
Article:
Clark, A.G., Foster, M., Prifling, B. et al. (4 more authors) (2024) Testing causality in scientific modelling software. ACM Transactions on Software Engineering and Methodology, 33 (1). pp. 1-42. ISSN 1049-331X

https://doi.org/10.1145/3607184

Testing Causality in Scientific Modelling Software

ANDREW G. CLARK, Department of Computer Science, The University of Sheffield, United Kingdom
MICHAEL FOSTER, Department of Computer Science, The University of Sheffield, United Kingdom
BENEDIKT PRIFLING, Institute of Stochastics, Ulm University, Germany
NEIL WALKINSHAW, Department of Computer Science, The University of Sheffield, United Kingdom
ROBERT M. HIERONS, Department of Computer Science, The University of Sheffield, United Kingdom
VOLKER SCHMIDT, Institute of Stochastics, Ulm University, Germany
ROBERT D. TURNER, Department of Computer Science, The University of Sheffield, United Kingdom

From simulating galaxy formation to viral transmission in a pandemic, scientific models play a pivotal role in developing scientific theories and supporting government policy decisions that affect us all. Given these critical applications, a poor modelling assumption or bug could have far-reaching consequences. However, scientific models possess several properties that make them notoriously difficult to test, including a complex input space, long execution times, and non-determinism, rendering existing testing techniques impractical. In fields such as epidemiology, where researchers seek answers to challenging causal questions, a statistical methodology known as Causal Inference has addressed similar problems, enabling the inference of causal conclusions from noisy, biased, and sparse data instead of costly experiments. This paper introduces the Causal Testing Framework: a framework that uses Causal Inference techniques to establish causal effects from existing data, enabling users to conduct software testing activities concerning the effect of a change, such as Metamorphic Testing, *a posteriori*. We present three case studies covering real-world scientific models, demonstrating how the Causal Testing Framework can infer metamorphic test outcomes from reused, confounded test data to provide an efficient solution for testing scientific modelling software.

CCS Concepts: • **Computing methodologies** → **Model verification and validation**; *Causal reasoning and diagnostics*; • **Software and its engineering** → **Software testing and debugging**.

Additional Key Words and Phrases: Software Testing, Causal Inference, Causal Testing

ACM Reference Format:
Andrew G. Clark, Michael Foster, Benedikt Prifling, Neil Walkinshaw, Robert M. Hierons, Volker Schmidt, and Robert D. Turner. 2023. Testing Causality in Scientific Modelling Software. 1, 1 (June 2023), 43 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Authors' addresses: Andrew G. Clark, agclark2@sheffield.ac.uk, Department of Computer Science, The University of Sheffield, Regent Court, 211 Portobello, S1 4DP, Sheffield, United Kingdom; Michael Foster, m.foster@sheffield.ac.uk, Department of Computer Science, The University of Sheffield, Regent Court, 211 Portobello, S1 4DP, Sheffield, United Kingdom; Benedikt Prifling, benedikt.prifling@uni-ulm.de, Institute of Stochastics, Ulm University, Helmholtzstraße 18 , 89081, Ulm, Germany; Neil Walkinshaw, n.walkinshaw@sheffield.ac.uk, Department of Computer Science, The University of Sheffield, Regent Court, 211 Portobello, S1 4DP, Sheffield, United Kingdom; Robert M. Hierons, r.hierons@sheffield.ac.uk, Department of Computer Science, The University of Sheffield, Regent Court, 211 Portobello, S1 4DP, Sheffield, United Kingdom; Volker Schmidt, volker.schmidt@uni-ulm.de, Institute of Stochastics, Ulm University, Helmholtzstraße 18 , 89081, Ulm, Germany; Robert D. Turner, r.d.turner@sheffield.ac.uk, Department of Computer Science, The University of Sheffield, Regent Court, 211 Portobello, S1 4DP, Sheffield, United Kingdom.

1 INTRODUCTION

The use of scientific modelling software to model, simulate, and understand complex phenomena has become commonplace. Such systems have played a pivotal role in improving our scientific understanding across a wide range of phenomena and disciplines, and are increasingly used outside of academia. Governments, for example, make extensive use of scientific modelling software to simulate and evaluate various policies and interventions [75]. Perhaps most notably, this has included the use of epidemiological models to predict the impact of a number of COVID-19 mitigation measures [56, 102].

Testing such models is particularly challenging [51]. They typically have vast input spaces comprising hundreds of parameters, as well as complex output spaces. Executing large numbers of tests is often impossible, because each execution can require a significant amount of time and resource to execute. Compounding this issue further, scientific models are often non-deterministic, meaning developers must run each test case multiple times and observe the distribution of outputs. Furthermore, these systems are often developed by scientists with a limited amount of training as software engineers [53].

Collectively, these issues make it difficult (and sometimes impossible) to determine whether the output of a test case or modelling scenario is correct or not. This is referred to as the test oracle problem [11]. Instead, to determine whether a software system is fit for purpose, a tester generally corroborates evidence to investigate smaller, more specific relationships between inputs and outputs. By making changes to particular input parameters and observing changes to particular output variables, there is an implicit assumption that the input parameters in question somehow influence the computation (i.e. have a 'causal' effect) of the outputs.

In this paper we are specifically concerned with this intrinsic challenge: How can we test the (implicitly causal) input-output relationships in a system with a vast and complex input space, which may be non-deterministic and suffer from the test oracle problem, without the ability to resort to large numbers of test executions?

The challenge of analysing causal relationships in limited, noisy data instead of running costly experiments is well-established in the statistical context. In areas such as epidemiology, a powerful statistical methodology known as causal inference (CI) has been employed to answer causal questions that cannot be answered experimentally due to ethical concerns, such as *Does smoking cause lung cancer?* [28]. By incorporating domain knowledge about known causal relationships between variables (or absence thereof), CI can produce *estimands* that isolate the causal quantity of interest. That is, 'recipes' for analysing data in a causally-valid way. Conventional statistical methods can then be employed to quantify the presence (or absence) of specific causal relationships, correcting for bias in the data, without the need for experimental procedures.

This paper is motivated by the observation that CI and software testing share a common goal in many cases: to establish precise and salient causal relationships. Moreover, by viewing software testing through a causal lens, we can leverage well-established CI techniques that conceptually address several testing challenges presented by scientific models for causality-driven testing activities, such as metamorphic testing.

To this end, we introduce a testing framework that incorporates an explicit model of causality into the testing process, facilitating the direct application of CI techniques to software testing problems, such as metamorphic testing. To achieve this, we take a model-based testing (MBT) perspective [65], in which testing is based on a model of the expected behaviour of the system-under-test that typically either describes the allowed sequences of events or gives a formal relation between the inputs and outputs [46, 105]. Traditionally, MBT has focused on models expressed using state-based languages, such as finite state machines [60] and labelled transition systems [103], or models that define

the allowed input-output relationships using languages, such as Z [45] and VDM [31]. However, given the focus on causality in this paper, we require a model that specifies the expected *causal relationships* between system inputs and outputs. Here, we assume that such causal information is represented by a causal directed acyclic graph (DAG) [44, 78].

Our decision to incorporate causal DAGs into the testing process is motivated by two main factors. First, testing can be viewed as a causal activity in which the tester checks whether expected causal relationships hold; in order to automate this process, we require the expected causal relationships to be expressed. Second, the causal DAG is a lightweight and intuitive model that is widely used by domain experts in areas such as epidemiology and social sciences to make causal assumptions actionable and transparent [40, 101].

In this paper, we make three contributions. First, we introduce a conceptual framework that approaches software testing activities, such as metamorphic testing, as CI problems, and clarifies the components necessary to leverage state-of-the-art CI techniques. While previous work [10] has shown that CI is, generally speaking, a universally applicable technique, we believe we are the first to apply it to the software testing field in this way. Second, we provide a reference implementation of the framework that can form the basis for future CI-driven tools for testing scientific modelling software. Third, we conduct three case studies applying the proposed framework to real-world scientific models from different domains, evaluating its ability to predict metamorphic test outcomes from observational data.

The remainder of this paper is structured as follows. Section 2 provides a motivating example and necessary background. Section 3 introduces our conceptual framework that frames causality-driven testing activities as problems of CI. Section 4 then introduces our reference implementation of this framework, before demonstrating its application to three real-world scientific models in Section 5 and discussing the main findings and threats to validity in Section 6. Section 7 reviews related work, and Section 8 concludes the paper.

2 BACKGROUND AND PRELIMINARIES

This section defines the scope of the paper and introduces the main challenges associated with testing scientific modelling software, as outlined in Kanewala and Bieman's survey on the same topic [51]. We present these challenges in the context of a real-world, motivating example that is used as one of three case studies in Section 5. We then provide a background on model-based testing and, in particular, metamorphic testing [20], a known solution to some of these challenges. We conclude this section with a brief introduction to causal inference, the statistical methodology employed by the framework presented in Section 3.

2.1 Black-Box Software Systems

In this paper, we view and test software from a black-box perspective [71], focusing on the relationships between its inputs and outputs rather than its inner-workings and source code. More formally, in this paper, we conceptualise the system-under-test (SUT) as follows:

Definition 2.1. A *system-under-test* (SUT) is a software system comprising a set of input variables, I, and output variables, O, such that $I \cap O = \varnothing$. We consider inputs to be parameters whose values are set prior to execution that influence the resulting system behaviour. We consider outputs to be features of the system that can be measured at any point during or after execution without inspecting or modifying the source code.

Given our focus on causality in this paper, we provide an informal definition of causality in Definition 2.2. This follows from Pearl's characterisation of causation, which states that "variables earn causal character through their capacity to sense and respond to changes in other variables" [81].

Definition 2.2. We say that a variable $X = x$ *causes* a variable Y if there exists some value x' such that, had the value of X been changed to x', the value of Y would change in response.

Furthermore, we are primarily interested in scientific modelling software. Informally, we consider this to be any form of software that has a significant computational component and simulates, models, or predicts the behaviour of complex, uncertain phenomena to support policy and scientific decisions [51, 59]. We focus on this form of software as it typically possesses a number of challenging characteristics that preclude the application of many conventional testing techniques, but can be addressed by the framework introduced in Section 3. In the following section, we introduce a motivating example to familiarise the reader with these challenging properties.

2.2 Motivating Example: Covasim

Covasim [35, 56] is an epidemiological agent-based model that has been used to inform COVID-19 policy decisions in several countries [26, 55, 76, 92]. Given the critical applications of such scientific models, it is of paramount importance that they are tested to the best of our abilities. However, Covasim has a number of characteristics that make testing particularly challenging.

Covasim has a **vast and complex input space**, with 64 unique input parameters, 27 of which are complex objects characterised by further parameters. Furthermore, the **precise values for many of the inputs are unknown** and are instead described by a distribution, meaning that any given scenario can be simulated using a potentially intractable number of input configurations.

Covasim also suffers from **long execution times and high computational costs**. Non-trivial runs of Covasim can take hours and accumulate large amounts of data. To compound this issue further, the model is also **non-deterministic**: running the same simulation parameters multiple times (with a different seed) will yield different results, meaning that each modelling scenario must be simulated several times to observe a distribution of outcomes.

Additionally, Covasim encounters the oracle problem: for most modelling scenarios, **the precise expected output is unknown**. This makes Covasim a traditionally "untestable" [110] system as it is difficult to determine whether the output of a given test is correct.

Despite these challenges, Covasim features a mixture of unit, integration, and regression tests achieving 88% code coverage[1]. However, many of these tests lack a test oracle and appear to rely on the user to determine correctness instead. For example, the vaccine intervention has two tests [34] that instantiate and run the model with two different vaccines and plot the resulting model outputs on a graph for manual inspection.

While the existing vaccination tests reveal the difference in outcome *caused* by changing from one vaccine to another, the experimental approach employed would not scale well if the tester wanted to test more general properties that cover larger value ranges. For example, tests covering multiple versions of vaccine (Pfizer, Moderna, etc.) and outcomes (infections, hospitalisations, etc.). However, this is not a criticism of Covasim, but a statement that conventional testing techniques are impractical for testing scientific modelling software. Hence, there is a clear need for testing techniques more sympathetic to their challenging characteristics.

2.3 Model-Based Testing

An approach that is often used to test black-box systems is model-based testing [14]. The main principle behind model-based testing is to provide a model that captures the expected behaviour of the SUT [104]. Such a model incorporates

[1]Code coverage obtained from commit 7da3bc4.

invaluable domain expertise and can form the basis for test generation, with work in this area going back to the 1950s [65]. In addition, if the model has formal semantics, testing can be represented as a process in which one compares the behaviour of two models: the known specification model M and an unknown model N that represents the behaviour of the SUT. It is then possible to reason about the effectiveness of testing [36, 103]. Note that since a model describes the expected behaviour of the SUT, it can also form the basis of a test oracle, and this is at least implicit in most MBT work [36, 103].

For testing black-box systems (i.e. where the internal workings are unknown to the test developer), an appropriate model will typically specify formal relations between the inputs and outputs of the SUT. For example, pre/post models can be defined in various modelling languages, such as Z [96] and B [16], that model a system as a collection of variables and captures the expected behaviour in terms of pairs of pre-conditions and post-conditions [104]. In this way, testers use their domain expertise to specify how they expect the SUT to respond under different settings.

However, for complex software like Covasim that suffers from the test oracle problem [11], it is seldom possible to specify the expected outputs or post-conditions corresponding to a particular set of inputs or pre-conditions. As discussed in Section 2.2, this is partly due to the exploratory nature of Covasim that makes it difficult (if not impossible) to establish what 'correctness' looks like. This is typically the case for any form of scientific software primarily used to predict or simulate future events, such as meteorological software for predicting the weather. Under such circumstances, the domain expertise needed to specify a model of the expected behaviour are fundamentally unattainable, preventing the tester from capturing static input-output relations, such as pre/post models, a priori.

One solution that effectively avoids the oracle problem and has been advocated as a technique for testing scientific software [51] is *metamorphic testing* [20]. The basic idea is to model the expected behaviour of the SUT as so-called *metamorphic relations* that describe the expected change in output in response to a specific *change* in input. For example, to test an implementation of sin, we may assert that $\forall x.\ \sin(x) = \sin(\pi - x)$. These relations provide a means of generating test cases and validating the observed behaviour [93]. By stating the expected behaviour in terms of *changes* to inputs and outputs, we can test the system without knowing the precise expected outcome corresponding to some inputs.

Statistical metamorphic testing (SMT) [42] generalises this to non-deterministic systems, which produce different outputs when run repeatedly under identical input configurations. Rather than comparing outputs directly, the SUT is run multiple times for each input configuration and statistical tests are performed on the corresponding distributions of outputs. However, the potentially high computational costs involved in this process are a major limitation to the applicability of SMT to scientific models.

2.4 Causal Inference

The framework we present in Section 3 uses a family of statistical techniques, known as causal inference (CI), designed to make claims about causal relationships between variables [52]. Our goal is to use this family of techniques to provide an efficient method for testing scientific software. Here we provide a brief introduction to the essential notions of CI used in this work. For a more comprehensive overview, we refer the reader to [44, 79].

2.4.1 Preliminaries. Causality is often presented in terms of the "ladder of causality" [82], which groups different tasks into three 'rungs': Rung one is *observation and association* as per traditional statistical methods; Rung two is *intervention*, which imagines the effects of taking particular actions: "What if I do...?", and rung three is *counterfactual*, which imagines the effects of retrospective actions: "What if I had done...?".

Traditional statistical approaches are limited to rung one. By simply observing the association between variables (in our case input and output variables), without systematically controlling the selection of values or resorting to additional domain knowledge, it is impossible to answer fundamentally *causal* questions [79]. This problem is commonly captured by the adage: "correlation does not imply causation".

CI enables us to estimate and quantify causal effects in order to make claims about causal relationships [52]. Informally, the causal effect of a treatment T on an outcome Y is the change in Y that is caused by a specific change in T [82]. In this context, a *treatment* is a variable that represents a particular action or intervention, such as changing a line of code, and an *outcome* is an observable feature or event, such as the occurrence of a fault.

One of the main challenges underlying CI is the design of experiments or statistical procedures that mitigate sources of bias to isolate and measure causality (rungs two and three) as opposed to association (rung one). In fields such as medicine, randomised control trials (RCTs) are often considered as the gold standard approach for CI [17]. RCTs mitigate sources of bias by randomly assigning subjects to either the treatment or control group [54]. However, there are many situations in which RCTs cannot be performed due to ethical or practical reasons [2].

Where RCTs cannot be performed, researchers often turn to observational data and statistical models as means for conducting CI. At a high level, this observational approach to CI can be broken down into two tasks: identification and estimation. Identification involves identifying sources of bias that must be adjusted for statistically in order to obtain a causal estimate. Estimation is the process of using statistical estimators, adjusted for the identified biasing variables, to estimate the causal effect.

2.4.2 Metrics. Several metrics can be used to measure causal effects. Perhaps the most desirable is the *individual treatment effect* (ITE), which describes the effect of a given treatment on a particular individual. In the majority of cases, however, individual-level inferences are unattainable due to the *fundamental problem of causal inference* [47]; namely that, for a given individual, it is usually only possible to observe the outcome of a single version of treatment (e.g. an individual either takes an aspirin for their headache or does not).

To address this, researchers typically turn to population-level causal metrics, such as the *Average Treatment Effect* (ATE) [44]:

$$\text{ATE} = \sum_{z \in Z} \mathbb{E}[Y \mid X = x_t, Z = z]P(Z = z) - \sum_{z \in Z} \mathbb{E}[Y \mid X = x_c, Z = z]P(Z = z)$$

The ATE quantifies the average additive change in outcome we expect to observe in response to changing some treatment variable X from the *control value* x_c to the *treatment value* x_t, while adjusting for all biasing variables Z. However, in some instances, it is desirable to refine our inferences to specific sub-populations defined by some notable characteristic. To this end, the conditional ATE (CATE) can be obtained by applying the ATE to specific sub-populations of interest [1].

