ATHABASCA UNIVERSITY

A MATHEMATICAL DEFINITION OF A CODING PROCESS TO CAUSALLY ASSESS CODING COMPETENCE

BY

HANAN SALEH

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN INFORMATION SYSTEMS

FACULTY OF SCIENCE AND TECHNOLOGY

SCHOOL OF COMPUTING AND INFORMATION SYSTEMS

ATHABASCA UNIVERSITY

OCTOBER, 2023

Athabasca University
Faculty of Graduate Studies

Approval of Thesis

The undersigned certify that they have read the thesis entitled

A MATHEMATICAL DEFINITION OF A CODING PROCESS TO CAUSALLY ASSESS CODING COMPETENCE

Submitted by

Hanan Saleh

In partial fulfillment of the requirements for the degree of

Master of Science in Information Systems

The thesis examination committee certifies that the thesis
and the oral examination is approved

Supervisor:

Dr. Vivekanandan Kumar
Athabasca University

Dr. Gustavo Carrero
Athabasca University

External Examiner:
Dr. Ben Daniel Motidyang
Otago University

January 25, 2024

Dedication

This work is dedicated to my late father, Omar Haymour, whom I love and miss very much.

Acknowledgments

I want to begin by thanking Allah SWT for His endless mercy and grace for giving me the strength to complete this work.

This work would not have been possible without the support of my supervisors, Dr. Vive Kumar and Dr. Gustavo Carrero. Your contrasting approach to supervision has made this an incredibly enriching experience. Dr. Kumar, your calmness and words of encouragement gave me the confidence to shape my research in a way I saw fit. Your wisdom and ability to always see the 'bigger picture' served as both motivation and reassurance. Dr. Carrero, thank you for making me a better researcher and writer. You always raised challenging but meaningful questions. I will forever be grateful to you for constantly pushing me out of my comfort zone. Our engaging discussions helped me better understand my research and what I hoped to accomplish, even when I could not see it myself.

I would also like to thank my external examiner, Dr. Ben Daniel. Thank you for being so poised and thoughtful throughout the examination process.

To my mother, thank you for always being my biggest fan. I love you very much.

To my children, Tareq, Amira, Alia and Omar, I love you more than anything. I hope you can one day forgive me for missing so many 'family movie nights' because 'mom has work to do.'

To my husband, my rock, my number one supporter, I love and appreciate you.

Abstract

The search for causal relations from observational or experimental data is an open and pervasive problem that spans many fields of research. In the area of learning, this is especially important. The ability to determine the effect of a new teaching strategy or the cause of an upswing in student performance is persistently desirable. In computer science courses, integrated development environments (IDE) offer students a plethora of features promising to accelerate the coding process and instill the necessary competency skills for seamless migration to industry. In this work, we introduce a mathematical definition of a coding process and apply causal discovery methods to data collected from an IDE. We approached the problem statically and dynamically and found strong evidence of a causal effect of consultations on coding competency. We set forth the groundwork for future analyses of these processes and exhibit the extendibility of this work to other disciplines.

Keywords: coding process, causal discovery, time series data, dynamical systems

Table of Contents

List of Tables

List of Figures

Chapter 1.

1. Introduction

1.1 Causality

The motivation behind most research is either explorative or causal in nature (Pearl, 2009a). The former involves investigating new methodologies or techniques to solve existing problems in more effective ways. The latter involves answering questions that require causal reasoning based on data obtained from either randomized controlled trials or observations (Winship & Morgan, 1999).

Causal inference is the act of deducing that a certain intervention or treatment was the 'cause' of an observed 'effect' (Rubin & Zell, 2018). For instance, statements such as "if the pan is too hot, the pancakes will burn" or "the smoke alarm went off, so there must be a fire" are examples of some of the causal inference tasks we perform on a daily basis with little mental exertion. Human beings have been deriving causal connections for hundreds of years and the human brain is the supreme tool for this task. We make connections between cause and effect or reason and consequence with limited prior knowledge and little experimentation. These connections play a critical role in humanity's evolution and progress since knowledge of both causal connections and causal chains is the foundation of human reasoning, inference and decision-making (Chan et al., 2005; Pena et al., 2008).

