able to nullify the observed effect".

6 Conclusions

This paper introduced to the empirical software engineering community techniques to assess and mitigate the so-called omitted variable bias. These techniques are crucially based on a causal model of the relations among variables of interest, formalized by means the causal DAG notation (Pearl 2009).

First, if the causal structure among variables admits an *adjustment set* that only includes measurable variables, one can correct for an omitted variable bias by simply conditioning on all variables in the adjustment set. If this is not possible, one can perform a *sensitivity analysis*, whose goal is investigating the impact of unknown confounders. The paper presented different kinds of sensitivity analyses, including both so-called *tipping-point* analyses based on canonical distributional assumptions, and more precise, but also computationally expensive, analyses based on *simulation*. We demonstrated these techniques on two illustrative examples—the relation between programming languages and code quality, and the effect on team size on software development effort—taken from recent statistical analyses of data in these two domains (Ray et al. 2014; Feldt et al. 2025).

The main high-level takeaway of this work is to *think before you act*. The most effective way of designing an empirical study is to start with an elicitation of the causal model(s) that underlie the phenomena under study, followed by a systematic and explicit sensitivity analysis of possible confounders. This will lead to a clearer understanding of the limitations of a particular study and, in turn, to a more effective study design—one that is less likely to incur major threats to validity.

Author Contributions Carlo A. Furia and Richard Torkar contributed in equal measure to the conception and design of the study, the acquisition and analysis of data, the interpretation of the findings, and the drafting and critical revision of the manuscript, and they assume joint responsibility for the integrity and accuracy of the work as a whole.

Funding Open access funding provided by University of Gothenburg. Not applicable.

Data Availability The authors hereby declare that the empirical data, together with the corpus of analysis scripts employed in the conduct of the present study, have been duly curated and are publicly accessible at the following online repository: https://figshare.com/s/fe607d8eb7c4cedbac75 Furia and Torkar (2025). Access thereto is unrestricted, and the materials are made available to enable verification, replication, and extension of the findings reported herein.

Declarations

Ethical Approval Not applicable.

Informed Consent Not applicable.

Conflicts of Interest The authors hereby unequivocally affirm that they are, to the best of their knowledge, unencumbered by any competing financial, professional, or personal interests of whatsoever nature that might, directly or indirectly, bear upon, be construed as influencing, or otherwise pertain to the research and findings herein presented.

Clinical Trial Number Not applicable.

References

Zanga A, Ozkirimli E, Stella F (2022) A survey on causal discovery: Theory and practice. Int J Approx Reason 151:101–129. https://doi.org/10.1016/j.ijar.2022.09.004

Amrehin V, Greenland S, McShane B (2019) Scientists rise up against statistical significance. Nature 567:305–307

AndaurNavarro CL, Damen JAA, Takada T, Nijman SWJ, Dhiman P, Ma J, Collins GS, Bajpai R, Riley RD, Moons KGM, Hooft L (2021) Risk of bias in studies on prediction models developed using supervised machine learning techniques: Systematic review. BMJ 375. https://doi.org/10.1136/bmj.n2281

Clark AG, Foster M, Prifling B, Walkinshaw N, Hierons RM, Schmidt V, Turner RD (2022) Testing causality in scientific modelling software. arXiv preprint arXiv:2209.00357

Angrist JD, Imbens GW, Rubin DB (1996) Identification of causal effects using instrumental variables. J Am Stat Assoc 91(434):444–455. https://doi.org/10.1080/01621459.1996.10476902

Athey S, Imbens GW (2017) The state of applied econometrics: Causality and policy evaluation. J Econ Perspect 31(2):3–32. https://doi.org/10.1257/jep.31.2.3

Berrie L, Arnold KF, Tomova GD, Gilthorpe MS, Tennant PWG (2025) Depicting deterministic variables within directed acyclic graphs: an aid for identifying and interpreting causal effects involving derived variables and compositional data. Am J Epidemiol 194(2):469–479. https://doi.org/10.1093/aje/kwae153