An alternative causal metric is the *Risk Ratio* (RR) [44]:

$$\text{RR} = \frac{\sum_{z \in Z} \mathbb{E}[Y \mid X = x_t, Z = z]P(Z = z)}{\sum_{z \in Z} \mathbb{E}[Y \mid X = x_c, Z = z]P(Z = z)}$$

The RR captures the multiplicative change in an outcome Y caused by changing the treatment variable X from the control value x_c to the treatment value x_t while adjusting for all biasing variables Z.

Other effect metrics such as the *odds ratio* (OR) and the *effect of treatment on the treated* (ATT) also exist but fall outside the scope of this paper. Furthermore, to quantify uncertainty, effect measures are typically accompanied by 95% confidence intervals that quantify the interval within which we are 95% confident the true estimate lies [74].

2.5 Causal DAGs

CI generally depends on domain expertise and causal assumptions that cannot be tested in practice [89]. Given that different domain experts may make different assumptions about the same problem and that these may lead to different results, it is essential that all assumptions are made transparent. To this end, causal DAGs provide an intuitive graphical method for communicating the causal assumptions necessary to solve CI problems [78]. Formally, a causal DAG is defined as follows [44]:

Definition 2.3. A causal DAG G is a directed acyclic graph (DAG) $G = (V, E)$ comprising a set of nodes representing random variables, V, and a series of edges, E, representing causality between these variables, where:

(1) The presence/absence of an edge $V_i \rightarrow V_j$ represents the presence/absence of a direct causal effect of V_i on V_j.
(2) All common causes of any pair of variables on the graph are themselves present on the graph.

In Figure 1, (X), (Y), and (Z) are nodes representing *random variables*, which, in this context, are variables that can take different values for different individuals (e.g. people or software executions). We say that X is a *direct cause* of Y because there is an edge from X directly into Y. We refer to Y as a *descendant* of Z and X because there is a sequence of edges, known as a *path*, such that, if you follow the direction of those edges, you can reach Y from Z. That is, $\mathsf{Z} \rightarrow \mathsf{X} \rightarrow \mathsf{Y}$.

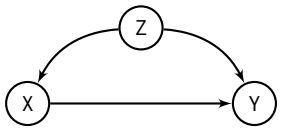

Fig. 1. An example causal DAG for the causal effect of X on Y confounded by Z.

As mentioned in the previous section, in order to estimate the causal effect of X on Y, we need to identify and adjust for all variables that bias the relationship $\mathsf{X} \rightarrow \mathsf{Y}$. Using a causal DAG, we can achieve this automatically by applying a pair of graphical tests, the *back-door criterion* and *d-separation*, which are formally defined as follows:

Definition 2.4. A path p is *blocked* or *d-separated* by a set of variables Z if and only if at least one of the following conditions hold [80]:

(1) p contains a chain $i \rightarrow k \rightarrow j$ or a fork $i \leftarrow k \rightarrow j$ where $k \in Z$.
(2) p contains a collider $i \rightarrow k \leftarrow j$ where $k \notin Z$ and for all descendants k' of k, $k' \notin Z$.

Definition 2.5. A set of variables Z is said to satisfy the *back-door criterion* relative to an ordered pair of variables (X, Y) if both of the following conditions hold [80]:

(1) No variable in Z is a descendant of X.
(2) Z blocks every path between X and Y that contains an arrow into X.

A set of variables Z is said to be a *sufficient adjustment set* relative to a pair of variables (X, Y) if adjusting for Z blocks all back-door paths between X and Y. Conceptually, this corresponds to a set of variables that, once adjusted for, mitigate all known sources of bias and that is therefore capable of isolating the *causal effect* of interest. For example, in Figure 1, Z satisfies the back-door criterion relative to (X, Y) because Z blocks every path between X and Y with an

arrow into X. Therefore, we can endow the ATE of X on Y with a causal interpretation and estimate its value directly using the following closed-form statistical expression:

$$\sum_{z \in Z} \mathbb{E}[Y \mid X = 1, Z = z] P(Z = z) - \sum_{z \in Z} \mathbb{E}[Y \mid X = 0, Z = z] P(Z = z)$$

Overall, causal DAGs provide a principled and automated approach for designing statistical 'recipes' capable of measuring causal relationships and endowing statistical measures with causal interpretations. In the following section, we introduce a framework that facilitates the application of this approach to the testing of scientific modelling software. Furthermore, we opt to use graphical CI over other CI frameworks, such as potential outcomes [90] or structural equation modelling [58], as it provides a transparent and intuitive way to both specify and test causal relationships, without necessarily requiring users to know their precise functional form.

3 CAUSAL TESTING FRAMEWORK

This section introduces the Causal Testing Framework (CTF): a conceptual framework that approaches causality-driven testing activities as CI problems. That is, testing activities that intend to establish the (inherently causal) relationship between inputs and outputs, such as metamorphic testing. By framing testing activities in this way, it is possible to leverage CI techniques to make strong claims about causal relationships between inputs and outputs, and to do so in an efficient manner by exploiting data from previous test executions.

In the remainder of this section, we define four key components of our causal testing framework: specifications, programs, tests, and oracles [97], giving an example using Covasim (see Section 2) for each component. We also provide informal guidance for constructing causal DAGs and examine the relationship between the CTF and metamorphic testing.

3.1 Causal Specification

In the CTF, our primary aim is to test scientific models in terms of the effects of interventions. Given the diverse range of possible scenarios that a typical scientific model can simulate, we further focus on testing individual modelling scenarios. We define a modelling scenario as a series of constraints placed over a subset of the SUT's (see Definition 2.1) input variables that characterise the scenario of interest. Therefore, in the causal testing framework, the set of programs are programs that implement modelling scenarios $\mathcal{M}$ (Definition 3.1).

Definition 3.1. A *modelling scenario* $\mathcal{M}$ is a pair (X, C) where X is a non-strict subset of the model's input variables and C is a set of constraints over realisations of X, which may be empty.

The expected behaviour of scientific modelling software in a given scenario depends on a series of underlying modelling assumptions. It is therefore essential that such modelling assumptions are made transparent and readily available, particularly for the purposes of testing. Indeed, past investigations into modelling failures have highlighted the importance of transparency and accountability [75]. In the same vein, causal testing requires an explicit record of causal assumptions to enable the transparent and reproducible application of graphical CI techniques. To this end, we use a causal DAG that captures causality amongst a subset of the SUT's input and outputs. Therefore, we define a *causal specification* (Definition 3.2) as a pair comprising a modelling scenario ($\mathcal{M}$) and a causal DAG ($\mathcal{G}$).

Definition 3.2. A *causal specification* is a pair $\mathcal{S} = (\mathcal{M}, \mathcal{G})$ comprising a modelling scenario $\mathcal{M}$ and a causal DAG $\mathcal{G}$ capturing the causal relationships amongst the inputs and outputs of the SUT that are central to the modelling scenario.

Example 3.3. Consider a scenario in Covasim where we want to test the effect of prioritising the elderly for vaccination V on the total vaccine doses administered N_D, total vaccinated agents N_V, maximum number of doses per agent M_D, and cumulative infections I. Further, let us restrict our simulation length to 50 days, the initial number of infected agents to 1000, and the population size to 50,000. Our modelling scenario is then characterised by the constraints $\{\texttt{days} = 50, \texttt{pop\_size} = 50000, \texttt{pop\_infected} = 1000\}$, and the causal DAG is the set of edges $\{V \rightarrow N_V, V \rightarrow N_D, V \rightarrow I\}$. Note the absence of edge $V \rightarrow M_D$. Here we are asserting that V may cause a change in N_V, N_D, and I, but should cause no change to M_D. This is because at most two doses of the vaccine are administered to each at agent so changing the target population should not affect this.

3.2 Constructing Causal DAGs

In the testing context, causal DAGs offer a flexible, lightweight means by which to capture potential causal relationships between inputs and outputs. Here we present a set of guidelines for constructing the graph (informed by our experience with the case studies).

We start by constructing a complete directed graph over the set of inputs and output: $I \cup O$. Then, to simplify this structure, we apply the following assumption:

Assumption 1. *Outputs cannot cause inputs.*

Assumption 1 follows from temporal precedence (that a cause must precede its effect) [83] and the observation that, in a given test execution, outputs temporally succeed inputs. This enables us to delete all edges from outputs to inputs.

Then, in many cases, we can also apply the following assumption to remove all edges from inputs to inputs:

Assumption 2. *Inputs cannot cause changes to the values of other inputs and, therefore, cannot share causal relationships.*

As stated in Definition 2.1, in this paper, we assume that all inputs are assigned their values prior to execution. Under this characterisation, changes to the value of one input cannot *physically* affect another input's value and, therefore, inputs cannot share causal relationships. Of course, there are caveats to this; if a system has input validation, for example, the assignment of one input's value may *physically* restrict which values can be selected for a second input. Note that, in such cases, our framework is still applicable, but the user would have to consider more edges manually to construct their DAG.

This leaves us with the following forms of potential causal relationships to consider: $I \rightarrow O$ and $O \rightarrow O$ (and $I \rightarrow I$ if Assumption 2 cannot be applied). Output to output causality may occur in software where an earlier output is used in the computation of a later output. For example, in a weather forecasting model, a prediction of the weather in three days time is affected by the weather predicted for one and two days time.

This is the point at which the tester's domain knowledge is fed into the model, by pruning edges where they are certain that there is no causal relationship (see Definition 2.2 for an informal definition of causality). We recommend following this approach of pruning edges from a complete directed graph over adding edges to an initially empty graph, as the absence of an edge carries a stronger assumption than the presence of one [101]. This follows from the fact that the presence of an edge states that there exists *some* causal relationship, whereas the absence of an edge states that there is *precisely* no causal relationship.

3.3 Causal Testing

Causal testing draws its main inspiration from CI, which focuses on the effects of *interventions* on *outcomes*. In this context, an intervention manipulates an input configuration in a way that is expected to *cause* a specific outcome to change. Here, we refer to the pre-intervention input configuration as a *control* and the post-intervention input configuration as a *treatment*. A causal test case then specifies the expected change in outcome caused by this intervention (i.e. the expected causal effect). When phrased this way, causal tests bear a remarkable similarity to metamorphic tests, highlighting the fact that, at its core, metamorphic testing can be viewed as an inherently a causal activity. We explain this relationship further in Section 3.4.

Definition 3.4. An *intervention* $\Delta : \mathcal{X} \rightarrow \mathcal{X}'$ is a function which manipulates the values of a subset of input realisations.

Definition 3.5. A *causal test case* $\mathcal{T}$ is a 4-tuple $(\mathcal{M}, \mathcal{X}, \Delta, \mathcal{Y})$ that captures the expected causal effect, $\mathcal{Y}$, of an intervention, Δ, made to an input valuation, $\mathcal{X}$, on some model outcome in the context of modelling scenario $\mathcal{M}$. The expected causal effect $\mathcal{Y}$ is an informal expression of some change in outcome that is expected to be caused by executing $\mathcal{T}$. We refer to the input realisation $\mathcal{X}$ as the control input configuration.

Example 3.6. Continuing with our vaccination example, suppose we want to create a causal test case that investigates the effect of switching vaccine from `Pfizer` to an age-restricted version (`Pfizer′`) on only the maximum number of doses per agent M_D. We can start by using the modelling scenario outlined in the previous example and then specify our control input configuration as the input realisation $\mathcal{X} = \{\texttt{vaccine} = \texttt{Pfizer}\}$. We then define an intervention that takes the control input configuration and replaces the vaccine with the age-restricted version: $\Delta(\mathcal{X}) = \mathcal{X}[\texttt{vaccine} := \texttt{Pfizer}']$. We complete our causal test case by specifying the expected causal effect, $\mathcal{Y}$: the intervention should cause no change to M_D and we therefore expect that the ATE will be zero.

Finally, we must consider the test oracle: the *procedure* used to determine whether the outcome of a causal test case ($\mathcal{T}$) is correct (i.e. whether it realises the expected causal effect $\mathcal{Y}$). In the context of causal testing, the oracle must ascertain the correctness of causal estimates relative to a modelling scenario ($\mathcal{M}$). Therefore, we refer to our oracle as a causal test oracle (Definition 3.4).

Definition 3.7. A *causal test oracle* $\mathcal{O}$ is a procedure, such as an assertion, that determines whether the outcome of a causal test case $\mathcal{T}$ is correct or incorrect. This procedure checks whether the application of the intervention Δ to the control input configuration $\mathcal{X}$ has caused the expected causal effect $\mathcal{Y}$ in the context of modelling scenario $\mathcal{M}$.

Example 3.8. Continuing with our Covasim example, for the causal test case $\mathcal{T}$ defined in the previous example, our causal test oracle must check whether applying the intervention (i.e. replacing the `Pfizer` vaccine with an age-restricted version `Pfizer′`) has no effect on M_D, as specified by the expected causal effect $\mathcal{Y}$. We can implement this test oracle as the following assertion: $\texttt{ATE}_{\texttt{M}_\texttt{D}} = \texttt{0}$. This checks whether the change in M_D caused by the intervention (ATE_{M_D}) is zero, as expected.

Notice the subtle difference between the expected causal effect, $\mathcal{Y}$, of the causal test case, $\mathcal{T}$, and the causal test oracle, $\mathcal{O}$: the former is a statement of the *expected test outcome* while the latter is the *actual procedure* used to check whether the anticipated outcome holds. We make this distinction with the transparency of the causal testing process in mind, avoiding situations where two testers may implement the procedure to ascertain the validity of a given causal test case in different ways, potentially leading to different test outcomes. In other words, the CTF considers the expected outcome ($\mathcal{Y}$) and the procedure used to check this has been realised ($\mathcal{O}$) as separate entities that carry equal importance.

Any discrepancy between the test result and the expected outcome revealed by the test oracle implies one of two problems: (i) the implementation contains a bug or an error, or (ii) the underlying causal knowledge is incorrect. It follows that causal testing lends itself to an iterative testing process [68], whereby the user inspects the source code to explain any identified discrepancies and, if no bugs are found, reviews the causal DAG to check if the underlying science is correct.

Collectively, the components of the CTF enable the application of graphical CI techniques to testing activities that concern the causal effect of some intervention. In theory, the CTF should therefore provide the following advantages over existing solutions:

(1) The ability to derive test outcomes *experimentally*[2] (by strategic model executions that isolate a particular cause-effect relationship by design) and *observationally* (by applying CI techniques to past execution data).
(2) The ability to identify and adjust for confounding bias in observational data using a causal DAG. From a testing perspective, this effectively relaxes the experimental conditions ordinarily required to reach causal conclusions. Namely, the need for carefully controlled, unbiased test data.
(3) The ability to derive *counterfactual* test outcomes using appropriate statistical models. This would enable testers to infer how the model would likely behave, had it been run under a different parameterisation. Therefore, where practical constraints preclude further executions of the SUT, counterfactual inference can offer a cost-effective alternative.

In Section 5, we apply the CTF to a series of real-world scientific models to understand how a modeller can leverage these advantages in a testing context to improve the efficiency and applicability of metamorphic testing; a state-of-the-art approach for testing scientific modelling software.

3.4 Relationship to Metamorphic Testing

At a high level, the CTF and metamorphic testing share the same objective: to evaluate the *effect* caused by making a change to some input.

Metamorphic testing provides a means of generating "follow-up test cases" using metamorphic relations which should hold over a number of different parameter values [11, 93]. In contrast to typical program invariants, which must hold for every execution of a given program, metamorphic relations hold between different executions. In other words, they investigate the effect of a change (or *intervention* in causal language) on an input. This is a key similarity between causal testing and metamorphic testing.

In this sense, metamorphic tests can be thought of as quasi-experiments[3] designed to answer causal questions about the SUT. For example, a metamorphic test for our property of the sin function in Section 2 that $\forall x.\ \sin(x) = \sin(\pi - x)$ can be thought of as a quasi-experiment that confirms whether changing the input from $X = x$ to $X = \pi - x$ causes no change to the output. That is, there should be *no causal effect*. This synergism suggests that metamorphic testing can be re-framed and solved as a problem of CI and, therefore, benefit from its advantages. To this end, in Section 5, we demonstrate how the CTF can conduct metamorphic testing using CI techniques.

One advantage of causal testing over traditional metamorphic testing is that causal testing does not necessarily require dedicated test runs of the system to be performed if sufficient test data already exists. Even (and especially) if this data is biased, CI can account for this, meaning that testing can be performed on systems which cannot be

[2]We use the term 'experimental' loosely here; the CTF performs a quasi-experiment in which the SUT is executed with a pair of input configurations that isolate the causal effect of the intervention on the output. Specifically, the SUT is executed twice: once using the pre-intervention configuration and once using the post-intervention configuration. This is repeated multiple times for non-deterministic systems.
[3]We liken metamorphic tests to quasi-experiments rather than controlled experiments as they lack an explicit randomisation mechanism.

tested for reasons of practicality. Furthermore, systems can be tested retroactively, enabling concerns about a model's correctness to be investigated even after the model has been run. This is potentially advantageous in the context of scientific models, where their integrity and correctness can be called into question years after policies based on their output have already been made. In such situations, the DAG makes clear the assumptions made about the functionality of the model so adds weight to any conclusions made.

4 CTF REFERENCE IMPLEMENTATION

This section provides an overview of our open-source Python reference implementation of the Causal Testing Framework (CTF)[4], comprising over 4000 lines of Python code, and outlines four stages of the CTF workflow: Specification, Test Cases, Data Collection, and Testing.

4.1 Causal Specification

To begin causal testing, we form a causal specification (Definition 3.2), comprising two components: a modelling scenario and a causal DAG. We form the modelling scenario by specifying a set of constraints over the inputs that characterise the scenario-under-test, such as $x_1 < x_2$. Next, we specify our causal DAG using the DOT language [32], in which graphs are expressed as a series of edges, such as $x_1 \rightarrow x_2$, following the guidelines outlined in Section 3.2.