However, causal connections are not always easy to derive from simply experimenting or observing. The world is a complex place, and most often, several variables are at play, rendering the ability to disentangle cause from effect difficult. In order to derive any causal conclusions effectively, we need a way to fix one variable while observing the change in others. Randomized control trials (RCT) are the true and trusted methodological approach to causal discovery (Winship & Morgan, 1999; Pearl

2009a; Pearl & Mackenzie, 2018). An RCT is an experimental design where subjects are randomly assigned to either a control or experimental group. Consequently, any difference in the outcome between both groups is due to the treatment variable. Despite being efficacious, RCTs can be expensive, longwinded and even unethical in some cases. In the era of big data, there is a growing impulse to determine causal relations from observational data alone. However, this data may be static, providing no indication of the temporal changes it underwent or the mechanisms that led to them. Associational methods such as correlation and dependence analysis can be conducted after the data collection process. However, these are shallow indicators of causality as they do not explain the data-generating process. An example of such spurious associations is the correlation between ice cream sales and crime rates. It would be nonsensical to think that crime rates and ice cream sales are causally related due to the correlation between them. Observations that appear to be related to one another may share a common driver, giving rise to the observations (Reichenbach, 1956). Dependence can manifest in the context of causations also. Two events can be causally dependent if they fall along the same causal chain (Lewis, 1974). Static observations can be thought of as a collection of points from a dynamic system at equilibrium (Dash, 2005). This is in violation of a crucial premise for causal discovery, which necessitates the observance of a dynamic change in system variables when one variable is fixed to a certain value (Peters et al., 2017).
Causal graphs offer a graphical approach to simulating distributional changes using both a graph and the observational distribution (Pearl, 2009a; Pearl, 2009b).

However, generating a graph that represents true system dependencies requires expert knowledge and awareness of all influencing variables (observed or unobserved). Despite these hindrances, several methods have been proposed and have produced successful results in various settings depending on the type of data and the problem at hand.

1.2 The Importance of Causality

The persistent desire for informed decision-making and the establishment of better policies spans many fields of interest. This is especially critical in highly sensitive domains such as human resources, criminal justice system and education, to name a few. Due to the complexity and intricacy of these domains, disentangling variable interactions and non-spurious dependencies is usually beyond the confines of human cognition. However, they still require an element of human reasoning.

The abundance of data and increased computational power has led to a heavy reliance on data-driven approaches for many inference tasks such as prediction, forecasting or the uncovering of relations between variables of interest (Hwang, 2018; Guo et al., 2020). Two of the most popular approaches to gaining such insights would be building a statistical or a machine-learning model from the data at hand. You can think of these models as simplified descriptions of the data. The former relies on a rich statistical toolbox of supervised and unsupervised methods such as regression and clustering. The latter requires learning a complex function that is trained to fit the data using a relevant optimization method such as gradient or stochastic descent. Both sets of models have produced impressive results in ideal settings where the data is independent and identically distributed (i.i.d.) (Schölkopf & von Kügelgen, 2022). The data, being the driving force of either methodology, is usually treated like a black box, and the generating process is not extensively considered. For instance, questions like: “How did this data come about?” or “What are the mechanisms that led to this data?” are not rigorously investigated. Observations are passive and do not encode any temporal structure or interventional information. Distributional shifts that are commonly present between different samples are typically ignored or engineered away and are combated by ‘dumping’ more data on the model as it is acquired. The more data that a model incorporates or sees during training, the better its performance during testing. This sets forth the need for stringent assumptions and recurrent hyper-parameter adjustments. Any

insights or variable relations derived from the data could be merely coincidental and not grounded on a robust theoretical base. An elegant model or an overarching theoretical framework can almost never be established and the ability of these models to generalize beyond the training distribution would be highly unlikely (Davison, 2003).