Bradley VC, Kuriwaki S, Isakov M, Sejdinovic D, Meng X-L, Flaxman S (2021) Unrepresentative big surveys significantly overestimate US vaccine uptake. Nature 600:695–700. https://doi.org/10.1038/s41586-021-04198-4

Camerer CF, Dreber A, Forsell E, Ho T-H, Huber J, Johannesson M, Kirchler M, Almenberg J, Altmejd A, Chan T, Heikensten E, Holzmeister F, Imai T, Isaksson S, Nave G, Pfeiffer T, Razen M, Hang W (2016) Evaluating replicability of laboratory experiments in economics. Science 351(6280):1433–1436. https://doi.org/10.1126/science.aaf0918

Furia CA, Torkar R(2025) Replication package for: Mitigating omitted variable bias in empirical software engineering. https://figshare.com/s/fe607d8eb7c4cedbac75

Chinn S (2000) A simple method for converting an odds ratio to effect size for use in meta-analysis. Stat Med 19(22):3127–3131. https://doi.org/10.1002/1097-0258(20001130)19:22%3C3127::aid-sim784%3E3.0.co;2-m

Cohen J (1988) Statistical power analysis for the behavioral sciences. Lawrence Erlbaum Associates, 2nd edition

Cook TD, Campbell DT (1979) Quasi-experimentation: Design & analysis issues for field settings. Houghton Mifflin. ISBN 9780395307908

Dvorzak M, Wagner H (2016) Sparse Bayesian modelling of underreported count data. Stat Model 16(1):24–46. https://doi.org/10.1177/1471082X15588398

Jaynes ET (1976) Confidence intervals vs bayesian intervals. In Harper WL, Hooker CA (eds) Foundations of probability theory, statistical inference, and statistical theories of science. D. Reidel Publishing Company, vol 2, pp 175–257. https://bayes.wustl.edu/etj/articles/confidence.pdf

Berger ED, Hollenbeck C, Maj P, Vitek O, Vitek J (2019) On the impact of programming languages on code quality: A reproduction study. ACM Trans Program Lang Syst 41(4):21:1-21:24. https://doi.org/10.1145/3340571

Wong WE, Gao R, Li Y, Abreu R, Wotawa F (2016) A survey on software fault localization. IEEE Trans Software Eng 42(8):707–740. https://doi.org/10.1109/TSE.2016.2521368

Fang H, Lamba H, Herbsleb J, Vasilescu B (2022) This is damn slick!" Estimating the impact of tweets on open source project popularity and new contributors. In: Proceedings of the 44th international conference on software engineering, ICSE. ACM

Feldt R, Shepperd M, Furia CA, Torkar R (2025) Causal systematic reviews – synthesizing scientific knowledge through causal links. In preparation

Fey CF, Tianyou H, Delios A (2023) The measurement and communication of effect sizes in management research. Manag Organ Rev 19(1):176–197. https://doi.org/10.1017/mor.2022.2

Fisher RA (1935) The design of experiments. Oliver & Boyd Edinburgh, Scotland, Edinburgh

Furia CA, Torkar R, Feldt R (2022) Applying Bayesian analysis guidelines to empirical software engineering data: The case of programming languages and code quality. ACM Transactions on Software Engineering and Methodology 31(3). https://doi.org/10.1145/3490953

Furia CA, Torkar R, Feldt R (2023) Towards causal analysis of empirical software engineering data: The impact of programming languages on coding competitions. ACM Trans Softw Eng Methodol 33(1). https://doi.org/10.1145/3611667

Gelman A, Carlin JB, Stern HS, Dunson DB, Vehtari A, Rubin DB (2014) Bayesian data analysis. CRC Press, 3 edition

Baah GK, Podgurski A, Harrold MJ (2010) Causal inference for statistical fault localization. In: Proceedings of the 19th international symposium on Software testing and analysis, pp 73–84