4.2 Causal Test Case

Now that we have a causal specification, we define a causal test case that describes the intervention whose effect we wish to test. In our reference implementation, a causal test case is an object that requires us to specify a control input configuration, a treatment input configuration, and the expected causal effect. In the following steps, this information will enable us to collect appropriate test data (Data Collection), design quasi-experiments isolating the causal effect of interest within this data, and define test oracles that ascertain whether the expected causal effect is observed (Causal Testing).

4.3 Data Collection

After creating a causal specification and causal test case, the next step is to collect data corresponding to the modelling scenario. We can achieve this either (quasi-)experimentally (in situations where we are able to directly execute the SUT) or observationally (in situations were we are not able to execute the SUT, but are instead able to draw upon prior execution data).

4.3.1 Experimental Data Collection. Experimental data collection executes the model *directly* under both the control and treatment input configuration to isolate the causal effect of the intervention. To this end, our reference implementation provides an abstract experimental data collector class, requiring us to implement one method that executes our model with a given input configuration. This method enables the CTF to run the model under the conditions necessary to isolate causality directly.

4.3.2 Observational Data Collection. Since it is often infeasible to run models a statistically significant number of times, we also provide the option to use observational, existing test data. This data may not meet the experimental conditions necessary to isolate the causal effect and thus may contain biases that lead purely statistical techniques astray. However,

[4]https://github.com/CITCOM-project/CausalTestingFramework

by employing graphical CI techniques, the CTF can identify and mitigate bias in the data, providing an efficient method for testing scientific models *a posteriori*.

There are two caveats to this. First, the causal DAG must be correctly specified. While this is not generally verifiable, several techniques exist that can quantify the sensitivity of casual estimates to unobserved confounding, including the robustness value [25] and the e-value [106]. These techniques could be employed to justify that the DAG-informed adjustment set yields causal estimates that are robust to missing confounders. Second, the observational data must be consistent with the constraints of the causal specification. To this end, our reference implementation includes an observational data collector class that takes a CSV file of existing test data as input and uses the Z3 theorem prover [29] to identify and remove any executions of the SUT that violate constraints. By execution, we refer to an individual run of the SUT with some set of inputs that produces some set of outputs. We assume the CSV file comprises a row for each such execution, with a column for each input and output value. Next, we describe how the CTF infers test outcomes from this data.

4.4 Causal Testing

Given a causal test case, testing is carried out in two stages: causal inference (CI) and applying the test oracle.

4.4.1 Causal Inference. To infer the causal effect of interest, our reference implementation applies the two steps of CI outlined in Section 2: identification and estimation. For identification, the CTF algorithmically identifies an adjustment set (see Section 2.4) for the causal effect of interest. Then, for estimation, we design an appropriate estimator that adjusts for the identified adjustment set, and apply the estimator to our data to estimate the desired causal metric (e.g. ATE or RR, see Section 2). To this end, our reference implementation provides regression and causal forest [108] estimators which can be customised to add additional features such as squared and inverse terms to change the shape of the model. In addition, the CTF includes an abstract estimator class that enables users to define their own estimators. This step outputs a causal test result containing the inferred causal estimate for the desired causal metric (e.g. ATE or RR, see Section 2.4) and 95% confidence intervals. The user is, of course, free to relax their confidence intervals should they wish to obtain a more precise estimate with a higher level of associated risk, or vice versa.

4.4.2 Test Oracle. After applying CI, all that remains is the test oracle procedure. That is, to check whether the causal test results match our expectations. For this purpose, our reference implementation provides several test oracles that check for positive, negative, zero, and exact effects. Alternatively, to handle more complex outputs, a user can specify a custom oracle that ascertains whether a causal test result should pass or fail.

Now that we have discussed the workflow of our CTF reference implementation, in the following section, we demonstrate its application to three vastly different real-world scientific models.

5 CASE STUDIES

This section applies the Causal Testing Framework (CTF) to four testing scenarios covering three real-world scientific models from different domains, approaching (statistical) metamorphic testing as a CI problem. Our goal here is to conduct a series of *evaluative* case studies [86] that appraise the CTF with respect to three attributes: *accuracy*, *efficiency*, and *practicality*. Here, we do not aim to make generalisable conclusions, but to evaluate the CTF with respect to each of these attributes within the context of each subject system. To this end, across our case studies, we corroborate evidence to collectively answer the following research questions:

RQ1 (Accuracy): Can we reproduce the results of a conventional MT/SMT approach by applying the CTF to observational data? As mentioned in Section 1, CI is a generally applicable technique [10] promising the ability to infer test outcomes from existing data that is potentially confounded. In the context of testing scientific software, this approach has the potential to reduce the overhead associated with SMT by enabling the inference of metamorphic test outcomes from existing execution data. This is in contrast to a conventional approach which may require numerous potentially costly executions.

In this research question, we consider whether the CTF is able to predict metamorphic test outcomes from observational data with sufficient accuracy to make *actionable inferences*. By actionable inferences, we refer to predicted outcomes that provide a truthful and meaningful insight into the actual behaviour of the SUT.

RQ2 (Efficiency): In terms of the amount of data required, is the CTF more cost effective than a conventional MT/SMT approach? In practice, the utility and applicability of the CTF depends on the amount of observational data required to make actionable inferences. Hence, for the CTF to be considered a useful tool and a viable alternative to conventional MT and SMT approaches, it must be capable of making actionable inferences using no more data than is required by a conventional approach.

To this end, in order to understand the efficiency and therefore utility of the proposed approach, this research question investigates the relationship between the amount of observational data and the accuracy of insights provided by the inferred metamorphic test outcomes.

RQ3 (Practicality): What practical effort is required from the tester to conduct MT/SMT using the CTF?. The CTF requires causal knowledge and domain expertise that, in turn, depend on human effort. This human effort cannot be overlooked. Hence, in order to determine whether the technique can be considered practical and applicable, it is necessary to investigate the trade-off between the human cost and the benefits offered by the CTF.

In this research question, we provide a qualitative account of the human effort involved in applying the CTF to each case study.

In the remainder of this section, we cover each of the three case studies in accordance to the following high-level structure. First, we describe the characteristics of the subject system and our justification for selecting it. We then provide a brief overview of the testing activity (the broad testing objective) and the process of acquiring data for analysis. Following this, we describe the application of the CTF and analyse the generated data. We conclude by analysing the outcomes and answering the relevant research questions. The contribution of each case study to the research questions will be highlighted throughout the case studies and the collective findings will be discussed in Section 6.

5.1 Poisson Line Tessellation Model

In this case study, we use the CTF to conduct statistical metamorphic testing (SMT) on a Poisson Line Tessellation (PLT) model. This model is of particular significance as it formed the case study of the paper that introduced the concept of SMT [42]. As such it provides an ideal basis upon which to compare and contrast our CI-led approach against the conventional SMT approach. In particular, we show how the CTF can infer the same metamorphic test outcomes as the traditional SMT approach but from significantly fewer model executions. The code for this case study can be found in our open source repository[5].

[5]https://github.com/CITCOM-project/CausalTestingFramework/tree/683e6c55/examples/poisson-line-process

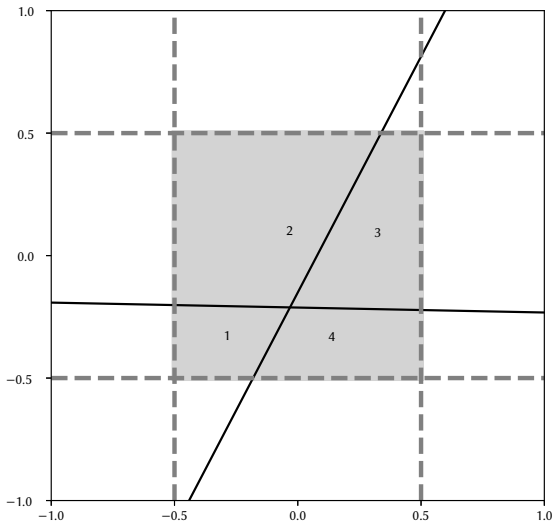

Fig. 2. A tessellation generated by the PLT model with a width (W), height (H), and intensity (I) of 1. There are two lines which intersect the sampling window (L_t, highlighted in grey). The intersection of these lines forms four polygons in total (P_t).

5.1.1 Subject System. The PLT model uses a Poisson process to generate a series of lines that are positioned and oriented at random within a given sampling window to form a tessellation. While the behaviour of this model is predominantly random by design, it can be configured using three numerical input parameters to produce tessellations with predictable properties. In order to test these properties, we extract four numerical outputs from the resulting tessellation.

We selected this model because it has been the subject of prior research on SMT [42] and has a number of well-characterised input-output relationships. In addition, Poisson process models are commonly used to model random processes for a range of important applications, including simulating road networks [21, 66] and modelling photon arrival in 3D imaging [94]. It is the stochastic yet predictable behaviour of Poisson process models that make them an interesting but difficult subject to test.

We now describe the behaviour of the PLT model, referring to the example tessellation in Figure 2. The PLT model has three positive floating point input parameters: the width W and height H of a sampling window (shaded in grey in Figure 2), and the intensity I of the Poisson process. Informally, the intensity parameter controls the average rate at which lines are placed. Given these inputs, the model generates a set of straight lines that intersect the origin-centred sampling window by drawing from a Poisson process on $[0, \infty) \times [0, 2\pi)$[6], where the orientation is uniformly distributed on $[0, \pi]$. The model then outputs the total number of lines intersecting the sampling window, L_t, and the number of polygons formed by the intersecting lines, P_t.

In Figure 2, for example, the inputs $W = H = I = 1$ produce a tessellation in which there are two lines intersecting the sampling window ($L_t = 2$) that form four polygons ($P_t = 4$). Then, by dividing L_t and P_t by the sampling window

[6]The interval $[0, \infty)$ corresponds to the random distance of the lines to the origin, and the interval $[0, 2\pi)$ corresponds to the random angle of the point on the line that is closest to the origin. In the case of the orientation distribution, the upper interval bound is π since rotating a line by an angle of π (i.e. 180 degrees) leads to the same orientation.

area (i.e. $W \times H$), we obtain two further outputs corresponding to the number of lines and polygons per unit area (L_u and P_u, respectively). Since $W = H = 1$ in Figure 2, it follows that $L_u = L_t = 2$ and $P_u = P_t = 4$.

5.1.2 Testing Activity. In this case study, we replicate the SMT approach followed by Guderlei et al. in their seminal SMT paper [42] to explore whether the CTF can achieve comparable results to traditional SMT approaches. Here we investigate whether the CTF can do so without the need for a large number of model executions (as is usually the case with SMT) and with a practically feasible amount of input from the tester.

As in the original paper, we expect the following two metamorphic relations to hold for the PLT model:

(1) Doubling I should cause P_u to increase by a factor of 4.
(2) P_u should be independent of W and H.

5.1.3 Data Generation. We generated two sets of execution data. First, to obtain a "gold standard", we replicate the SMT approach followed in the original study [42]. Specifically, we sampled 50 input configurations, with the bounds for width and height incremented together over the interval $\{n \in \mathbb{N} | 1 \leq n \leq 10\}$ (i.e. $W = H = 1, W = H = 2, \ldots, W = H = 10$), such that the sampling window is always square, and the control and treatment values for intensity are powers of 2 up to 16. We then executed each configuration 100 times to account for non-determinism, resulting in 5000 model runs.

Second, to explore how the CTF enables us to re-use past execution data to infer the outcome of metamorphic test cases, we simulated an observational data set comprising 1000 executions of the PLT model. To produce this data set, we generated 1000 random input configurations using Latin hypercube sampling [30, 67] over the distributions $W, H \sim \mathbf{U}(0, 10)$ and $I \sim \mathbf{U}(0, 16)$. This sampling method provides even coverage of the input space and thus reduces our dependence on a statistical model to fill gaps in the data.

5.1.4 Causal Testing. To begin causal testing, we specify our modelling scenario and causal DAG. In line with the data generation process, our modelling scenario for this case study constrains the window to be a square with a maximum width (and height) of 10 and places an upper limit of 16 on the intensity parameter:

$$\{0 < W \leq 10, 0 < I \leq 16, W = H\}$$

We then construct the causal DAG shown in Figure 3 to model the following assumptions. First, we add the causes of L_t and P_t based on the theoretical approximations $L_t \approx 2i(w+h)$ and $P_t \approx \pi i^2 wh$ [22]. We do not, however, include a direct edge from I to P_t as the intensity (I) affects the number of polygons (P_t) indirectly through the number of intersecting lines (L_t). We then add the edge $L_t \rightarrow P_t$ since the number of polygons (P_t) is determined by the intersection of lines (L_t). Finally, we add edges from W and H to L_u and P_u since these quantities depend on the window area.

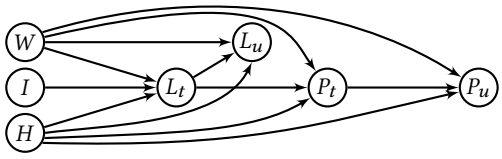

Fig. 3. A causal DAG for the PLT model.

Having created our causal specification, we now perform a series of causal tests to investigate the two metamorphic relations mentioned above: (1) whether doubling I causes P_u to increase by a factor of 4, and (2) whether the sample window size has a causal effect on P_u.

Effect of I on P_u. First, we test whether doubling I causes P_u to increase by a factor of 4 for $I \in \{1, \ldots, 16\}$ and $W, H \in \{1, \ldots, 10\}$. Since we are interested in the multiplicative effect of I on P_u, we use the *risk ratio* (RR, see Section 2), which quantifies the factor by which the intervention (doubling I) causes the outcome change:

$$\text{RR} = \frac{\mathbb{E}[P_u \mid I = i_t]}{\mathbb{E}[P_u \mid I = i_c]} \tag{1}$$

To estimate the RR using the CTF and observational data, we need to consider whether there is confounding bias in the data and design a regression model accordingly. To achieve this, we perform identification on the causal DAG shown in Figure 3, revealing that there is no confounding over the effect of I on P_u in this scenario. Therefore, we do not need to include additional terms for confounders in our regression model. However, because we expect P_u to vary quadratically with I, we opt to include the term I^2. This assumption is informed by domain expertise [42] but can be validated by varying I and observing changes to P_u. This process yields a regression model of the following form:

$$P_u \sim x_1 I + x_2 I^2 \tag{2}$$

We then apply the regression model to our observational data to obtain a causal estimate of the RR (Equation (1)).

Effect of W on P_u. Second, we test whether the sample window size W has a causal effect on P_u. Since we are only interested in whether there is *some* effect, we use the *average treatment effect* (ATE, see Section 2), which quantifies the additive change in outcome caused by the intervention (increasing W):

$$\text{ATE} = \mathbb{E}[P_u \mid W = w_t] - \mathbb{E}[P_u \mid W = w_c] \tag{3}$$

Ordinarily, to investigate whether W affects P_u using SMT, we would need to execute a fresh, customised set of test cases, this time fixing the value of I and varying W. In the CTF, however, we can infer this effect from the *same* 1000 model runs (i.e. re-using data from *previous* test executions to predict *new* test outcomes).

To achieve this, we start by performing identification on the causal DAG (Figure 3) for the effect of W on P_u, once again revealing the absence of confounding. We then modify the regression model shown in Equation (3) to include terms for W and W^{-1}, reflecting the hypotheses that W *does* affect P_u and that they share an inverse relationship (this can be validated by varying W and observing P_u). Although I is not a confounder here, we retain the I and I^2 terms to increase the accuracy of the model. The DAG in Figure 3 allows us to show that this does not bias our predictions. This process yields the following regression model:

$$P_u \sim x_1 W + x_2 W^{-1} + x_2 I + x_2 I^2 \tag{4}$$

We then apply this model to the *original* data to obtain a causal estimate for the ATE (Equation (3)). The effect of H could be investigated similarly, but we omit this due to space constraints.

5.1.5 Results. Table 1 shows the results for our investigation into the effect of I on P_u using Equation (2). The first 10 rows show the RRs obtained via the conventional SMT approach for various values of W and H, and the final row shows the RRs estimated using the CTF and observational data. The discrepancy between the regression estimations

Table 1. RR of doubling I under different values of W and H. The bottom row gives the value estimated using regression. Bold values round to 3, violating the expected behaviour.

W	H	$\frac{\mathbb{E}[P_u\|I=2]}{\mathbb{E}[P_u\|I=1]}$	$\frac{\mathbb{E}[P_u\|I=4]}{\mathbb{E}[P_u\|I=2]}$	$\frac{\mathbb{E}[P_u\|I=8]}{\mathbb{E}[P_u\|I=4]}$	$\frac{\mathbb{E}[P_u\|I=16]}{\mathbb{E}[P_u\|I=8]}$
1	1	**2.5888**	**3.4461**	3.6178	3.6187
2	2	**3.0359**	3.5410	3.6003	3.7264
3	3	3.5025	3.5945	4.0191	3.6545
4	4	**3.1138**	3.5285	4.1562	3.7290
5	5	3.6686	3.7686	3.9408	3.8751
6	6	3.6933	3.6988	3.9219	3.9707
7	7	3.7127	3.6271	3.9862	3.9370
8	8	**3.4957**	3.8300	3.8861	4.0110
9	9	3.5633	4.0009	3.9342	3.9338
10	10	3.8275	3.7525	4.0128	4.0181
CTF Estimate		**2.8280**	**3.1711**	**3.4772**	3.6993

and the SMT results are likely due to Equation (2) not including W and H terms, which the SMT results explicitly control for. However, this does not represent a biased result as Figure 3 shows there is no confounding.

These results show that both approaches identify an inconsistency between the metamorphic relations and implementation from the original study [42]: for lower values of W, H, and I, doubling I causes P_u to increase by a factor that is closer to three than four, meaning our metamorphic relation is not satisfied. This is a particularly interesting result since P_u should be independent of W and H.

Furthermore, these results show that the CTF was able to identify the same discrepancy as conventional SMT, but using a fifth of the data. This highlights the potential of CI-led approaches to offer economical alternatives to testing techniques that depend on repeated potentially costly executions of the SUT.