In contrast, a causal model offers much more than these models can (Pearl, 2018; Schölkopf et al., 2021). It can answer questions that are both descriptive and explanatory in nature on three cognitive levels (Pearl & Mackenzie, 2018; Wold, 1956). First, questions about the associations between variables of interest (this can also be done by any one of the above two mentioned models or directly from the data). Second, it can answer questions about interventions (if we manipulate X what will happen to Y). Third, it can answer counterfactual questions that are beyond our cognitive abilities as humans. Therefore, causal connections are far more powerful than mere associations. Any relations that a causal model encodes are invariant across different distributions (Peters et al. 2017; Guo et al. 2020), allowing us to generalize past the observational data. Therefore, the inferences that we are certified to draw from a statistical or machine learning model are inferior to those we are able to draw from a causal model. These causal inference tasks are more in alignment with human-level and can even surpass human-level abilities in more complex settings. The complexity of real-world scenarios and the intricate mechanisms that connect the variables, known and unbeknownst, can not be captured by statistical or machine-learning models (Pearl & Mackenzie, 2018).

1.3 Motivation

Computer science students spend a significant amount of time developing code in an integrated development environment (IDE). Students rely on these IDEs to expedite and streamline the development process. In some cases, an IDE can alleviate instructor overload by providing timely and meaningful feedback on how to optimize code and adhere to best practices. There is a rich body of data that can be collected by an IDE,

which can provide insight into the learning process and student coding habits. As such, this data can be analyzed and leveraged to discover causal models that will ultimately improve learner outcomes.

The main purpose of this work is to establish a causal link between didactic feedback provided in an integrated development environment (IDE) and student coding competence. We apply several causal discovery techniques to data collected from an IDE, where we consider the data in two different ways: statically and dynamically. In addition, we develop a mathematical definition of a coding process and the objects by which it is entailed, setting forth the groundwork for future analysis of such processes.

In Chapters 2-4, we provide the necessary background information. This includes a discussion of the importance of causality and its superiority over associational statistics. Associations can be spurious and, as a consequence, produce unreliable models that obfuscate decision-making. Then, we provide an overview of two prolific causal modelling frameworks, followed by a summary of pertinent causal discovery methods for static and dynamic data.

In Chapter 5, we discuss the importance of learning analytics in computer science and the presence of a rich body of data that can be collected from IDEs and leveraged for this purpose. In addition, we explain that these IDEs can be instrumented to provide meaningful interventions to facilitate learning.

In Chapter 6, we provide a detailed description of a coding process in general and define the objects it entails. In addition, we provide relevant definitions to the variables in our study.

In Chapter 7, we provide a detailed description of the data, including the collection process and any preprocessing it underwent before applying the various causal discovery techniques. We also present the results from each technique and a detailed discussion ensues in section seven.

We conclude with a list of recommendations for future studies (Chapter 8), the applicability of our approach in other disciplines (Chapter 9) and some concluding remarks about future research (Chapter 10).

Chapter 2.

2. Causal Models

There are two prolific and intuitive frameworks that are used for causal modelling: Structural Causal Models (SCM) and Causal Graphical Models (CGM) (Schölkopf & von Kügelgen, 2022). These models are closely related. A CGM relies on causal conditionals that can be calculated directly from the data using conditional probabilities. An SCM relies on functional assignments that can also be calculated from the data using methods such as ordinary least squares, where each variable is regressed on all its influencing factors. Any noise or errors are attributed to unobserved influencing factors. SCMs are more powerful as they are able to answer more questions than CGMs but are more difficult to generate.

2.1 Structural Causal Models

Let **S** be a system represented by the following sets:

$$\mathbf{V} = \{v_i\}_{i=1}^{i=m}$$
$$\mathbf{U} = \{u_i\}_{i=1}^{i=m}$$
$$\mathbf{F} = \{f_i(\mathbf{Pa}(v_i), u_i)\}_{i=1}^{i=m}$$

where **V** and **U** are the endogenous and exogenous variables of the system respectively. **F** is a set of functions inducing a probability distribution over **V**. $\mathbf{Pa}(v_i)$ is the set of nodes, a.k.a parents, that causally influence v_i such that $\mathbf{Pa}(v_i) = \{v_k \mid v_k \in \mathbf{V}, k \neq i\}$. By influence, we mean that if we observe a change in any $\mathbf{Pa}(v_i)$ then we will observe a change in v_i.