González-Barahona JM, Robles G (2012) On the reproducibility of empirical software engineering studies based on data retrieved from development repositories. Empir Software Eng 17:75–89

Greenland S, Morgenstern H (2001) Confounding in health research. Annu Rev Public Health 22:189–212. https://doi.org/10.1146/annurev.publhealth.22.1.189

Greenland S, Pearl J, Robins JM (1999) Causal diagrams for epidemiologic research. Epidemiology 10(1):37–48

Gren L, Torkar R, Feldt R (2017) Group development and group maturity when building agile teams: A qualitative and quantitative investigation at eight large companies. J Syst Softw 124:104–119. https://doi.org/10.1016/j.jss.2016.11.024

Halpern J (2015) A modification of the halpern-pearl definition of causality. In: Twenty-fourth international joint conference on artificial intelligence

Heyn H-M, Knauss E (2022) Structural causal models as boundary objects in ai system development. In: 1st International conference on AI engineering-software engineering for AI

Hespanhol L, Vallio CS, Costa LM, Saragiotto BT (2019) Understanding and interpreting confidence and credible intervals around effect estimates. Braz J Phys Ther 23(4):290–301. https://doi.org/10.1016/j.bjpt.2018.12.006

Hoekstra R, Morey RD, Rouder JN, Wagenmakers E-J (2014) Robust misinterpretation of confidence intervals. Psychon Bull Rev 13:1033–1037

Holt NE, Briand LC, Torkar R (2014) Empirical evaluations on the cost-effectiveness of state-based testing: An industrial case study. Inf Softw Technol 56(8):890–910. https://doi.org/10.1016/j.infsof.2014.02.011