Table 2 shows the results of our investigation into the effect of W on P_u using Equation (4) and the same random 1000 data points as for the last row of Table 1. Here, each row shows how P_u changes when W is increased from W_c to W_t with the intensity fixed to $I = 1$. According to the original study [42], changes to W should *not cause* a change to P_u. Our results show that this property holds for all but the first row because these rows have confidence intervals that contain zero, meaning there is no statistically significant causal effect. However, the 95% confidence intervals for the first row of Table 2 show that, when W is increased from $W = 1$ to $W = 2$, there is a statistically significant causal effect on P_u of -7.3786. Although they are wide, indicating that the causal effect is variable, the fact that they do not contain zero indicates that the effect is statistically significant.

Table 2. Estimated ATE of increasing W from W_c to W_t on P_u with $I = 1$ in the PLT model with 95% confidence intervals.

W_c	W_t	ATE	95% CIs
1	2	-7.3786	[-13.9182, -0.8390]
2	3	-2.7097	[-9.8029, 4.3836]
3	4	-1.5424	[-11.1209, 8.0361]
4	5	-1.0755	[-13.7084, 11.5574]
5	6	-0.8421	[-16.7413, 15.0572]
6	7	-0.7087	[-19.9729, 18.5556]
7	8	-0.6253	[-23.3084, 22.0578]
8	9	-0.5697	[-26.7043, 25.5649]
9	10	-0.5308	[-30.1383, 29.0767]

This conflicting result indicates a problem with either the program or the metamorphic property. In this case, we believe that the problem stems from basic geometry: lines are less likely to intersect a smaller sample window. As

the sample window becomes larger, there is more area to average over so P_u becomes more reliable. Therefore, the metamorphic relations should ideally specify a minimum window size to which they apply.

Overall, this case study has provided evidence related to all three research questions.

RQ1. In this case study, we demonstrated the CTF's ability to reproduce published SMT results from [42] using a sample of randomly generated test data. First, we estimated the risk ratio of doubling I. Table 1 shows that our regression model was able to give sufficiently accurate results to discover an inconsistency that was also revealed by SMT, even though it did not explicitly control for W and H like SMT did. In the second part of the case study, we investigated this inconsistency further, and estimated the ATE of increasing W on P_u. While we expected this to be zero, Table 2 shows that there is actually a statistically significant negative relationship when we increase W from 1 to 2.

RQ2. This case study demonstrated the CTF's ability to find the same bugs as SMT using only a fraction of the data. Furthermore, the second part of this case study involved using the *same data* as for the first part to test a *different relationship* after having discovered a potential bug in the system. By contrast, the traditional SMT approach would need to perform additional controlled runs of the system, which vary W while holding I constant, to test this new property.

RQ3. The DAG for this case study, shown in Figure 3, required minimal effort to construct. There are no internal variables here, and the relationship between the inputs and outputs is well documented in [42]. The main drawback is the requirement for the domain expert to have an approximate idea of the "shape" of the relationships between different variables, for example that P_u varies with I^2 rather than just I, in order to obtain accurate estimates.

This case study has shown that not only can we conduct SMT using the CTF, but we can do so *using previous execution data* and *less data* than a traditional SMT method. Furthermore, we demonstrated how this approach allowed us to refine our metamorphic relations and find faults *without running the SUT additional times.*

5.2 Cardiac Action Potential Model

In this case study, we use the CTF to conduct sensitivity analysis on the Luo-Rudy 1991 ventricular cardiac action potential model [62] (LR91) in a straightforward and efficient way. Sensitivity analysis is commonly used to validate and verify scientific models, with a specific focus on identifying which inputs have the greatest impact on model outputs [57, 91]. Here, we take a CI-led approach and measure the ATE of several input parameters on one output, quantifying the extent to which this output is affected by changes to the inputs. As test oracles, we construct a series of metamorphic relations that capture the expected magnitude and direction of each ATE.

Throughout this case study, we follow part of an existing study [19] that conducts uncertainty and sensitivity analysis on LR91 using a Gaussian Process Emulator (GPE) [87] trained on runs of the model. This work provides an invaluable source of domain expertise that precisely quantifies several cause-effect relationships between the inputs and outputs of LR91 that we use as the basis for constructing our metamorphic relations. However, in contrast to the data-driven approach employed in the original study, we employ causal knowledge and domain expertise to justify and hand-craft a simple regression model that reaches the same conclusions. The code for reproducing this case study can be found in our open source repository[7].

[7] https://github.com/CITCOM-project/CausalTestingFramework/tree/683e6c55/examples/lr91

5.2.1 Subject System. The Luo-Rudy 1991 ventricular cardiac action potential model [62] (LR91) is a mathematical model comprising a system of differential equations that describe the rapid rise and fall in the voltage across the membrane of a mammalian ventricular cell. This characteristic rise and fall in voltage is referred to as an *action potential*. The behaviour of this model is controlled by 24 constants, 8 rate variables, 8 state variables, and 25 algebraic variables.

We selected LR91 as a case study as it follows a different modelling paradigm to our other subject systems and has supported extensive and important research into cardiovascular physiology. Furthermore, amongst its vast and largely uncertain input space, LR91 has several well-characterised input-output relationships suitable for causal analysis.

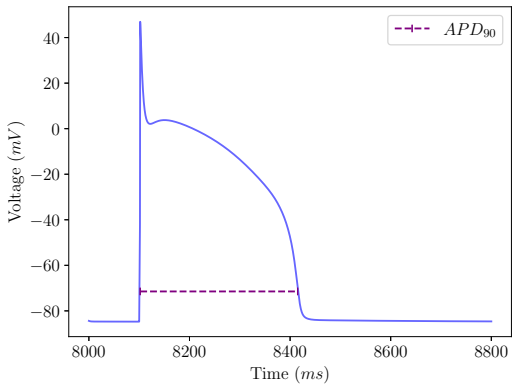

Fig. 4. An example action potential produced by the Luo-Rudy 1991 model, simulating the rise and fall of voltage across a mammalian ventricular cell, and the output of interest: APD_{90}.

An example action potential produced by LR91 is shown in Figure 4, demonstrating the rapid rise (known as depolarisation) and corresponding fall (repolarisation) of the voltage over time. In this case study, we quantify the effect of six conductance-related input parameters on one attribute of the action potential: *action potential duration to 90% of repolarisation* (APD_{90}). That is, the amount of time taken for the action potential to repolarise by 90%. This output concerns the falling phase of the action potential in which the cell returns to its resting voltage [41] and is shown in Figure 4.

5.2.2 Testing Activity. In this case study, we replicate part of an existing study [19] that conducts a sensitivity analysis on LR91 using a Gaussian Process Emulator (GPE) [87]. In short, the approach in [19] trained a GPE on 200 runs of LR91, with input configurations sampled via Latin Hyper Cube Sampling [98] from a series of normalised uniform design distributions to ensure even coverage of the input space. The GPE was then used to calculate the expectation of a given output, conditional on an input of interest, to quantify the effect of varying each of the six inputs on the eight output parameters, over the range of the design distribution.

From a CI perspective, we can obtain similar information by computing the ATE of each input on each output over the range of the design distribution. Specifically, we can set our control value to the mean value of the design distribution and uniformly increment our treatment value from the minimum to the maximum value of the design distribution. This yields a series of ATEs that quantify the expected change in output caused by changing the input parameters by specific amounts above and below their mean, revealing the magnitude of each input's effect on the outputs.

Due to space limitations, we limit our analysis to the effect of the six inputs on one output, APD_{90}. We have selected this output because the original paper uses it to illustrate the approach. Based on the results reported in [19], we expect the following metamorphic properties to hold:

(1) Increasing the parameters G_K, G_b, and G_{K1} should cause APD_{90} to decrease.
(2) Increasing the parameter G_{si} should cause APD_{90} to increase.
(3) Increasing the parameters G_{Na} and G_{Kp} should have no significant effect on APD_{90}.
(4) The following monotonic relationship should hold over the (absolute) magnitude of the parameters' effects:

$$|APD_{90}^{G_{si}}| > |APD_{90}^{G_K}| > |APD_{90}^{G_b}| > |APD_{90}^{G_{K1}}|$$

5.2.3 Data Generation. To gather data from LR91, we followed the same approach as [19], where the 200 input configurations were sampled from the design distributions using Latin Hyper Cube sampling and then normalised. We then executed each of these input configurations on an auto-generated Python implementation of LR91 from the cellML modelling library [18]. We extended this implementation to enable us to sample the input values via Latin Hyper Cube sampling and automatically extract the outputs[8].

5.2.4 Causal Testing. To approach sensitivity analysis as a CI problem, we first specify our modelling scenario and causal DAG. For this set of tests, the modelling scenario constrains each input to the range of its uniform design distribution (as specified in the original paper [19]):

$$\{17.250 \leq G_{Na} \leq 28.750, 0.0675 \leq G_{si} \leq 0.1125, 0.2115 \leq G_K \leq 0.3525,$$
$$0.4535 \leq G_{K1} \leq 0.7559, 0.0137 \leq G_{Kp} \leq 0.0229, 0.0294 \leq G_b \leq 0.0490\}$$

As in the original study, these input values were then normalised to the range [0, 1].

We then specify the expected cause-effect relationships (and absence thereof) as the causal DAG shown in Figure 5. While not essential, we include the isolated nodes G_{Na} and G_{Kp} in our DAG to make our expectation for the absence of a causal effect explicitly clear. For each relationship, we then create a suite of causal test cases covering a series of interventions that incrementally increase/decrease the value of the inputs over the range of the design distribution. For each input, this is achieved by setting the control value to 0.5 (the mean) and uniformly sampling 10 treatment values over the range [0, 1]. This produces a total of 10 test cases per input that vary its value from 0.5 to each of the treatment values: [0, 0.1, 0.2, ... 1.0]. Using the CTF, we then perform identification and estimation. Here, the cause-effect relationships are straightforward and there is no confounding to adjust for, enabling us to fit a regression model $APD_{90} \sim x_0 + x_1 G_z$ for each input $z \in \{si, K, Na, Kp, K1, b\}$. Using these models, we then predict the ATE and 95% confidence intervals for each test.

5.2.5 Results. The results, as summarised in Figure 6, show that all expected metamorphic relationships pass with statistical significance (95% confidence intervals do not contain 0) and are visually similar to Figure 5 in the original study [19]. Specifically, the first metamorphic relation holds as G_K, G_{K1}, G_b have negative effects, the second metamorphic relationship holds because G_{si} has a positive effect, and the third metamorphic relation holds as G_{Na} and G_{Kp} have no significant effect. Furthermore, the fourth metamorphic relation holds as the gradients corresponding to these effects reveal that the effect sizes follow the expected monotonic relationship: $|APD_{90}^{G_{si}}| > |APD_{90}^{G_K}| > |APD_{90}^{G_b}| > |APD_{90}^{G_{K1}}|$.

[8]Our LR91 model is available at: https://github.com/AndrewC19/LR91/tree/769e7ff

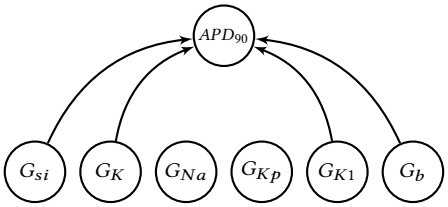

Fig. 5. LR91 modelling scenario's Causal DAG, where the sensitivity of APD_{90} to each conductance input is computed as the causal effect (ATE).

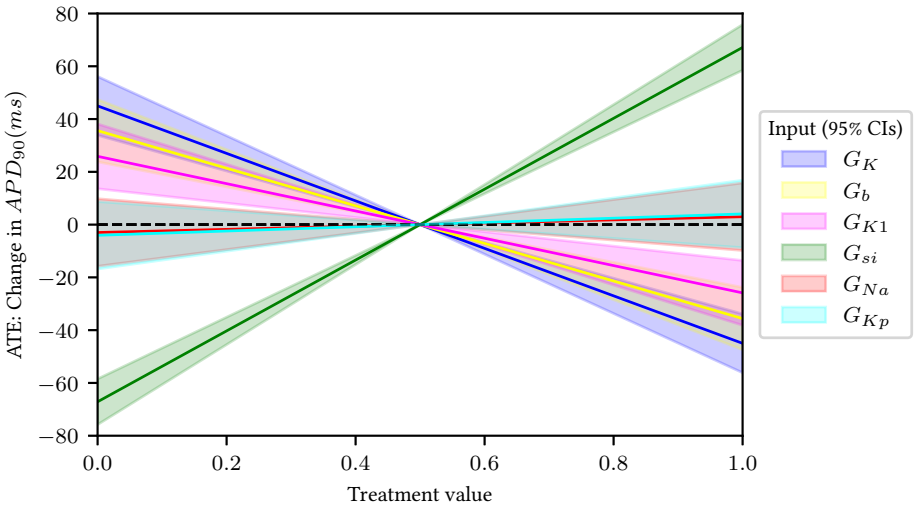

Fig. 6. Sensitivity of APD_{90} in response to changes to the mean value of input parameters in LR91.

This case study has provided insights into **RQ1** and **RQ3**. As a result of following the data generation approach of the original paper, however, this case study did not afford us the opportunity to evaluate the efficiency of the CTF.

RQ1 *Accuracy.* In this case study, we used the CTF to conduct a sensitivity analysis on the LR91 model, achieving visually similar results to an existing approach that employed a GPE [19]. However, we achieved this using a significantly simpler statistical model whose design was informed by causal reasoning as opposed to associations within the data. This contrast between a model-based and black-box approach to reasoning about system behaviour raises an interesting discussion around *explainability* that we return to in Section 6.

RQ3 *Practicality.* In this case study, the process of specifying the causal DAG was straightforward and required minimal domain expertise that were easily gleaned from the original study [19]. Since the resulting DAG contained no confounding (Figure 5), the regression model for each causal test simply regressed the input-under-test against against APD_{90}. By contrast, Gaussian Processes (as used in the original study) have several practical limitations, including the need to specify an appropriate kernel function for the problem at hand [70], and a complexity of $O(n^3)$ that hinders the feasibility of the approach when dealing with large amounts of data [88].

Overall, in this case study, we have shown that the CTF reaches the same conclusions as the original study. However, the CTF achieves this by using a simpler, more practical statistical model guided by causality instead of associations within the data.

5.3 Covasim: Experimental Casual Testing

In this case study, we demonstrate the ability of the CTF to conduct statistical metamorphic testing (SMT) of Covasim [56] using the experimental mode of the CTF (Section 4.3). That is, isolating the causal effect of interest via strategic executions of the SUT, rather than applying graphical CI to observational data. Our aim here is to provide evidence to support our claim that metamorphic testing is a fundamentally causal activity that can be framed and solved as a problem of CI. The code for this case study can be found in our open source repository[9].

5.3.1 Subject System. Covasim is the epidemiological agent-based model that was introduced as a motivating example in Section 2. As a brief reminder, it is a complex, real-world scientific model that is primarily used to simulate detailed COVID-19 scenarios in order to evaluate the impact of various interventions, such as vaccination and contact tracing [56], in specific demographics. These scenarios are configured via 64 input parameters and described by 56 time-series outputs. It has been used to inform a number of important policy decisions across a range of countries, including the UK, US, and Australia [55, 76, 77, 100],

We cover two testing scenarios using Covasim. In this section, we elaborate upon our example from Section 2 and use the experimental mode of the CTF to test the effect of prioritising the vaccination of elderly people on several vaccine-related outcomes, revealing an interesting bug in the process. Then, in Section 5.4 we test the effect of increasing the β parameter (transmissibility) on cumulative infections using execution data from other tests (i.e. data that has not been customised to explore this specific effect).

5.3.2 Testing Activity. Revisiting our example from Section 3, our aim is to determine the effect of prioritising vaccination for the elderly on the following outputs: cumulative infections, number of doses given, maximum number of doses per agent, and number of agents vaccinated.

Our expectation here is that prioritising the elderly should lead to an increase in infections. This is because we are less likely to vaccinate agents in the model with a greater propensity for spreading the virus (e.g. younger individuals who attend a school or workplace). We also expect the number of vaccines and doses administered to decrease as there are fewer elderly agents in the model. In contrast, the maximum number of doses should not change, as the vaccine is set to be administered at most two times per agent.

5.3.3 Data Generation. We executed the model under two input configurations 30 times each using an experimental data collector (see Section 4.3) for every test. For both input configurations, we used the default Covasim parameters, but fixed the simulation length to 50 days, initial infected agents to 1000, population size to 50,000, and made the default Pfizer vaccine available from day seven. However, for the second configuration, we also sub-targeted (prioritised) vaccination to the elderly using the `vaccinate_by_age` method from the Covasim vaccination tutorial[10].

5.3.4 Causal Testing. Although we provide a causal DAG (Example 3.3) as an illustrative example for this scenario in Section 3, it is not necessary to perform identification since, under the experimental mode of operation (Section 4.3), we explicitly control for potential biases. Consequently, there is no confounding to adjust for in the resulting data, enabling

[9] https://github.com/CITCOM-project/CausalTestingFramework/tree/683e6c55/examples/covasim_/vaccinating_elderly
[10] https://github.com/InstituteforDiseaseModeling/covasim/blob/master/examples/t05_vaccine_subtargeting.py

us to calculate the ATE directly by contrasting the average cumulative infections produced by the control (vaccinate everyone) and treatment executions (prioritise the elderly).

5.3.5 Results. As expected, prioritising the elderly causes the cumulative infections to increase (ATE: 2399.7, 95% CIs: [2323.7, 2475.8]) and causes no change to the maximum doses (ATE: 8.9×10^{-16}, 95% CIs: [3.7×10^{-17}, 4.1×10^{-16}]).

However, when we examine the number of doses given (which, as stated in Example 3.3, we would expect to remain fixed), the tests in fact show that the SUT erroneously causes the number of doses administered and the number of people vaccinated to increase sharply by 481351 (95% CIs: [480550, 482152]) and 483506 (95% CIs: [482646, 484367]), respectively. This is an obvious and potentially problematic bug, as it reveals that more agents have been vaccinated than there are agents in the simulation (by a factor of 9.7).