The system **S** represents a SCM.

Definition 2.0. Endogenous Variable. A variable $v_i \in \mathbf{V}$ is endogenous if it is *observed* and is causally influenced by zero or more endogenous variables $\mathbf{Pa}(v_i) = \{v_k \mid v_k \in \mathbf{V}, k \neq i\}$ of **S**.

Since the complexity of any real-world system can never be accurately captured and, therefore, modelled, we assume that each variable in **V** has an exogenous variable in **U** acting on it. Exogenous variables can be viewed as unexplained phenomena which account for prediction errors.

Definition 2.1. Exogenous Variable. A variable $u_i \in \mathbf{U}$ is exogenous if it is *unobserved* and causally influences one and only one endogenous variable $v_i \in \mathbf{V}$.

The presence of u_i is assumed, but the extent of its effect is not known., The elements of **U** are mutually independent of one another and exclusive to their counterparts in **V**. In other words, an element u_i of **U** can only influence v_i from **V** (Pearl, 2009b).

F is a set of functions such that for random variable v_i,

$$v_i = f_i(\mathbf{Pa}(v_i), u_i)$$

where f_i is a function (known or unknown) that encodes an invariant mechanism between the effect v_i and its cause(s), $\mathbf{Pa}(v_i)$ and u_i. These equations are not like a typical algebraic equation where variables can be moved between either side of the equal sign but rather are interpreted as assignments (Pearl & Mackenzie 2018; Heinze-Deml et al. 2018; Peters et al. 2017). They are called structural equations since they define the generating process.

Definition 2.2. Structural Equation. A structural equation of a variable v_i is an equation that defines a complex relationship in the form of a function f_i between v_i and all

variables **Pa**(v_i) that influence it. A structural equation in a structural causal model defines the relationship between the effect (left-hand side) and its direct causes (right-hand side).

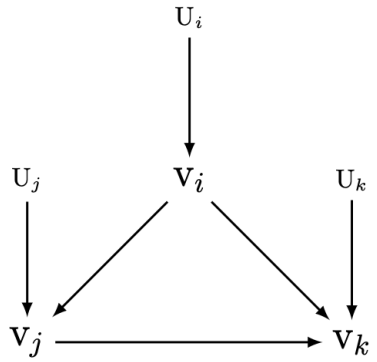
These models allow prediction in i.i.d. settings under changing distributions and are able to answer counterfactual questions (Peters et al., 2017).

2.2 Causal Graphical Models

A Graphical model represents a family of probability distributions over a set of random variables in terms of a directed or undirected graph (Jordan, 2004). These models harness the power of both probability theory and graph theory to offer a principled approach to dealing with uncertainty and complexity (Koller et al., 2007). In addition, they help reduce intricate systems with many interacting variables into smaller modules that are easier for humans to visualize and comprehend. A CGM consists of a graph that is both directed and acyclic, i.e. a directed acyclic graph (DAG). The former means that any edge between any two nodes is unidirectional (→ or ←). The latter means that no path starting at a node can lead back to it.

Definition 2.3. Directed Acyclic Graph (DAG). A DAG (Figure 2.1) is a graphical representation of variable relations within a system. Each node in the graph represents a system variable and each edge in the graph is unidirectional such that any edge from node x to node y means that x is a direct cause of y. Acyclicity means that the graph can not contain any cycles; a path starting at a node cannot lead back to it.