Imbens GW, Wooldridge JM (2009) Recent developments in the econometrics of program evaluation. J Econ Liter 47(1):5–86. https://doi.org/10.1257/jel.47.1.5, https://www.aeaweb.org/articles?id=10.1257/jel.47.1.5
Imbens GW (2020) Potential outcome and directed acyclic graph approaches to causality: Relevance for empirical practice in economics. J Econ Liter 58(4):1129–1179. https://doi.org/10.1257/jel.20191597
Robins JM, Hernán MÁ, Brumback B (2000) Marginal structural models and causal inference in epidemiology. Epidemiology 11(5):550–560
Ioannidis JPA (2005) Why most published research findings are false. PLoS Med 2(8). https://doi.org/10.1371/journal.pmed.0020124
Hirano K, Imbens GW (2004) The propensity score with continuous treatments. In: Gelman A, Meng X-L (eds) Applied Bayesian modeling and causal inference from incomplete-data perspectives: an essential journey with donald rubin's statistical family, Wiley Series in Probability and Statistics, chapter Chapter 7, pp 73–84. Wiley. https://doi.org/10.1002/0470090456.ch7
Bollen KA, Pearl J (2013) Eight myths about causality and structural equation models. In Morgan SL (ed) Handbook of causal analysis for social research, pp 301–328. Springer Netherlands, Dordrecht. https://doi.org/10.1007/978-94-007-6094-3_15
Kosuke Imai and In Song Kim (2019) When should we use unit fixed effects regression models for causal inference with longitudinal data? Am J Polit Sci 63(2):467–490. https://doi.org/10.1111/ajps.12417
Krishnamurthi S, Vitek J (2015) The real software crisis: repeatability as a core value. Commun ACM 58(3):34–36. https://doi.org/10.1145/2658987
Laubach ZM, Murray EJ, Hoke KL, Safran RJ, Perng W (2021) A biologist's guide to model selection and causal inference. Proc R Soc B Biol Sci 288(1943):20202815. https://doi.org/10.1098/rspb.2020.2815
Lee S, Binkley D, Feldt R, Gold N, Yoo S (2021) Causal program dependence analysis. arXiv preprint arXiv:2104.09107
Levén W, Broman H, Besker T, Torkar R (2024) The Broken Windows Theory applies to technical debt. Empir Softw Eng 29(4):73. https://doi.org/10.1007/s10664-024-10456-6
Lin DY, Psaty BM, Kronmal RA (1998) Assessing the sensitivity of regression results to unmeasured confounders in observational studies. Biometrics 54(3):948–963
Liu Y, Mattos DI, Bosch J, Olsson HH, Lantz J (2022) Bayesian causal inference in automotive software engineering and online evaluation. arXiv preprint arXiv:2207.00222
D'Agostino McGowan L (2022) `tipr`: An R package for sensitivity analyses for unmeasured confounders. J Open Source Softw 7(77):4495. https://doi.org/10.21105/joss.04495
Madeyski L, Kitchenham B (2017) Would wider adoption of reproducible research be beneficial for empirical software engineering research? J Intell Fuzzy Syst 32(2):1509–1521. https://doi.org/10.3233/JIFS-169146
Jørgensen M, Dybå T, Liestøl K, Sjøberg DIK (2016) Incorrect results in software engineering experiments: How to improve research practices. J Syst Softw 116:133–145. https://doi.org/10.1016/j.jss.2015.03.065
Mathur MB, Smith LH, Ding P, VanderWeele TJ (2021) EValue: Sensitivity analyses for unmeasured confounding and other biases in observational studies and meta-analyses. https://doi.org/10.32614/CRAN.package.EValue
McElreath R (2020) Statistical rethinking: A Bayesian course with examples in R and Stan, 2nd edn. CRC Press, Florida, USA
McFarland D, McFarland H (2015) Big Data and the danger of being precisely inaccurate. Big Data Soc 2:12. https://doi.org/10.1177/2053951715602495
McShane BB, Gal D, Gelman A, Robert C, Tackett JL (2019) Abandon statistical significance. Am Stat 73(S1):235–245
Mehrabi N, Morstatter F, Saxena N, Lerman K, Galstyan A (2021) A survey on bias and fairness in machine learning. ACM Comput Surv 54(6). https://doi.org/10.1145/3457607
Menzies T, Shepperd M (2019) "Bad smells" in software analytics papers. Inf Softw Technol 112:35–47
Morasca S, Russo G (2001) An empirical study of software productivity. In: 25th Annual international computer software and applications conference. COMPSAC 2001, pp 317–322. https://doi.org/10.1109/CMPSAC.2001.960633
Open Science Collaboration (2015) Estimating the reproducibility of psychological science. Science 349(6251):aac4716. https://doi.org/10.1126/science.aac4716
Holland PW (1986) Statistics and causal inference. J Am Stat Assoc 81(396):945–960. https://doi.org/10.2307/2289064
Pearl J (1982) Reverend Bayes on inference engines: A distributed hierarchical approach. In: Proceedings of the Second AAAI conference on artificial intelligence, AAAI'82, pp 133–136. AAAI Press
Pearl J (2009) Causality: Models, reasoning and inference. Cambridge University Press, 2nd edition
Penzenstadler B, Torkar R, Martinez Montes C (2022) Take a deep breath: Benefits of neuroplasticity practices for software developers and computer workers in a family of experiments. Empir Softw Eng 27(4):98. https://doi.org/10.1007/s10664-022-10148-z

Tennant PWG, Murray EJ, Arnold KF, Berrie L, Fox MP, Gadd SC, Harrison WJ, Keeble C, Ranker LR, Textor J, Tomova GD, Gilthorpe MS, Ellison GTH (2021) Use of directed acyclic graphs (DAGs) to identify confounders in applied health research: review and recommendations. Int J Epidemiol 50(2):620–632. https://doi.org/10.1093/ije/dyaa213

Ray B, Posnett D, Filkov V, Devanbu P (2014) A large scale study of programming languages and code quality in Github. In: Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering, FSE 2014, pages 155–165, New York, NY, USA. Association for Computing Machinery. https://doi.org/10.1145/2635868.2635922

Rezaalipour M, Furia CA (2024) An empirical study of fault localization in Python programs. Empir Softw Eng 29(2):92