We raised an issue[11] on Covasim's GitHub repository to report this bug in September 2021 and the Covasim developers replied in November confirming that the bug had been fixed for version 3.1. Although the developers did not explain the cause of the bug nor how it was fixed, the change log for version 3.1 stated the following: *Rescaling now does not reset vaccination status; previously, dynamic rescaling erased it.*

This testing scenario has provided insights related to **RQ2** and **RQ3**. Due to employing the experimental mode of the CTF (Section 4.3), we have not inferred test outcomes from observational data and therefore this case study does not offer any insights into the accuracy associated with the observational approach.

RQ2 (Efficiency). We used the experimental mode of the CTF to quantify the effect of introducing a vaccination policy on a number of variables, essentially conducting SMT in the conventional way. We repeated both the source and follow-up test cases for each metamorphic relation 30 times for each test (of which there were four), requiring a total of $30 \times 2 \times 4 = 240$ executions of Covasim. We show how, under the experimental mode of operation, the CTF can conduct SMT in the conventional way and demonstrate that, in situations where observational data is unavailable, the CTF can match the efficiency of conventional SMT.

RQ3 (Effort). The amount of human effort required to apply the CTF was low. We did not need to provide a DAG and we did not need to specify a regression model. Instead, the main expenditure of human effort in this case study lies in the process of implementing the test harness for experimental data collection; a step that is required for most model-based testing techniques.

Overall, this case study has demonstrated how the CTF can also be employed under the experimental mode of operation to essentially conduct a conventional SMT approach. This revealed a problematic bug related to vaccination, highlighting the importance of applying metamorphic testing in the scientific context.

5.4 Covasim: Observational Causal Testing

We now consider the effect of increasing transmissibility (β) on cumulative infections, but this time applying the CTF to simulated confounded observational data. Here we compare the outcomes inferred by the CTF to the same outcomes achieved using a conventional SMT approach. Our goal here is to understand whether the CTF can operate accurately and efficiently within the challenging context presented by Covasim.

This case study presents a significant testing challenge. There are 156 distinct locations that can be simulated in Covasim that will lead to differing rates of transmission. This is because different locations are modelled with different

[11] https://github.com/InstituteforDiseaseModeling/covasim/issues/370

age distributions and household contact patterns, leading to differences in key attributes of the population, such as susceptibility, that also affect infection dynamics.

Furthermore, Covasim is non-deterministic. Each metamorphic test requires multiple repeats of the source and follow-up tests, making conventional SMT extremely costly in this context. For example, if we repeat both the source and follow-up test cases 30 times for each location, we would need to run $30 \times 2 \times 156 = 9360$ simulations. Although we do not provide precise timing measurements, on a moderate specification machine[12] each of these runs takes between 1 and 2 minutes to complete, requiring between 156 and 312 hours to run all simulations (without parallelisation). The code for this case study can be found in our open source repository[13].

5.4.1 Data Generation. When reasoning about transmissibility and the spread of COVID-19 using Covasim, there are several parameters that can affect the output. These include the variant of the virus and population characteristics such as age and household size, with older populations being more susceptible to infection and higher household contacts leading to quicker viral spread. These population characteristics cannot be specified directly, but can be indirectly altered by selecting a geographical location.

For this case study, we generate two sets of data. First, we directly apply a conventional SMT approach to Covasim in which we execute the model 30 times with $\beta = 0.016$ and $\beta = 0.02672$ for each location, before averaging and contrasting their respective cumulative infections. We select these values of β as they correspond to the β values for the Beta and Alpha variants of COVID-19 available in Covasim.

Second, we simulate (uncontrolled) observational data. To achieve this, we assign a different dominant variant (Alpha, Beta, Delta, Gamma) to each location at random, each of which has its own specific β value ($\beta_\alpha = 0.02672, \beta_\beta = 0.016, \beta_\delta = 0.0352, \beta_\gamma = 0.0328$). For each location, we then create a normal distribution centred around the location-specific β value and a standard deviation of 0.002. We select this standard deviation to give some variance in the exact value of β used for each run of the location, without introducing too much overlap with other variants. We then run 30 simulations for each location, sampling a fresh β value from its distribution on each run. For all simulations, we use a population size of 1 million individuals, 1000 initially infectious individuals, and a duration of 200 days. This results in a data set comprising 4680 simulations (30 per location).

5.4.2 Causal Testing. To begin causal testing, we form our causal specification by specifying a modelling scenario and the causal DAG shown in Figure 7. Our modelling scenario uses the default Covasim parameters apart from β (the input-under-study) and the location. We also fixed the duration, population size, and initial infected agents as follows:

$$\{\texttt{days} = 200,\ \texttt{pop\_size} = 1000000,\ \texttt{pop\_infected} = 1000\}$$

Next, we consider the adjustment sets implied by the causal DAG in Figure 7. While there are many possible adjustment sets for this causal DAG, there are three notable choices to discuss.

First, we could use the smallest adjustment set $\{L\}$. This has the advantage of conditioning on the least variables, but restricts estimation to using location-specific data only (i.e. not borrowing data from *similar* locations). Second, we could use $\{A, C_H\}$. This would enable us to additionally borrow information from locations that have similar average ages and household contacts. From an information theoretic standpoint, however, this is not a sensible choice as the average age is not a good measure for the shape of the age distribution (two populations with a similar average age may have vastly different age distributions). To this end, we can consider a third adjustment set $\{S, C_S, C_W, C_H\}$. Here,

[12] MacBook Pro, Core i7, 16GB 2133 MHz LPDDR3 RAM
[13] https://github.com/AndrewC19/covasim_case_study/tree/65bc40a

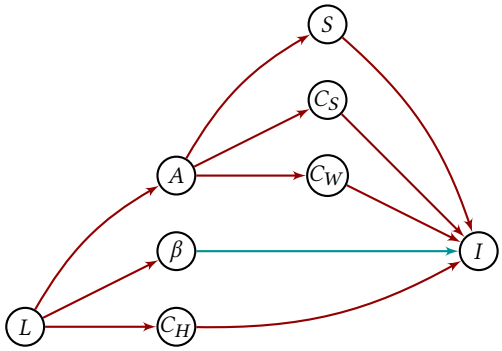

Fig. 7. A causal DAG for the Covasim modelling scenario where the causal effect of β on I is confounded. Here, L denotes the location; A denotes the average age of the population; β denotes the transmissibility of the virus; C_H, C_S, C_W denote household, school, and workplace contacts; S denotes average susceptibility of the population; and I denotes the total cumulative infections.

we replace A with the variables related to age that directly affect cumulative infections: the number of school and workplace contacts (assignment to these environments is determined by age) and susceptibility (which varies with age).

For this case study, we select this third adjustment set on the basis that it most accurately captures the key causal measures while allowing us to borrow data from other locations that are similar with respect to these attributes. This yields the following closed-form statistical expression that is capable of directly estimating the causal effect (CATE) of interest:

$$CATE = \mathbb{E}[I \mid \beta = 0.02672, S, C_S, C_W, C_H] - \mathbb{E}[I \mid \beta = 0.016, S, C_S, C_W, C_H]$$

Then, to estimate the value of this estimand, we implement a regression model of the following form, where Z is our adjustment set $\{S, C_S, C_W, C_H\}$ and each variable in this adjustment set has three coefficients: x_1^z, x_2^z, x_3^z:

$$I \sim x_0 + x_1 ln(\beta) + \sum_{z \in Z} x_1^z ln(z) + x_2^z ln(z)^2 + x_3^z ln(z) ln(\beta)$$

This regression model encodes three key assumptions. First, due to the exponential nature of viral infection, we apply a log transformation to the variables on the right-hand-side of the equation [12, 99]. Second, we add a quadratic term for each of our adjusted variables. This captures the possibility of curvilinear relationships between I and the parameters. Third, we include an interaction term between β and each of our adjusted parameters. This captures our expectation that the effect of β on cumulative infections is moderated by the number of contacts and susceptibility of the population, and enables the model to make location-specific estimates i.e. conditional ATEs (CATEs; see Section 2.4)[14].

At this point, we have specified a causally-valid statistical model that is capable of directly estimating the causal effect of β on cumulative infections for each location separately. We can therefore compute the average values for the variables S, C_S, C_W, and C_H for each location using our observational data, and substitute these into the model alongside the values $\beta = ln(0.016)$ and $\beta = ln(0.2672)$[15]. By contrasting the respective estimates for I, we obtain an estimate of the causal effect for each location in Covasim.

5.4.3 Results. Figure 8 summarises the results of applying the CTF to Covasim to predict the effect of increasing transmissibility (β) on cumulative infections across all locations. These results show three values for each location:

[14] We formed these assumptions by varying each parameter in isolation and observing the change in cumulative infections. An epidemiologist, however, may know more precise characterisations of these relationships a priori.

[15] We take logarithms of the treatment and control values here to maintain the interpretability of our estimate.

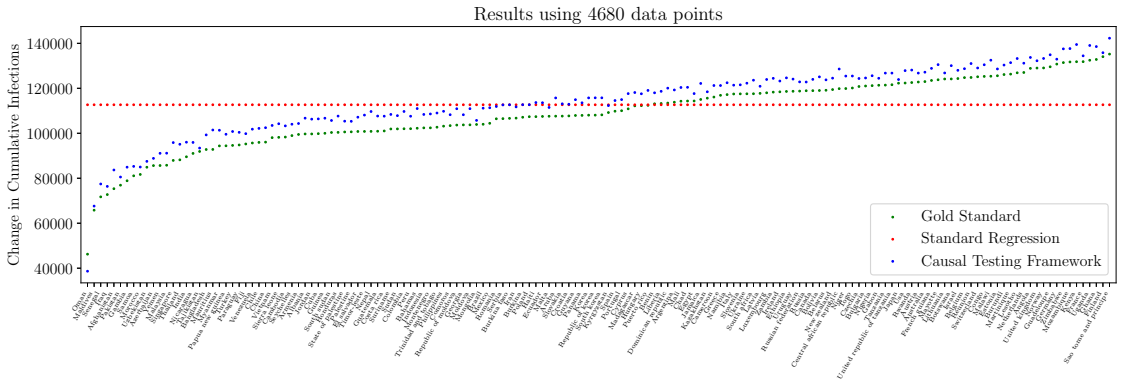

Fig. 8. A comparison of the metamorphic test outcomes predicted by the CTF and a naive regression model. The metamorphic test in question increases the value of β from 0.016 to 0.02672.

(i) the gold standard achieved by applying an SMT approach, (ii) a naive estimate with the simple regression model $I \sim x_0 + x_1 ln(\beta) + x_2 ln(\beta)^2$ (i.e. without employing causal knowledge), and (iii) a causal estimate achieved using the CTF and the approach outlined in this section.

By comparing the CTF results to the gold standard shown in Figure 8, we can see that the CTF is able to estimate the effect of increasing β from 0.016 to 0.02672 for each location with reasonable accuracy. Specifically, across the location specific estimates, the CTF has a root mean square percentage error (*RMSPE*) of 0.055. This outperforms the naive regression model which provides a uniform prediction that is moderately accurate for 'average' locations, but extremely inaccurate for more 'extreme' locations ($RMSPE = 0.2$).

While these results suggest that the CTF generally overestimates the effect by an average of roughly 5.5% cumulative infections, the overall ordering of the predicted effect sizes is generally consistent with that of the gold standard. We tested this preservation of ordering by calculating the Kendall rank correlation between the (ascending) ordering of the CTF results and the gold standard, returning a value of 0.944 ($p < 0.005$).

By contrast, Figure 9 shows the results achieved using the smallest adjustment set, L, and regression model $I \sim x_0 + x_1 ln(\beta) + x_2 ln(\beta)^2 + ln(x_3)\beta L$. This approach makes location-specific estimates using only the data available for the location in question and is essentially an attempt to apply SMT to incomplete, confounded data. Because each location-specific stratum contains only 30 executions that cover a narrow range of β values, the regression model has to make inaccurate extrapolations, leading to significant over- and under-estimates of the true effect ($RMSPE = 0.515$) and poor rank preservation, as indicated by a Kendall's rank correlation of 0.228 ($p < 0.005$). This stark contrast in performance highlights the value of employing causal knowledge and domain expertise to use data more efficiently.

While Figure 8 demonstrates the accuracy with which the CTF can predict SMT outcomes from confounded observational data, these results used the full data set comprising 4680 simulations. Although this is half of the 9360 executions that would typically be required for a conventional SMT approach, this is still a significant amount of data that may not be available in practice. To investigate how much is necessary in practice, we repeatedly applied the CTF to randomly sampled subsets of the data of decreasing size and calculated the RMSPE and Kendall's rank correlation. We repeated this process 30 times to obtain a distribution of outcomes and report 95% confidence intervals to demonstrate

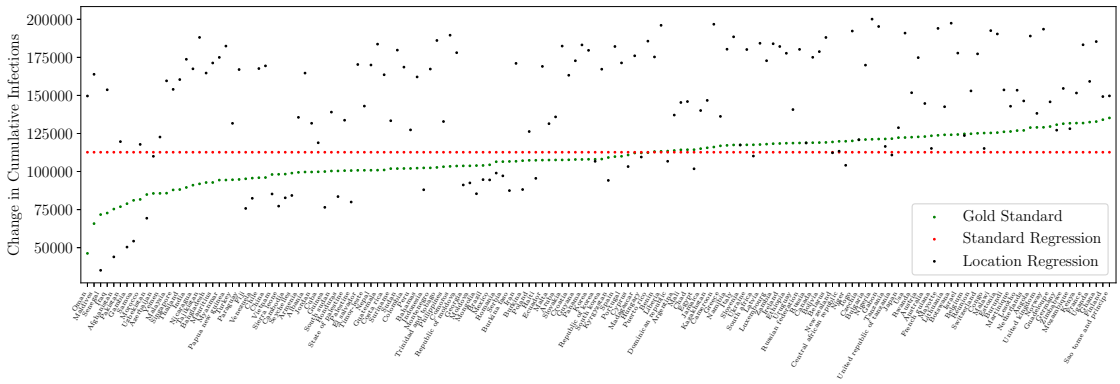

Fig. 9. A comparison of the metamorphic test outcomes predicted by a naive regression model and the same model with an interaction between location and β.

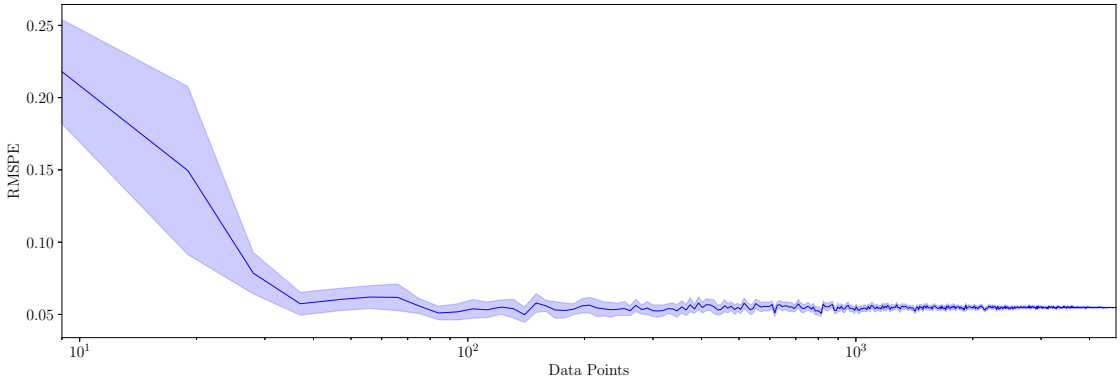

Fig. 10. Relationship between root mean square percentage error (RMSPE) of CTF predictions and amount of data used (log scale) with 95% confidence intervals.

the error. Figure 10 and Figure 11 show the results of these experiments. We use a logarithmic scale on the x-axis for these figures as the accuracy changes most significantly between 1 and 200 data points.

Figure 10 shows that the RMSPE is greatest with small amounts of data (tens of data points) and quickly reduces to a stable RMSPE of roughly 0.06 by around 200 data points. Similarly, Figure 11 shows that the Kendall's rank correlation is initially low (between 0.2 and 0.4) but rapidly increases to a stable value of around 0.9 when 100 to 200 data points are available. This plateau in absolute and comparative error reduction indicates that SMT outcomes can be accurately predicted using only small amounts of data and that larger amounts of data provide negligible gains in accuracy.

This testing scenario has provided evidence for all research questions.

RQ1 (Accuracy). Figure 8 shows the accuracy with which the CTF can infer a series of 156 SMT outcomes from confounded observational data *a posteriori*. Although the majority of estimates miss the true effect by around 5.5%, the ordering of the effect sizes is largely consistent with the gold standard. This finding suggests that, in this case study, the

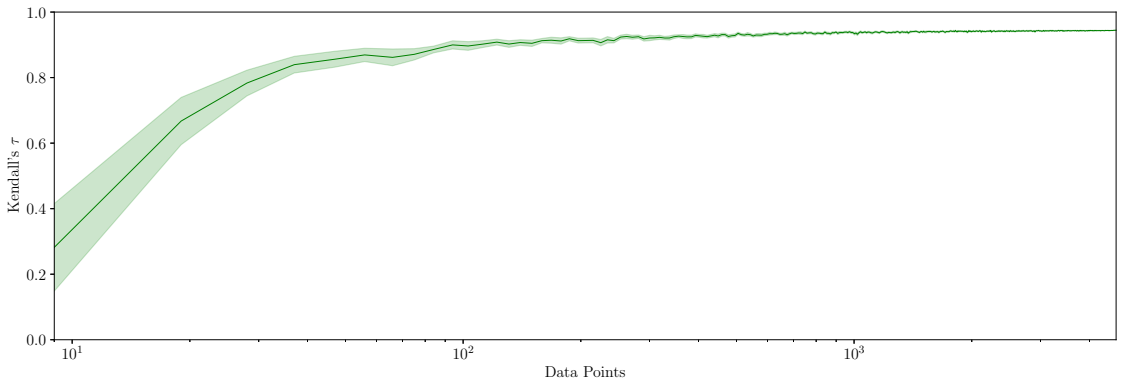

Fig. 11. Relationship between Kendall's rank correlation (τ) of CTF predictions and amount of data used (log scale) with 95% confidence intervals.