Figure 1



Note: A directed acyclic graph (DAG) over three variables. v_i is a direct cause of both v_j and v_k, v_j is a direct cause of v_k. Note that there are no cycles in this graph.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graphical model such that $\mathcal{V}$ represents a set of nodes or vertices in the graph and $\mathcal{E}$ is the set of directed edges connecting the members of $\mathcal{V}$. An edge $e_{i,j} \in \mathcal{E}$ from v_i to v_j such that $\{v_i \in \mathcal{V}, v_j \in \mathcal{V} \mid i \neq j\}$ implies that v_i is a direct cause of v_j respectively. The distribution is considered Markovian with respect to the graph, meaning that any conditional independencies the graph encodes can be verified by the joint probability distribution of the observations, which can thus be factorized as follows:

$$p(v_1, \ldots, v_n) = \prod_{i=1}^{i=n} p(v_i \mid \mathbf{Pa}(v_i)),$$

where n is the cardinality of the set $\mathcal{V}$. Equation 3 represents the factorized representation of the probability distribution entailed by the causal relations encoded in the graph. In other words, the joint probability distribution can be written as the product of the probability of each variable given its parents.

Definition 2.4. Conditional Independence. Two variables v_i and v_j are conditionally independent given v_k iff

$$P(v_i \mid v_j, v_k) = P(v_i \mid v_k)$$

Definition 2.5. Markov Condition. Given a DAG $\mathcal{G}$ and a joint distribution P over $\mathcal{V}$, then $\mathcal{G}$ is said to be **Markovian** w.r.t. $\mathcal{V}$ iff,

$$v_k \perp\!\!\!\perp_{\mathcal{G}} v_j \mid v_i \implies v_k \perp\!\!\!\perp_{P} v_j \mid v_i.$$

In other words, any conditional independencies found in the DAG must also be present in the observational distribution. In addition, the distribution is assumed to be faithful to the DAG such that any conditional independencies found in the data can be verified by the causal connections in the graph.

Definition 2.6. Faithfulness. Given a DAG $\mathcal{G}$ and a joint distribution P over $\mathcal{V}$, then P is said to be faithful to the DAG $\mathcal{G}$ iff the conditional independencies in the distribution are also encoded in the DAG,

$$v_k \perp\!\!\!\perp_{P} v_j \mid v_i \implies v_k \perp\!\!\!\perp_{\mathcal{G}} v_j \mid v_i.$$

With such causal models, we can answer both associational and interventional questions using the joint distribution and the graphical model.

Chapter 3.

3. Causal Discovery in i.i.d. Data

'Data are profoundly dumb' (Pearl et al., 2018). Data does not give any indication of the generating process. Any meaningful or overarching insights can only be drawn from a causal model. A large body of continuously-evolving research is dedicated to discovering causal models from data. Most causal discovery methods are implemented under the assumption that the data is *independent and identically distributed* (i.i.d) i.e. each observation is drawn independently from the same or different samples governed by the same probability distribution. Such a stringent assumption does not produce robust models deployed on real-world data. Data obtained from different environments will automatically nullify the validity of a model since they violate the above assumption. The most prolific approaches that will help either model from the previous section fall under two main categories:

- Constraint-based methods
- Score-based methods

Constraint-based methods leverage conditional independencies in the data to return a family of applicable graphs up to a certain equivalence class (see definition 3.1), whereas score-based methods leverage a goodness-of-fit approach by scoring learned parametric models.

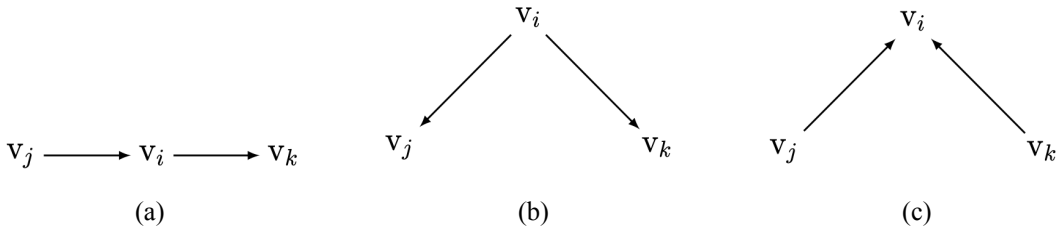
3.1 Constraint-Based Methods

These methods exploit the conditional independencies that are found in the data. They typically begin with node pairs (v_j, v_k) and test whether node subsets $\mathbf{v} = \{v_i \in \mathscr{V} \setminus \{v_j, v_k\}\}$ d-separate (v_j, v_k), where d-separation is defined below.