Wasserstein RL, Lazar NA (2016) The ASA statement on p-values: Context, process, and purpose. Am Stat 70(2):129–133. https://www.amstat.org/asa/files/pdfs/P-ValueStatement.pdf

Rubin DB (1974) Estimating causal effects of treatments in randomized and nonrandomized studies. J Educ Psychol 66(5):688–701

Sagan C (1986) Broca's brain: reflections on the romance of science. Random house publishing group, ISBN 9780345336897

Scholz M, Torkar R (2021) An empirical study of linespots: A novel past-fault algorithm. Softw Test Verif Reliab 31(8):e1787

Siebert J (2022) Applications of statistical causal inference in software engineering. arXiv preprint arXiv:2211.11482

Spirtes P, Zhang K (2016) Causal discovery and inference: concepts and recent methodological advances. In: Applied informatics, vol 3, pp 1–28. SpringerOpen

Splawa-Neyman J, Dabrowska DM, Speed TP (1990) On the application of probability theory to agricultural. Experiments essay on principles. Section 9. Stat Sci 5(4):465–472. https://doi.org/10.1214/ss/1177012031

Stuart EA (2010) Matching methods for causal inference: A review and a look forward. Stat Sci 25(1):1–21

Williams TC, Bach CC, Matthiesen NB, Henriksen TB, Gagliardi L (2018) Directed acyclic graphs: a tool for causal studies in paediatrics. Pediatr Res 84:487–493. https://doi.org/10.1038/s41390-018-0071-3

Torkar R, Furia CA, Feldt R, Gomes de Oliveira Neto F, Gren L, Lenberg P, Ernst NA (2022) A method to assess and argue for practical significance in software rngineering. IEEE Trans Software Eng 48(6):2053–2065. https://doi.org/10.1109/TSE.2020.3048991

Tsunoda M, Ono K (2014) Pitfalls of analyzing a cross-company dataset of software maintenance and support. In: 15th IEEE/ACIS international conference on software engineering, artificial intelligence, networking and parallel/distributed computing, SNPD 2014, Las Vegas, NV, USA, June 30 - July 2, 2014, pp 1–6. IEEE Computer Society. https://doi.org/10.1109/SNPD.2014.6888729

VanderWeele TJ (2017) On a square-root transformation of the odds ratio for a common outcome. Epidemiology 28(6):e58. https://doi.org/10.1097/EDE.0000000000000733

TJ VanderWeele and Peng Ding (2011) Sensitivity analysis in observational research: introducing the E-value. Ann Intern Med 167(4):268–274

Yule GU (1903) Notes on the theory of association of attributes in statistics. Biometrika 2(2):121–134. https://doi.org/10.1093/biomet/2.2.121

Vansteelandt S, Joffe M (2014) Structural nested models and g-estimation: The partially realized promise. Stat Sci 29(4):707–731. https://doi.org/10.1214/14-STS493

Vehtari A, Gelman A, Gabry J (2017) Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. Stat Comput 27(5):1413–1432. https://doi.org/10.1007/s11222-016-9696-4

Wang H, Wang H, Zhang H (2008) Software productivity analysis with CSBSG data set. In: International conference on computer science and software engineering, CSSE 2008, Volume 2: Software Engineering, December 12-14, 2008, Wuhan, China, pages 587–593. IEEE Computer Society. https://doi.org/10.1109/CSSE.2008.1178

Watanabe S (2010) Asymptotic equivalence of Bayes cross validation and widely applicable information criterion in singular learning theory. J Mach Learn Res 11:3571–3594

Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B (2012). Experimentation in software engineering Springer. https://doi.org/10.1007/978-3-642-29044-2

Zou D, Liang J, Xiong Y, Ernst MD, Zhang L (2021) An empirical study of fault localization families and their combinations. IEEE Trans Software Eng 47(2):332–347. https://doi.org/10.1109/TSE.2019.2892102