CTF is better suited to drawing comparative conclusions about the effect sizes, such as *"Oman is affected significantly less than Finland"* than absolute conclusions, such as *"Finland observes an increase in cumulative infections of 135829"*.

RQ2 (Efficiency). As shown in Figures 10 and 11, after 200 data points, there is negligible improvement to the absolute and comparative accuracy of the estimator. This suggests that, in this case study, the CTF is significantly more efficient than a conventional SMT approach which would require 9360 executions of the SUT (assuming the source and follow-up tests are repeated 30 times each), with each execution requiring roughly one to two minutes on a moderate specification machine, as noted in earlier in this case study.

RQ3 (Practicality). In this case study, we leveraged our limited domain expertise to specify a causal DAG and regression model that facilitates efficient and accurate inference of test outcomes. Most notably, to borrow data from similar locations, we leveraged our knowledge of viral transmission in Covasim to add terms to our regression model for the attributes that influence the effect of transmissibility on cumulative infections, such as contacts and susceptibility. We achieved this using a relatively small DAG containing only eight nodes and employing commonplace regression modelling techniques, such as quadratic, logarithmic, and interaction terms.

Overall the findings of this case study highlight the potential offered by a CI-led approach to SMT: whereas a conventional SMT approach would require thousands of carefully controlled executions to test 156 metamorphic relations, the CTF can accurately infer these outcomes from only 200 data points. Furthermore, the CTF enables a tester to infer these outcomes *a posteriori* from potentially confounded data instead of executing the SUT further times. This approach essentially relaxes the constraints ordinarily placed on data used for SMT, facilitating the re-use of existing data while maintaining the ability to draw *causal conclusions*.

6 DISCUSSION

In this section, we discuss the findings of our three research questions outlined in Section 5, pertaining to the *accuracy*, *efficiency*, and *practicality* of the proposed approach. We also discuss notable additional findings that fall outside the scope of our research questions, including a pair of bugs identified in the case studies.

6.1 RQ1 (Accuracy): Can we reproduce the results of a conventional MT/SMT approach by applying the CTF to observational data?

Throughout our case studies, we applied the CTF to a number of different subject systems from different domains to predict MT and SMT outcomes from observational data. That is, data that had not been collected specifically for the testing task in question.

In Section 5.1, for example, we were able to predict the outcome of two statistical metamorphic tests for a tessellation model with sufficient accuracy to reveal a faulty metamorphic relation. We then confirmed this using a conventional SMT approach. Similarly, in Section 5.2, we predicted several metamorphic test outcomes for a cardiac action potential model, reproducing the results of an existing study. In Section 5.4, we then showed how observational data could be re-used to predict multiple different statistical metamorphic test outcomes for an epidemiological model with high comparative accuracy.

> The CTF is able to accurately reproduce the results of both MT and SMT across a range of scientific modelling software.

This finding suggests that, by leveraging CI, the CTF can offer an alternative approach to SMT that does not rely on many potentially costly executions of the SUT. Instead, the CTF can be employed *retrospectively* to infer test outcomes from existing, potentially confounded test data, effectively relaxing the constraints ordinarily imposed on the data used for SMT. In this way, the CTF makes it possible to apply SMT where conventional approaches are currently prohibitively expensive, thereby mitigating the problem of long execution times, as discussed in Section 2.2 and Kanewala and Bieman's survey [51].

While our case studies show that the CTF can infer SMT outcomes with good accuracy for a range of systems, there are more advanced estimation techniques that could be employed to further increase the accuracy. To illustrate this point, in **Appendix** we demonstrate how a more advanced form of regression modelling known as spline regression can more accurately capture the theoretical shape of the cause-effect relationship between β (transmissibility) and cumulative infections (originally discussed in Section 5.4). In future work we will compare the performance and usability of more advanced statistical models, such as spline regression [64] and causal forests [4].

6.2 RQ2 (Efficiency): In terms of the amount of data required, is the CTF more cost-effective than a conventional MT/SMT approach?

In Section 5.1 (PLT model) and Section 5.4 (Covasim), we used the CTF to conduct SMT using less data than would be required by a conventional SMT approach. In the case of PLT, we were able to reproduce the results of a conventional SMT approach using a fifth of the data, uncovering a failed metamorphic relation in the process. Similarly, in Section 5.4 we used the CTF to infer the outcomes of 156 distinct metamorphic relations, as shown in Figure 8, using roughly half the amount of data required by a conventional SMT approach. We then incrementally reduced the amount of data and repeated our analysis to understand how the accuracy of the approach varies with respect to the amount of data, finding that near-identical results could be achieved using only 200 data points.

Furthermore, although we have not obtained precise timing measurements, we note that the CTF takes roughly a minute to produce all 156 of the location-specific effect estimates shown in Figure 8 on a moderate specification machine. On the other hand, an individual run of Covasim with the settings used in this case study took between one and two minutes on the same machine, and 9360 executions would be required to test these 156 effects using

conventional SMT (with 30 repeats per source and follow-up test case). This would amount to between 156 and 312 hours without parallelisation.

> The CTF is capable of reproducing the results of SMT using significantly less time and data than is required by a conventional SMT approach.

These findings demonstrate the potential of the CTF to infer the outcomes of metamorphic test cases using significantly less time and data than is required by a conventional SMT approach. Therefore, in conjunction with our findings for **RQ1**, our answer to **RQ2** suggests that the CTF can offer an efficient alternative to conventional MT and SMT approaches that is more compatible with the notoriously demanding properties of scientific software, such as non-deterministic behaviour and long execution times, as described in Section 2.2.

An open question surrounding the efficiency of the CTF is how the quality and diversity of the available data affects also the accuracy and scope of inferences. To this end, an interesting avenue for future work would be to investigate how existing test generation and selection strategies can be combined with the CTF to generate and prioritise test cases that, once executed, produce execution data with the greatest inferential potential. In a similar vein, Bareinboim and Pearl [10] have proposed general-purpose methods to combine different data sources generated under different conditions to maximise what can be learned from the data. Future work could also investigate how these data fusion techniques can be leveraged in a software testing context to further the inferential power of available data sources.

6.3 RQ3 (Practicality): What practical effort is required from the tester to conduct MT/SMT using the CTF?

Across our case studies, we primarily drew the causal knowledge necessary to elicit the causal DAGs and regression models from existing studies in which the anticipated cause-effect relationships are well-defined. For example, in Section 5.2, we used the results of an existing study [19] to specify the causal DAG for the cardiac action potential model (see Figure 5). Similarly, in Section 5.1 (PLT), we based the shape of our regression models on theoretical results that were also used as the basis of statistical metamorphic relations in the seminal paper on SMT [42]. The main expenditure of human effort here was gathering the domain expertise for each system; converting these into causal DAGs was straightforward and required little time. It stands to reason that this would be less time-consuming for a scientific modeller (for example), who would already have a reasonably strong understanding of the underlying subject matter.

As with any model-based testing technique, time and effort are necessary to obtain knowledge and turn it into a domain model. In addition, this process often assumes familiarity with software-specific notions, such as how to characterise a state in a state machine [24], or what events should (or should not) be possible at any given point. Furthermore, the resulting models tend to contain implementation-specific details likely to be unfamiliar to most scientific software users [51]. By contrast, the CTF relies on an intuitive, domain-agnostic model (i.e. a causal DAG) that makes essential assumptions transparent and requires a basic understanding of regression modelling. This set of requirements poses a lower barrier to entry for a typical user of scientific software.

More generally, from specification to testing, the components of the CTF outlined in Section 3 assume no prior knowledge of the implementation of the SUT. Instead, the CTF requires the user to specify domain-specific details that are independent of the implementation. This shifts the nature of the burden placed on scientific software testers from being software-specific to domain-specific. In doing so, the CTF facilitates the application of state-of-the-art testing techniques, such as metamorphic testing, to scientific modelling software *without the user even necessarily knowing what a metamorphic relation or test is.* This has been demonstrated throughout the case studies.

The main expenditure of effort in applying the CTF is the gathering of domain expertise; the task of expressing knowledge in a causal DAG and regression model is comparatively straightforward and involves limited effort. Furthermore, compared to other model-based testing techniques, the barrier to entry for the CTF is better suited to the typical skill set of scientific model users.

Our work is based on the contention that the effort required to employ the CTF is not unreasonable and that, relative to most model-based testing techniques, the necessary expertise are more familiar to a typical scientific model user [51]. Namely, the ability to elicit anticipated cause-effect relations in a causal DAG and familiarity with basic regression modelling techniques. However, to precisely quantify and empirically evaluate the feasibility and practicality of the approach, future work will look to conduct a human study in which various scientific developers apply the CTF to a range of scientific software.

6.4 Summary

Collectively, our answers to **RQ1** and **RQ2** suggest that the CTF offers an accurate and efficient approach that addresses several of the challenges associated with the testing of scientific software outlined by Kanewala and Bieman [51]. Most notably, through the ability to infer metamorphic test outcomes from small amounts of existing observational data, the CTF mitigates the prohibitively long execution times that typically prevent adequate testing of scientific software. Consequently, the CTF also increases the applicability of metamorphic testing to scientific software, helping to indirectly alleviate the test oracle problem [11]

Of course, the accuracy and efficiency offered by the CTF come at a cost. Our answer to **RQ3** suggests that the CTF presents a trade-off between practical effort and accuracy/efficiency: by leveraging causal knowledge and domain expertise, the CTF can apply SMT in situations where it is currently impractical. However, these domain expertise can be difficult to obtain for non-domain experts. In the case studies, we found the main expenditure of human effort to be in collecting the domain expertise necessary to apply the techniques; the process of converting these into a DAG and regression model required considerably less effort.

6.5 Additional Findings

Throughout our case studies, we also identified a number of additional findings that warrant discussion. First, we discuss the need for explainability and how causal DAGs help to address this. Second, we discuss a pair of bugs identified in the case studies using the CTF.

Explainability. When testing scientific software, the reasoning behind a particular test passing or failing (i.e. the test oracle procedure) is rarely made explicit. For example, modellers often use regression testing to check whether changes to the SUT have affected model predictions or results. Any deviations are then typically validated by a domain expert. This form of ad-hoc validation lacks transparency and, as such, cannot be easily interrogated by prospective users of the SUT. For applications such as infectious disease modelling, where software outputs may inform important policy decisions, there is a need for accountable and explainable test results. Explainability is also a topic of growing concern in fields such as healthcare [48] that are increasingly using black-box machine learning techniques but require transparent, accountable, and interpretable decision making [15].

To this end, the CTF incorporates *explainability* into the testing process. Specifically, by utilising causal DAGs for CI, the CTF includes a lightweight and transparent artefact that partially explains the reasoning behind reaching a particular test outcome (i.e. why a specific adjustment set, and therefore statistical model, yields a *causal* estimate). Furthermore, the causal test case (Definition 3.5) includes an explicit test oracle (Definition 3.7) that captures 'correctness' in terms of some causal metric, such as the *ATE* or *RR*. Both assets can be easily accessed and interrogated, increasing the explainability and reputability of tests.

With this built-in notion of explainability, we posit that the CTF also has the potential to complement existing techniques in the scientific modelling context that often rely on implicit domain expertise for testing, such as regression testing. However, the causal DAG and test oracle do not communicate all assumptions with the potential to influence test results and their interpretation. For example, the anticipated functional form of a particular cause-effect relationship will influence the design of the regression model and its resulting predictions. A potential avenue for future work would be to investigate methods for improving the explainability of the CTF. For example, one could look into more expressive graphical models of causality that capture the expected functional form.

Bugs Found. Our case studies also revealed two interesting, previously undiscovered bugs in two of the studied scientific models: the Poisson Line Tessellation model and Covasim.

First, in Section 5.1, we found that the relationship between intensity and number of polygons per unit area described in [42] was more fragile at smaller window sizes. This suggested that the window size (width and/or height) has a causal effect on the number of polygons per unit area, while [42] stated that these variables should be independent. We then designed a causal test case to confirm that increasing the window width from 1 to 2 whilst holding intensity constant *caused* a significant change in the number of polygons per unit area.

Second, in Section 5.3, we found a bug in Covasim's vaccine implementation where, upon prioritising the elderly for vaccination, the number of vaccinated individuals grew to nearly ten times the number of individuals in the simulation. While this does not appear to significantly affect the key outputs of the model, it is not difficult to imagine how such a bug could lead to an overestimation of the effects of interventions.

6.6 Threats to Validity

Our evaluative case studies in Section 5 do not claim to make generalisable conclusions regarding the accuracy, efficiency, and effort associated with the CTF. Instead, these case studies serve as proofs of concept that show - for the studied subject systems - how formulating metamorphic testing as a CI problem makes it possible to apply the approach in situations where conventional metamorphic testing methods are impractical. Nonetheless, there are some threats to validity worth considering here.

6.6.1 External Validity. In this work, the main threat to external validity is that our case studies only cover three subject systems involving a moderate number of input and output variables. As graphical CI requires domain expertise for the data-generating mechanism in the form of a causal DAG, a significant amount of time was spent familiarising ourselves with the subject systems and understanding their constituent cause-effect relationships. As a result, this limited our ability to systematically collect and analyse large numbers of varied subject systems.

Furthermore, our subject systems were all implemented in Python. Therefore, our findings do not necessarily generalise to scientific modelling software implemented in other languages. However, the CTF only requires execution data in CSV format to perform causal testing observationally and can thus be applied, in theory, to tabular data produced by *any* scientific model.

As a consequence of the aforementioned threats to external validity, we acknowledge that our results may not generalise to *all* forms of scientific modelling software. However, we attempt to mitigate the aforementioned threats to external validity by selecting models that differ in their complexity, subject matter, and modelling paradigm. In addition, as discussed in Section 5, the selected systems have important but vastly different applications across a variety of domains, and have all been the subject of prior research.

6.6.2 Internal Validity. In this paper, the main threat to internal validity is that we did not optimise the estimators and configuration parameters thereof for our case studies. While this avoids the problem of over-fitting, it means there may exist statistical models that are more suitable for modelling and inferring the behaviour of the input-output relationships under study.

In the same vein, we specified regression equations that capture the expected functional form of various input-output relationships. For example, when testing Covasim in Section 5.4, we specified a regression model which captures our broad understanding of how cumulative infections vary with various causally relevant parameters. We called upon our experience with the models and subject area to specify these equations. However, different domain experts may have different opinions about the correct functional forms of the input-out relationships and may therefore have specified these relationships differently or more accurately.

As a consequence of the above threats to internal validity, we acknowledge that there alternative statistical models may achieve more precise causal inferences for the subject systems. However, we partially mitigate the above threats to internal validity by manually inspecting the functional forms of the relationships between inputs and outputs of interest in the SUT. We achieve this by varying one parameter at a time and observing how the output in question changes in response (in a similar way to our sensitivity analysis case study in Section 5.2). We also include a more advanced regression model in **Appendix** that more accurately captures the relationship between transmissibility (β) and the number of cumulative infections in Covasim.

7 RELATED WORK

In this section, we provide a brief review of work related to the two main topics concerning our paper: approaches for testing scientific software and causality in software testing. Additionally, we summarise automatic approaches to generating causal DAGs and highlight a number of open research challenges.

7.1 Testing Techniques for Scientific Software

As stated in Kanewala and Bieman's survey [51], scientific models are seldom tested using systematic approaches. Instead, techniques such as sensitivity [73] and uncertainty analysis [33] are often employed to analyse and appraise scientific models. However, these approaches generally require many costly executions that make them prohibitively expensive at scale [27]. To address this issue, modellers have turned to emulator approaches [27, 87], where a surrogate model is developed to approximate the behaviour of the simulation and provide an efficient way to validate behaviour [19, 107]. However, these emulators are driven by statistical associations and are unable to draw *causal* inferences from existing test data.

Another issue that precludes the testing of scientific modelling software is the oracle problem [11]; the lack of a mechanism that can be used to ascertain whether the outcome of a test case is correct or not. Kanewala and Bieman's survey [51] identifies several approaches followed by scientific modellers to overcome the oracle problem, including: pseudo oracles, comparison to analytical solutions or experimental results, and expert judgement. In addition to these

solutions, modellers have also turned to metamorphic testing (see Section 2) to overcome the lack of oracle. This approach relies on the scientists being able to specify metamorphic relations capable of revealing faults. However, these relationships are notoriously challenging to identify [93].

To assist with the identification of metamorphic relations, Kanewala and Bieman developed a machine learning approach for predicting metamorphic relations for numerical software [50]. This is achieved by representing numerical functions as a statement-level control flow graph and extracting features from this graph to train a classifier. In recent years, several new approaches for automatically predicting metamorphic relations for a specific form of software have been proposed, including for cyber-physical systems [5, 6] and matrix calculation programs [85]. However, the generation of metamorphic relations remains a difficult problem with automatic solutions available for only a few specific forms of software.

7.2 Causality in Software Testing

In more conventional settings, CI techniques have been applied to the software testing problem of fault localisation (FL). Informally, FL concerns identifying locations of faults in a program [113] and often involves computing a "suspiciousness metric" for software components, such as program statements. However, these metrics are often confounded by other software components. To address this, Baah et al. [7] translated FL to a CI problem, using program dependence graphs as a model of causality to estimate the causal effects of program statements on the occurrence of faults. Subsequent papers build on this to handle additional sources of bias [8]; leverage more advanced statistical models [8, 84]; and adapt to different software components [9, 39, 84, 95].