Definition 3.0. d-separation. Two nodes v_j and v_k are said to be d-separated or blocked by a node v_i iff v_i lies on a path between v_j and v_k, such that v_i, v_j and v_k form a chain or fork, as shown in Figure 3.1.

If such a set exists, then v_i and v_j are not adjacent in $\mathscr{G}$. Otherwise, they are connected by an edge. A skeleton graph with undirected edges manifests from these adjacency lists. The direction of each edge is then determined by looking for colliders in the graph. For instance, if (v_i, v_j) and (v_i, v_k) belong to the adjacency list then $v_j - v_i - v_k$ exists in the skeleton graph. If $v_i \notin \mathbf{v}$ then v_i is a node that d-connects (v_j, v_k) (Figure 3.1.c). By repeating this process, corresponding subgraphs that comprise $\mathscr{G}$ will emerge. These methods also assume that the distribution is *faithful* to the DAG. Three underlying subgraphs of interest are chains, forks and colliders.

Figure 2



Note: The graphical representation of (a) a chain (b) fork and (c) collider

In chains and forks, all node pairs $\{(v_i, v_j), (v_i, v_k), (v_i, v_k)\}$ are likely dependent (Pearl 2009a). Variable v_i is said to d-separate v_j and v_k, since upon learning v_i, v_j and v_k become independent. Therefore, both graphical representations, chains and forks, encode the same conditional independencies in the distribution P and are said to belong to the same equivalence class.

Definition 3.1. Equivalence Class. Two graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ are said to belong to the same equivalence class if all conditional independencies $p(v_i, v_j \mid v_k) = p(v_i \mid v_k)\, p(v_j \mid v_k)$ found in $\mathcal{G}$ are also found in $\mathcal{G}'$ $\forall v_i, v_j, v_k \in \mathcal{V}, \mathcal{V}'$.

Colliders play a slightly different role. Variables that are seemingly independent, since they do not share a common cause, nor do they directly or indirectly cause one another, become dependent in the presence of a known common effect (Pearl 2009a).

These methods suffer from their inability to scale as the space of candidate DAGs increases since they require conditional independency tests for all permutations of all node triples in the graph. Hence, the time complexity is exponential in the number of nodes. Several methods such as inductive causality (IC) (Pearl, 2009b), Peter-Clark (PC) algorithm (Spirtes et al., 2000), fast causal inference (FCI) algorithm (Colombo et al., 2012) and several others have been proposed for this task with the attempt to mitigate the scalability issue. In addition, the conditional independency tests can be unreliable as they depend on the type of data (discrete or continuous) and sample size.

3.2 Score Based Methods

These methods attempt to learn a set of causal graphs $\{\mathcal{G}_i\}$ by learning a set of parameters θ for structural equations $\mathbf{F}$. However, the model class for the structural equations is restricted first. For instance, in most cases, the equations are assumed to be linear, and the noise is additive and follows a standard normal distribution, i.e. $\sim \mathcal{N}(0,1)$.

Corresponding graphs are read from the equations and the goodness of fit is determined using a scoring function. The most popular scoring is the Bayesian Information Criterion (BIC), which takes into consideration the parameters $\hat{\theta}$ of the structural equations, the chosen graph $\mathcal{G}$, the data $\mathcal{D}$, the cardinality of $\mathcal{V}$ and sample size n, which is given by:

$$S(\mathcal{D}, \mathcal{G}) = logP(\mathcal{D} \mid \hat{\theta}, \mathcal{G}) - \frac{|\mathcal{V}| \log n}{2} \tag{7}$$

However, such methods suffer from several issues. For one, the number of potential graphs is exponential in the number of variables. Greedy methods have been introduced to mitigate this issue by narrowing down the search space. Once a candidate graph has been scored, neighbouring graphs are then scored to see if any have a higher score than the current candidate. If so, the search moves on to a higher-scoring graph, and in turn, its neighbours are tested. Second, choosing an appropriate scoring function can be difficult. Third, restricting the model to a specific class can produce unreliable or inconclusive results. For instance, variable relations may be non-linear or the noise terms might follow a non-Gaussian distribution.

Chapter 4.

4. Causal Discovery in Time Series Data

The approaches in the previous chapter apply to statically collected data. Unlike statically collected data, data collected at both regular or irregular time steps offers a unique ability to capture the dynamics of any one variable over time. Causal discovery in time series data, at least intuitively, seems more advantageous than causal discovery in static data since it offers a time ordering for observations (Peters et al. 2017). A large body of research has been dedicated to causal discovery in time series data from both linear and non-linear systems. Many approaches and variations thereof have been adopted to dismantle these relations. We divide these approaches into two broad categories: methods for stochastic systems and methods for deterministic systems. In contrast to stochastic systems, deterministic systems are not separable, meaning they can not be broken down into a sum of parts. Information from the driver system will surely be contained in the driven system. Although a prevalent amount of research has been and is still being dedicated to causal discovery in stochastic processes, there has been a growing interest and desire to disentangle causal relations in deterministic systems.

4.1 Causal Discovery in Stochastic Systems

4.1.1 Granger Causality

Granger (Granger 1969) used predictability, not correlation (Sugihara et al., 2012) to see if one variable is causal to another. Let $X = \{x_1, x_2, \ldots, x_t\}$ and $Y = \{y_1, y_2, \ldots, y_t\}$ be two time series that are realizations of stochastic processes X_t and Y_t , respectively, and

$I = (X, Y)$ account for all the *relevant* information in the universe up to and including time $t - 1$. Granger's method states that X is causal to Y if the inclusion of past values of X reduces the error in predicting y at time t:

$$\sigma(y_t | y_{t-\tau}, x_{t-\tau}) < \sigma(y_t | y_{t-\tau}), \tag{8}$$

where σ denotes the variance in the prediction of y_t. In the above equation, the variance in the prediction of y_t when past values of y are included is compared to the variance in the prediction of y_t when past values of both y and x are used. In the language of conditional independence, we check to see if Y is independent of X_{past} given Y_{past}:

$$Y \not\perp\!\!\!\perp X_{\text{past } t} | Y_{\text{past } t}$$

In other words, if past values of x up to a certain time lag τ $\{x_{t-k}\}_{k=1}^{k=\tau}$ improve the predictability of y_t by producing smaller variance σ, then X Granger causes Y. Granger argued that X must play a mechanistic role in generating Y otherwise, it would not improve its prediction (Shojaie et al. 2022).

Granger's Method

A common approach in time series analysis is first to find a suitable way to model the series. Linear vector autoregressive (AR) models are a popular choice:

$$AR(n) : x_t = \epsilon_{xt} + \sum_{i=1}^{i=n} a_i x_{t-i} \tag{9}$$

$$AR(m) : y_t = \epsilon_{yt} + \sum_{i=1}^{i=m} b_i y_{t-i} \tag{10}$$

where $\{a_i, b_j \in \mathbb{R} \,|\, n, m > 0,\ i \in [1,n],\ j \in [1,m]\}$ are autoregressive coefficients. n and m are the order of the autoregressive models indicating the number of past values required to obtain a reasonable prediction of x_t and y_t respectively. The choice of n and m

is not an easy task, but a simple approach is to begin at $n = m = 1$ and move incrementally to a max value τ' and see which value of τ produces the smallest prediction error. The error terms ϵ_{yt} and ϵ_{xt} are chosen as white noise Gaussian processes with mean $\mu_{\epsilon_{xt}} = 0$ and $\mu_{\epsilon_{yt}} = 0$. There is no correlation between the error terms and therefore $E[\epsilon_{xt}, \epsilon_{x(t-k)}] = 0$ and $E[\epsilon_{yt}, \epsilon_{y(t-k)}] = 0$.