More recently, Lee et al. have introduced the Causal Program Dependence Analysis Framework and applied it to FL. This is a CI-driven framework that measures the strength of dependence between program elements by modelling their causal structure [61]. Unlike previous CI-based FL techniques, this framework does not use static analysis to construct its underlying model of causality, and instead approximates the causal structure by observing the effects of interventions. In a series of experiments, the framework has been shown to outperform slicing-based and search-based FL techniques, and help developers focus on key dependencies. Furthermore, due to its focus on dependence relations instead of coverage, it is less susceptible to coincidental correctness (executions that pass but cover faulty components).

In a similar vein, software testing often involves understanding *why* a particular outcome occurs, such as a program failure. To this end, Johnson et al. [49], developed a tool that explains the root cause of faulty software behaviour. This tool creates "causal experiments" by mutating an existing test to form a suite of minimally different tests that, contrary to the original, are not fault-causing. The passing and failing tests can then be compared to understand *why* a fault occurred. Similarly, Chockler et al. [23] developed a tool to *explain* the decisions of deep neural network (DNN) image classifiers. Following the actual causes framework [43], this tool offers explanations in the form of minimal subsets of pixels sufficient for the DNN to classify an image.

Another software testing technique concerning causality is cause-effect graphing, a black-box approach adapted from hardware testing. Here, input-output relationships are expressed in a variant of a combinatorial logic network, known as a cause-effect graph, created by manually extracting causes, effects, and boolean constraints from natural language specifications [69, 72]. Unlike the previous techniques, this approach does not use CI.

Recent work presented in [37] frames software testing in terms of causal reasoning. The authors conceptualise an iterative approach for test case generation in which test cases and the causal DAG are generated together and used to improve each other. However, the work is still at a preliminary stage, and the important link between CI and metamorphic testing is not discussed.

7.3 Automatic Generation of Causal DAGs

In this paper, we have assumed that all causal DAGs are specified manually by a domain expert. While this is an intuitive and widely accepted approach for conducting CI in fields such as epidemiology and social sciences, there are two potential methods that could, in theory, (partially) automate this process.

First, under certain strict assumptions and with large quantities of data, it is possible to predict the structure of causal DAGs from observational data. Where *model inference* provides a source of models for traditional MBT techniques [105], the field of *causal discovery* (CD) [63] provides methods to infer causal structures from data by exploiting asymmetries that distinguish association from causation [38]. However, due to the need for large amounts of data and their strict assumptions, we have had limited success in applying CD algorithms to model execution data. We plan to investigate this route further in future work.

Second, causal DAGs can be generated via static analysis of source code. DAGs derived in this way have already been used for FL [61, 84]. However, this approach relies on source code being openly available and produces a detailed, low-level model of causality for the SUT. While this level of granularity is ideal for the purpose of FL, the resulting causal DAG would be less suitable for a typical scientific modeller.

In addition to the aforementioned challenges, there is a fundamental barrier to using automatically generated models of causality for testing: inferred models represent the implemented system rather than the true specification. In other words, even if we could perfectly recover the DAG of the implementation, this would contain any bugs the implementation may have. We would, in effect, be testing the system against itself, so it would trivially look correct. Hence, the correctness of any inferred DAGs must be verified by a domain expert.

7.4 Machine Learning-Inferred Models of Tested Behaviour

In this work, we employ causality-informed linear regression models to infer metamorphic test outcomes. This aspect of our work relates to a significant body of work on machine learning approaches for inferring models from test executions. While Weyuker started this line of research 40 years ago [111], it has become particularly active in the last decade.

Most testing approaches that incorporate machine learning do so in the context of regression testing, where the inferred model represents the correct behaviour that can be used to identify any faults arising in subsequent software versions. Such approaches often use off-the-shelf machine learning and regression algorithms, chosen to fit the characteristics of the software behaviour in question. These have included standard linear regression [3], state machine inference [109], and decision trees [13] amongst others.

Such approaches are applicable to situations where (a) there is an established, reasonably correct system in place to derive tests from, and (b) there is a sufficiently large and diverse amount of execution data available. In our case, neither of these conditions holds. The computational models we analyse are exploratory in nature, and would not serve as a reliable oracle in their own right. Instead, we depend on causal properties provided by the developer in the DAG. Furthermore, computational models are subject to the various restrictions described in Section 2.2 - namely, high execution times, large and complex input spaces, and high computational costs. These restrictions prevent us from collecting a set of executions that is sufficiently large and diverse to accurately characterise the underlying behaviour.

8 CONCLUSION AND FUTURE WORK

In this paper, we presented the Causal Testing Framework (CTF): a conceptual framework that facilitates the application of causal inference (CI) techniques to software testing problems. This framework follows a model-based testing approach

to incorporate an explicit model of causality into the software testing process in the form of a causal DAG, enabling the direct application of graphical CI methods to software testing activities. Due to its fundamentally causal nature, we took a particular focus on metamorphic testing in this work.

A key contribution of the CTF is its ability to infer metamorphic test outcomes from previous execution data, despite the presence of confounding, providing an efficient method for testing scientific models in situations where it is currently impractical or infeasible. To demonstrate this, we applied our open source reference implementation of the CTF to three real-world scientific models of varying size and complexity, including a Poisson line tessellation model, a cardiac action potential model, and an epidemiological agent-based model. The results of these case studies suggest that, through the use of CI, the CTF can accurately infer metamorphic test outcomes from existing test data using significantly less data than is required by a conventional statistical metamorphic testing approach.

Software testing is an inherently causal process, and the field of CI holds much-untapped potential. To this end, the CTF lays the foundation for a new line of causality-driven software testing techniques. In one line of future work, we plan to apply the CTF to more causality-led testing activities, such as regression testing and A/B testing, to better understand how CI can support different testing activities. A separate direction of research would be to establish a (semi-)automatic, reliable process for the discovery of causal DAGs representing software systems. Such an artefact could be used as a starting point for a causal specification, reducing the amount of human effort required to apply the CTF and thus lower the barrier to entry.

ACKNOWLEDGMENTS

Foster, Walkinshaw, and Hierons are funded by the EPSRC CITCoM grant EP/T030526/1. For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY)[16] licence to any Author Accepted Manuscript version arising.

REFERENCES

[1] Jason Abrevaya, Yu-Chin Hsu, and Robert P Lieli. 2015. Estimating conditional average treatment effects. *Journal of Business & Economic Statistics* 33, 4 (2015), 485–505.

[2] Clement Adebamowo, Oumou Bah-Sow, Fred Binka, Roberto Bruzzone, Arthur Caplan, Jean-François Delfraissy, David Heymann, Peter Horby, Pontiano Kaleebu, Jean-Jacques Muyembe Tamfum, et al. 2014. Randomised controlled trials for Ebola: practical and ethical issues. *The Lancet* 384, 9952 (2014), 1423–1424.

[3] Aitor Arrieta, Jon Ayerdi, Miren Illarramendi, Aitor Agirre, Goiuria Sagardui, and Maite Arratibel. 2021. Using machine learning to build test oracles: an industrial case study on elevators dispatching algorithms. In *2021 IEEE/ACM International Conference on Automation of Software Test (AST)*. IEEE, 30–39.

[4] Susan Athey and Stefan Wager. 2019. Estimating treatment effects with causal forests: An application. *Observational Studies* 5, 2 (2019), 37–51.

[5] Jon Ayerdi, Valerio Terragni, Aitor Arrieta, Paolo Tonella, Goiuria Sagardui, and Maite Arratibel. 2021. Generating Metamorphic Relations for Cyber-Physical Systems with Genetic Programming: An Industrial Case Study. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Athens, Greece) *(ESEC/FSE 2021)*. Association for Computing Machinery, New York, NY, USA, 1264–1274. https://doi.org/10.1145/3468264.3473920

[6] Jon Ayerdi, Valerio Terragni, Aitor Arrieta, Paolo Tonella, Goiuria Sagardui, and Maite Arratibel. 2022. Evolutionary Generation of Metamorphic Relations for Cyber-Physical Systems. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Boston, Massachusetts) *(GECCO '22)*. Association for Computing Machinery, New York, NY, USA, 15–16. https://doi.org/10.1145/3520304.3534077

[7] George K. Baah, Andy Podgurski, and Mary Jean Harrold. 2010. Causal Inference for Statistical Fault Localization. In *Proceedings of the 19th International Symposium on Software Testing and Analysis* (Trento, Italy) *(ISSTA '10)*. Association for Computing Machinery, New York, NY, USA, 73–84. https://doi.org/10.1145/1831708.1831717

[16] Where permitted by UKRI a CC-BY-ND licence may be stated instead.

[8] George K. Baah, Andy Podgurski, and Mary Jean Harrold. 2011. Mitigating the Confounding Effects of Program Dependences for Effective Fault Localization. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering* (Szeged, Hungary) *(ESEC/FSE '11)*. Association for Computing Machinery, New York, NY, USA, 146–156. https://doi.org/10.1145/2025113.2025136

[9] Zhuofu Bai, Gang Shu, and Andy Podgurski. 2015. NUMFL: Localizing Faults in Numerical Software Using a Value-Based Causal Model. In *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 1–10. https://doi.org/10.1109/ICST.2015.7102597

[10] Elias Bareinboim and Judea Pearl. 2016. Causal inference and the data-fusion problem. *Proceedings of the National Academy of Sciences of the United States of America* 113, 27 (2016), 7345–7352. https://www.jstor.org/stable/26470690

[11] Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. 2015. The Oracle Problem in Software Testing: A Survey. *IEEE Transactions on Software Engineering* 41, 5 (2015), 507–525. https://doi.org/10.1109/TSE.2014.2372785

[12] Kenneth Benoit. 2011. Linear regression models with logarithmic transformations. *London School of Economics, London* 22, 1 (2011), 23–36.

[13] Lionel C Briand, Yvan Labiche, Zaheer Bawar, and Nadia Traldi Spido. 2009. Using machine learning to refine category-partition test specifications and test suites. *Information and Software Technology* 51, 11 (2009), 1551–1564.

[14] Manfred Broy, Bengt Jonsson, Joost-Pieter Katoen, Martin Leucker, and Alexander Pretschner (Eds.). 2005. *Model-Based Testing of Reactive Systems, Advanced Lectures [The volume is the outcome of a research seminar that was held in Schloss Dagstuhl in January 2004]*. Lecture Notes in Computer Science, Vol. 3472. Springer. https://doi.org/10.1007/b137241

[15] Nadia Burkart and Marco F Huber. 2021. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research* 70 (2021), 245–317.

[16] Michael J. Butler, Philipp Körner, Sebastian Krings, Thierry Lecomte, Michael Leuschel, Luis-Fernando Mejia, and Laurent Voisin. 2020. The First Twenty-Five Years of Industrial Use of the B-Method. In *Formal Methods for Industrial Critical Systems - 25th International Conference, FMICS 2020, Vienna, Austria, September 2-3, 2020, Proceedings (Lecture Notes in Computer Science, Vol. 12327)*, Maurice H. ter Beek and Dejan Nickovic (Eds.). Springer, 189–209.

[17] Nancy Cartwright and Eileen Munro. 2010. The limitations of randomized controlled trials in predicting effectiveness. *Journal of evaluation in clinical practice* 16 2 (2010), 260–6.

[18] cellML. 2022. cellML: Luo-Rudy 1991. https://models.cellml.org/exposure/456b07d6a7a5b45ed71caad0ea2c0b9d.

[19] Eugene TY Chang, Mark Strong, and Richard H Clayton. 2015. Bayesian sensitivity analysis of a cardiac cell model using a Gaussian process emulator. *PloS one* 10, 6 (2015), e0130252.

[20] Tsong Y. Chen, Shing C. Cheung, and Shiu Ming Yiu. 1998. *Metamorphic testing: A new approach for generating next test cases.* Technical Report HKUST-CS98-01. The Hong Kong University of Science and Technology.

[21] Vishnu Vardhan Chetlur and Harpreet S. Dhillon. 2018. Coverage Analysis of a Vehicular Network Modeled as Cox Process Driven by Poisson Line Process. *IEEE Transactions on Wireless Communications* 17, 7 (2018), 4401–4416. https://doi.org/10.1109/TWC.2018.2824832

[22] Sung Nok Chiu, Dietrich Stoyan, W. S. Kendall, and Joseph Mecke. 2013. *Stochastic Geometry and its Applications* (3rd ed.). John Wiley & Sons Inc, Chichester, West Sussex, United Kingdom.

[23] Hana Chockler, Daniel Kroening, and Youcheng Sun. 2021. Explanations for Occluded Images. https://doi.org/10.48550/ARXIV.2103.03622

[24] Tsun S. Chow. 1978. Testing software design modeled by finite-state machines. *IEEE transactions on software engineering* 3 (1978), 178–187.

[25] Carlos Cinelli and Chad Hazlett. 2020. Making sense of sensitivity: Extending omitted variable bias. *Journal of the Royal Statistical Society Series B-Statistical Methodology* 82, 1 (2020), 39–67.

[26] Jamie A. Cohen, Dina Mistry, Cliff C. Kerr, and Daniel J. Klein. 2020. Schools are not islands: Balancing COVID-19 risk and educational benefits using structural and temporal countermeasures. https://doi.org/10.1101/2020.09.08.20190942

[27] Stefano Conti and Anthony O'Hagan. 2010. Bayesian emulation of complex multi-output and dynamic computer models. *Journal of statistical planning and inference* 140, 3 (2010), 640–651.

[28] Jerome Cornfield, William Haenszel, E. Cuyler Hammond, Abraham M. Lilienfeld, Michael B. Shimkin, and Ernst L. Wynder. 1959. Smoking and Lung Cancer: Recent Evidence and a Discussion of Some Questions. *JNCI: Journal of the National Cancer Institute* 22, 1 (01 1959), 173–203. https://doi.org/10.1093/jnci/22.1.173 arXiv:https://academic.oup.com/jnci/article-pdf/22/1/173/2704718/22-1-173.pdf

[29] Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An Efficient SMT Solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, C. R. Ramakrishnan and Jakob Rehof (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 337–340.

[30] Jared L. Deutsch and Clayton V. Deutsch. 2012. Latin hypercube sampling with multidimensional uniformity. *Journal of Statistical Planning and Inference* 142, 3 (2012), 763–772. https://doi.org/10.1016/j.jspi.2011.09.016

[31] J. Dick and A. Faivre. 1993. Automating the generation and sequencing of test cases from model-based specifications. In *FME '93, First International Symposium on Formal Methods in Europe*. Springer-Verlag, Lecture Notes in Computer Science 670, Odense, Denmark, 268–284.

[32] John Ellson, Emden Gansner, Lefteris Koutsofios, Stephen C. North, and Gordon Woodhull. 2002. Graphviz— Open Source Graph Drawing Tools. In *Graph Drawing*, Petra Mutzel, Michael Jünger, and Sebastian Leipert (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 483–484.

[33] Ismail Farajpour and Sez Atamturktur. 2013. Error and uncertainty analysis of inexact and imprecise computer models. *Journal of Computing in Civil Engineering* 27, 4 (2013), 407–418.

[34] Institute for Disease Modeling. 2022. Covasim: Vaccine Tests. https://github.com/InstituteforDiseaseModeling/covasim/blob/master/tests/test_interventions.py.

[35] Institute for Disease Modelling. 2022. Covasim. https://github.com/InstituteforDiseaseModeling/covasim.

[36] Marie-Claude Gaudel. 1995. Testing can be formal Too. In *6th International Joint Conference CAAP/FASE Theory and Practice of Software Development (TAPSOFT'95) (Lecture Notes in Computer Science, Vol. 915)*. Springer, 82–96.

[37] Luca Giamattei, Roberto Pietrantuono, and Stefano Russo. 2023. Reasoning-Based Software Testing. https://doi.org/10.48550/ARXIV.2303.01302

[38] Clark Glymour, Kun Zhang, and Peter Spirtes. 2019. Review of causal discovery methods based on graphical models. *Frontiers in genetics* 10 (2019), 524.

[39] Ross Gore and Paul F. Reynolds. 2012. Reducing confounding bias in predicate-level statistical debugging metrics. In *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 463–473. https://doi.org/10.1109/ICSE.2012.6227169

[40] S Greenland, J Pearl, and J M Robins. 1999. Causal diagrams for epidemiologic research. *Epidemiology* 10, 1 (Jan. 1999), 37–48.

[41] Michael H Grider, Rishita Jessu, and Rian Kabir. 2019. Physiology, action potential. (2019).

[42] Ralph Guderlei and Johannes Mayer. 2007. Statistical Metamorphic Testing Testing Programs with Random Output by Means of Statistical Hypothesis Tests and Metamorphic Testing. In *Seventh International Conference on Quality Software (QSIC 2007)*. IEEE, 404–409. https://doi.org/10.1109/QSIC.2007.4385527

[43] Joseph Y Halpern and Judea Pearl. 2005. Causes and explanations: A structural-model approach. Part I: Causes. *The British Journal for the Philosophy of Science* 56, 4 (2005), 843–887.

[44] Miguel A Hernán and James M Robins. 2020. *Causal Inference: what if*. Chapman & Hall/CRC, Boca Raton.

[45] R. M. Hierons. 1997. Testing from a Z specification. *The Journal of Software Testing, Verification and Reliability* 7, 1 (1997), 19–33.

[46] Robert M. Hierons, Kirill Bogdanov, Jonathan P. Bowen, Rance Cleaveland, John Derrick, Jeremy Dick, Marian Gheorghe, Mark Harman, Kalpesh Kapoor, Paul Krause, Gerald Lüttgen, Anthony J. H. Simons, Sergiy A. Vilkomir, Martin R. Woodward, and Hussein Zedan. 2009. Using formal specifications to support testing. *Comput. Surveys* 41, 2 (2009), 9:1–9:76.

[47] Paul W Holland. 1986. Statistics and causal inference. *Journal of the American statistical Association* 81, 396 (1986), 945–960.

[48] Andreas Holzinger, Georg Langs, Helmut Denk, Kurt Zatloukal, and Heimo Müller. 2019. Causability and explainability of artificial intelligence in medicine. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 9, 4 (2019), e1312.

[49] Brittany Johnson, Yuriy Brun, and Alexandra Meliou. 2020. Causal testing: understanding defects' root causes. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. IEEE, 87–99.

[50] Upulee Kanewala and James M Bieman. 2013. Using machine learning techniques to detect metamorphic relations for programs without test oracles. In *2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, IEEE, 1–10.

[51] Upulee Kanewala and James M. Bieman. 2014. Testing scientific software: A systematic literature review. *Information and Software Technology* 56, 10 (2014), 1219–1232. https://doi.org/10.1016/j.infsof.2014.05.006

[52] Luke Keele. 2015. The Statistics of Causal Inference: A View from Political Methodology. *Political Analysis* 23, 3 (2015), 313–335. https://doi.org/10.1093/pan/mpv007

[53] Diane Kelly and Rebecca Sanders. 2008. The challenge of testing scientific software. , 30–36 pages.

[54] John Kendall. 2003. Designing a research project: randomised controlled trials and their principles. *Emergency medicine journal: EMJ* 20, 2 (2003), 164.

[55] Cliff C Kerr, Dina Mistry, Robyn M Stuart, Katherine Rosenfeld, Gregory R Hart, Rafael C Núñez, Jamie A Cohen, Prashanth Selvaraj, Romesh G Abeysuriya, Michałł Jastrzębski, et al. 2021. Controlling COVID-19 via test-trace-quarantine. *Nature Communications* 12, 1 (2021), 1–12.

[56] Cliff C Kerr, Robyn M Stuart, Dina Mistry, Romesh G Abeysuriya, Katherine Rosenfeld, Gregory R Hart, Rafael C Núñez, Jamie A Cohen, Prashanth Selvaraj, Brittany Hagedorn, et al. 2021. Covasim: an agent-based model of COVID-19 dynamics and interventions. *PLOS Computational Biology* 17, 7 (2021), e1009149.

[57] Jack PC Kleijnen. 1995. Verification and validation of simulation models. *European journal of operational research* 82, 1 (1995), 145–162.

[58] Rex B Kline. 2015. *Principles and practice of structural equation modeling*. Guilford publications.

[59] Konstantin Kreyman, David Lorge Parnas, and Sanzheng Qiao. 1999. Inspection procedures for critical programs that model physical phenomena.

[60] David Lee and Mihalis Yannakakis. 1996. Principles and Methods of Testing Finite-State Machines - A Survey. *Proc. IEEE* 84, 8 (1996), 1089–1123.

[61] Seongmin Lee, Dave Binkley, Robert Feldt, Nicolas Gold, and Shin Yoo. 2021. Causal program dependence analysis.

[62] Ching-Hsing Luo and Yoram Rudy. 1991. A model of the ventricular cardiac action potential. Depolarization, repolarization, and their interaction. *Circulation Research* 68, 6 (1991), 1501–1526.

[63] Daniel Malinsky and David Danks. 2018. Causal discovery algorithms: A practical guide. *Philosophy Compass* 13, 1 (2018), e12470.

[64] Lawrence C Marsh and David R Cormier. 2001. *Spline regression models*. Number 137. Sage.

[65] E. F. Moore. 1956. Gedanken-Experiments. In *Automata Studies*, C. Shannon and J. McCarthy (Eds.). Princeton University Press.

[66] Frédéric Morlot. 2012. A population model based on a Poisson line tessellation. In *2012 10th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*. IEEE, 337–342.

[67] Sahil Moza. 2020. sahilm89/lhsmdu: Latin Hypercube Sampling with Multi-Dimensional Uniformity (LHSMDU): Speed Boost minor compatibility fixes. https://doi.org/10.5281/zenodo.3929531

[68] Gail C Murphy, David Notkin, and Kevin Sullivan. 1995. Software reflexion models: Bridging the gap between source and high-level models. In *Proceedings of the 3rd ACM SIGSOFT symposium on Foundations of software engineering*. IEEE, 18–28.

[69] Glenford J Myers, Tom Badgett, Todd M Thomas, and Corey Sandler. 2004. *The Art of Software Testing*. Vol. 2. Wiley Online Library.

[70] Josh W Nevin, FJ Vaquero-Caballero, David J Ives, and Seb J Savory. 2021. Physics-informed Gaussian process regression for optical fiber communication systems. *Journal of Lightwave Technology* 39, 21 (2021), 6833–6844.

[71] Srinivas Nidhra and Jagruthi Dondeti. 2012. Black box and white box testing techniques-a literature review. *International Journal of Embedded Systems and Applications (IJESA)* 2, 2 (2012), 29–50.

[72] Khenaidoo Nursimulu and Robert L. Probert. 1995. Cause-Effect Graphing Analysis and Validation of Requirements. In *Proceedings of the 1995 Conference of the Centre for Advanced Studies on Collaborative Research* (Toronto, Ontario, Canada) *(CASCON '95)*. IBM Press, 46.

[73] Jeremy E Oakley and Anthony O'Hagan. 2004. Probabilistic sensitivity analysis of complex models: a Bayesian approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 66, 3 (2004), 751–769.

[74] Sheila F O'Brien and Qi Long Yi. 2016. How do I interpret a confidence interval? *Transfusion* 56, 7 (2016), 1680–1683.

[75] Marie Oldfield and Ella Haig. 2021. Analytical modelling and UK Government policy. *AI and Ethics* 2, 3 (jul 2021), 389–404. https://doi.org/10.1007/s43681-021-00078-9

[76] Jasmina Panovska-Griffiths, Cliff C Kerr, Robyn M Stuart, Dina Mistry, Daniel J Klein, Russell M Viner, and Chris Bonell. 2020. Determining the optimal strategy for reopening schools, the impact of test and trace interventions, and the risk of occurrence of a second COVID-19 epidemic wave in the UK: a modelling study. *The Lancet Child & Adolescent Health* 4, 11 (2020), 817–827.

[77] Jasmina Panovska-Griffiths, Cliff C Kerr, William Waites, Robyn Margaret Stuart, Dina Mistry, Derek Foster, Daniel J Klein, Russell M Viner, and Chris Bonell. 2020. The potential contribution of face coverings to the control of SARS-CoV-2 transmission in schools and broader society in the UK: a modelling study.

[78] Judea Pearl. 1995. Causal diagrams for empirical research. *Biometrika* 82, 4 (12 1995), 669–688. https://doi.org/10.1093/biomet/82.4.669 arXiv:https://academic.oup.com/biomet/article-pdf/82/4/669/698263/82-4-669.pdf

[79] Judea Pearl. 2009. Causal inference in statistics: An overview. *Statistics Surveys* 3 (2009), 96–146. https://doi.org/10.1214/09-SS057

[80] Judea Pearl. 2009. *Causality*. Cambridge university press, Cambridge.

[81] Judea Pearl. 2018. Does Obesity Shorten Life? Or is it the Soda? On Non-manipulable Causes. *Journal of Causal Inference* 6, 2 (2018), 20182001. https://doi.org/10.1515/jci-2018-2001

[82] Judea Pearl and Dana Mackenzie. 2018. *The Book of Why*. Allen Lane.

[83] Judea Pearl and Thomas S Verma. 1995. A theory of inferred causation. In *Studies in Logic and the Foundations of Mathematics*. Vol. 134. Elsevier, 789–811.

[84] Andy Podgurski and Yiğit Küçük. 2020. CounterFault: Value-Based Fault Localization by Modeling and Predicting Counterfactual Outcomes. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 382–393.

[85] Karishma Rahman and Upulee Kanewala. 2018. Predicting Metamorphic Relations for Matrix Calculation Programs. In *2018 IEEE/ACM 3rd International Workshop on Metamorphic Testing (MET)*. IEEE, 10–13.

[86] Paul Ralph. 2021. ACM SIGSOFT Empirical Standards Released. *SIGSOFT Softw. Eng. Notes* 46, 1 (feb 2021), 19. https://doi.org/10.1145/3437479.3437483

[87] Carl Edward Rasmussen. 2004. Gaussian Processes in Machine Learning. In *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 63–71. https://doi.org/10.1007/978-3-540-28650-9_4

[88] Carl Edward Rasmussen, Christopher KI Williams, et al. 2006. *Gaussian processes for machine learning*. Vol. 1. Springer.

[89] Kenneth J Rothman and Sander Greenland. 2005. Causation and causal inference in epidemiology. *American journal of public health* 95, S1 (2005), S144–S150.

[90] Donald B Rubin. 2005. Causal inference using potential outcomes: Design, modeling, decisions. *J. Amer. Statist. Assoc.* 100, 469 (2005), 322–331.

[91] Fanny Sarrazin, Francesca Pianosi, and Thorsten Wagener. 2016. Global Sensitivity Analysis of environmental models: Convergence and validation. *Environmental Modelling & Software* 79 (2016), 135–152.

[92] Nick Scott, Anna Palmer, Dominic Delport, Romesh Abeysuriya, Robyn Stuart, Cliff C Kerr, Dina Mistry, Daniel J Klein, Rachel Sacks-Davis, Katie Heath, et al. 2020. Modelling the impact of reducing control measures on the COVID-19 pandemic in a low transmission setting. *Med J Aust* 214, 2 (2020), 79–83.

[93] Sergio Segura, Gordon Fraser, Ana B Sanchez, and Antonio Ruiz-Cortés. 2016. A survey on metamorphic testing. *IEEE Transactions on software engineering* 42, 9 (2016), 805–824.

[94] Dongeek Shin, Ahmed Kirmani, Andrea Colaço, and Vivek K Goyal. 2013. Parametric Poisson process imaging. In *2013 IEEE Global Conference on Signal and Information Processing*. IEEE, IEEE, 1053–1056.

[95] Gang Shu, Boya Sun, Andy Podgurski, and Feng Cao. 2013. Mfl: Method-level fault localization with causal inference. In *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*. IEEE, IEEE, 124–133.

[96] J. M. Spivey. 1992. *The Z Notation: A Reference Manual* (2nd ed.). Prentice-Hall.

[97] Matt Staats, Michael W Whalen, and Mats PE Heimdahl. 2011. Programs, tests, and oracles: the foundations of testing revisited. In *2011 33rd international conference on software engineering (ICSE)*. IEEE, IEEE, 391–400.

[98] Michael Stein. 1987. Large sample properties of simulations using Latin hypercube sampling. *Technometrics* 29, 2 (1987), 143–151.

[99] James H Stock, Mark W Watson, et al. 2003. *Introduction to econometrics*. Vol. 104. Addison Wesley Boston.

[100] Robyn M Stuart, Romesh G Abeysuriya, Cliff C Kerr, Dina Mistry, Daniel J Klein, Richard Gray, Margaret Hellard, and Nick Scott. 2020. The role of masks in reducing the risk of new waves of COVID-19 in low transmission settings: a modeling study.

[101] Peter WG Tennant, Eleanor J Murray, Kellyn F Arnold, Laurie Berrie, Matthew P Fox, Sarah C Gadd, Wendy J Harrison, Claire Keeble, Lynsie R Ranker, Johannes Textor, et al. 2021. Use of directed acyclic graphs (DAGs) to identify confounders in applied health research: review and recommendations. *International Journal of Epidemiology* 50, 2 (2021), 620–632.

[102] Robin N Thompson. 2020. Epidemiological models are important tools for guiding COVID-19 interventions. *BMC medicine* 18, 1 (2020), 1–4.

[103] Jan Tretmans. 2008. Model Based Testing with Labelled Transition Systems. In *Formal Methods and Testing (Lecture Notes in Computer Science, Vol. 4949)*. Springer, 1–38.

[104] Mark Utting and Bruno Legeard. 2010. *Practical model-based testing: a tools approach*. Elsevier.

[105] Mark Utting, Alexander Pretschner, and Bruno Legeard. 2012. A taxonomy of model-based testing approaches. *Software Testing, Verification and Reliability* 22, 5 (2012), 297–312.

[106] Tyler J VanderWeele and Peng Ding. 2017. Sensitivity analysis in observational research: introducing the E-value. *Annals of internal medicine* 167, 4 (2017), 268–274.

[107] Ian Vernon, Michael Goldstein, and Richard Bower. 2014. Galaxy Formation: Bayesian History Matching for the Observable Universe. *Statist. Sci.* 29, 1 (2014), 81 – 90. https://doi.org/10.1214/12-STS412

[108] Stefan Wager and Susan Athey. 2018. Estimation and Inference of Heterogeneous Treatment Effects using Random Forests. *J. Amer. Statist. Assoc.* 113, 523 (2018), 1228–1242. https://doi.org/10.1080/01621459.2017.1319839

[109] Neil Walkinshaw, Ramsay Taylor, and John Derrick. 2016. Inferring extended finite state machine models from software executions. *Empirical Software Engineering* 21 (2016), 811–853.

[110] Elaine Weyuker. 1982. On Testing Non-Testable Programs. *Computer Journal* 25 (11 1982). https://doi.org/10.1093/comjnl/25.4.465

[111] Elaine J Weyuker. 1983. Assessing test data adequacy through program inference. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 5, 4 (1983), 641–655.

[112] Christof Wolf and Henning Best. 2013. The SAGE handbook of regression analysis and causal inference. *The SAGE Handbook of Regression Analysis and Causal Inference* (2013), 1–424.

[113] W Eric Wong, Ruizhi Gao, Yihao Li, Rui Abreu, and Franz Wotawa. 2016. A survey on software fault localization. *IEEE Transactions on Software Engineering* 42, 8 (2016), 707–740.

APPENDIX

A more advanced regression model for Covasim

In Section 5.4, we designed a regression model that broadly captures the expected relationship between cumulative infections and various causally relevant parameters, such as transmissibility β and household contacts C_H. This regression model uses conventional regression modelling techniques to specify the relationships of interest. Namely, quadratic terms, log transformations, and effect modifiers.

However, this model does not capture the relationship between β and cumulative infections perfectly because the relationship follows a sigmoid function (i.e. a characteristic S-shaped curve). Informally, we can explain this relationship as follows. Initially, when β is low, there are few infections because the rate of viral transmission is low. Then, as β increases past some critical threshold, an exponential growth in the transmission rate occurs. Eventually, enough of the population becomes infected and gains immunity or dies, rapidly reducing the rate of viral transmission. This sudden reduction causes cumulative infections to level off, completing the characteristic S shape.

One of the weaknesses of polynomial regression is its unpredictable tail behaviour [112]. This limitation is particularly problematic for modelling sigmoid relationships, where the tails are necessarily flat. To address this limitation, we employed a more advanced form of regression known as spline regression [64].

In short, spline regression involves constructing a piece-wise polynomial over contiguous regions of the data. Within each region, a separate polynomial function of degree n is fit to the subset of data. This approach to regression essentially breaks the problem into discrete stages and is an effective technique for capturing non-linear relationships. In many cases, a third-degree polynomial is used to model each region, in which case the resulting splines are referred to as cubic splines.

Based on our limited domain expertise, to capture the sigmoid relationship between β and cumulative infections, we used cubic splines with two (internal) knots. With this approach, our aim was to separate the data into three regions corresponding to the three distinct phases of the sigmoid function described above (initial slow growth in infections, exponential growth, and plateau in infections).

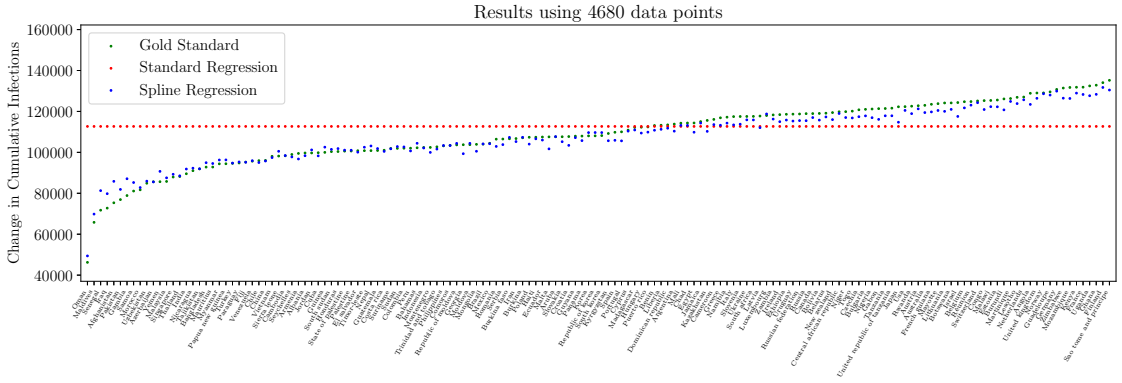

Fig. 12. A comparison of the metamorphic test outcomes predicted by a cubic spline regression with two knots and the naive regression.

Figure 12 shows the metamorphic test outcomes predicted using cubic spline regression. From an informal visual inspection, it is clear that the majority of estimates are more accurate than the previous regression model, which generally overestimated the effects and had a root mean square percentage error (*RMSPE* of 0.055) and a Kendall's rank correlation of 0.944 ($p < 0.005$). By contrast, the cubic spline approach had an *RMPSE* of 0.032 and a Kendall's rank correlation of 0.915 ($p < 0.005$). Therefore, the spline regression technique provided better absolute accuracy (indicated by *RMSPE*), but worse comparative accuracy (indicated by Kendall's rank correlation). The performance of both approaches could likely be improved by a domain expert who may have a more precise characterisation of the anticipated relationships.

We decided not to include the cubic splines approach in the case studies, as it requires more advanced statistical modelling knowledge that is unlikely to be commonplace to prospective users. However, it is worth including as an appendix because it introduces a potentially valuable trade-off. Namely, more advanced, semi-parametric statistical estimators can be employed with arguably less domain knowledge to learn intricate shapes from the available data. However, this introduces an additional burden: the need for expertise in such modelling techniques.

Overall, in this example, we were able to configure the spline regression model in a logical way that is justified by domain expertise (i.e. splitting the relationship into three key regions, each of which can be modelled with a cubic polynomial). This shows how more advanced statistical means can be employed to achieve better results. In future work, we will investigate the application of other semi- and non-parametric statistical models within the Causal Testing Framework.