Granger argues that if X is causal to Y then the inclusion of some past values of X will improve the prediction of Y. The new model for y_t becomes:

$$\hat{y}_t = \hat{\epsilon}_{yt} + \sum_{k=1}^{k=\tau} a_k y_{t-k} + b_k x_{t-k} \tag{11}$$

If X is not causal to Y then $b_k = 0 \ \ \forall k$ and thus $\hat{\epsilon}_{yt} \geq \epsilon_{yt}$. Granger assumes that there are no instantaneous effects i.e. k can not be 0. In addition, process X and process Y must be weakly stationary. This means that:

- The means μ_{xt} and μ_{yt} are independent of time i.e. $\forall t \ \ \mu_{xt} = \mu_x$ and $\mu_{yt} = \mu_y$
- The autocorrelation i.e. the relationship between two points in the series is independent of t and only depends on time lag τ : $E[x_t, x_{t-\tau}] = \gamma_x(\tau)$ and $E[y_t, y_{t-\tau}] = \gamma_y(\tau)$. You can think of autocorrelation as the rate of change of x and y with time t
- Linearity. Each data point can be written as a weighted linear combination of its predecessors up to a certain time lag (autoregressive model)
- The cause always proceeds the effect

Such assumptions will allow a range of mathematical and statistical tools to be used in the analysis (Granger et al., 2015). Stationarity is a very important assumption since we are stipulating that the process maintains a statistical equilibrium: if x_{t-k} causally effects y_t then x_{t+m-k} causally effects y_{t+m}. If the process is not stationary, several techniques, such as differencing, can be used to de-trend the process (Brockwell & Davis, 2009).

Granger's method and its variations are still one of the most prolific methods for causal discovery in stochastic systems. However, its application is not pragmatic in many real-world problems. Non-linear systems, latent variables, systems with confounding variables and those that are not separable are all excluded from its realm of applicability.

4.1.2 Transfer Entropy

Transfer Entropy (Schreiber 2000) is an information theoretical approach that is analogous to Granger's method. However, it is not restricted to linear systems with Gaussian errors. It is based on Claude Shannon's method of quantifying the amount of uncertainty contained in a random variable marginally or when conditioned on the observance of another variable (Shannon, 1948). Instead of comparing the variances, this method relies on comparing the amount of uncertainty in Y in the presence of X. It aims to quantify the information exchange between subsystems X and Y. This transfer of information is measured by computing the joint entropy of X and Y conditioned on any mutual information between both variables.

$H(X) = -\sum_{x} p(x) \log\ p(x)$ and $H(Y) = -\sum_{y} p(y) \log\ p(y)$ define the average amount of information contained in X and Y over all possible realizations of both X and Y respectively.

$H(Y \mid X) = H(X, Y) - H(Y)$ defines the average amount of information contained in Y once conditioned on X where $H(X, Y)$ is the joint entropy or amount of information contained in X and Y together:

$$H(X, Y) = -\sum_{x} \sum_{y} p(x, y) \log\ p(x, y).$$

In other words, the conditional entropy gives the amount of uncertainty left in Y in the presence of X.

Then, the amount of information transferred from X to Y up to a certain time lag τ is given by:

$$T_{X \to Y} = H(Y_t \mid Y_{t-\tau}) - H(Y_t \mid Y_{t-k}, X_{t-\tau}) \tag{12}$$

Intuitively, if there is a lot of information transfer from X to Y then the amount of uncertainty in Y is reduced in the presence of X. Thus the quantity in the minuend should be larger than that in the subtrahend.

4.2 Causal Discovery in Deterministic Systems

4.2.1 Convergent Cross Mappings

Many real-world systems suffer from inconsistent correlations or mirage correlations (Sugihara et al., 2012). A mirage correlation is a correlation that fluctuates in time between positive, negative or none at all. An old aphorism in statistics, "correlation does not imply causation," will always ring true; however, visualizations play a critical first step in data analysis, nonetheless, causal discovery. Sugihara argued that such contradictory findings are a hallmark of non-linear complex systems and any casual discovery method that relies, even partially, on correlations may not be reliable.
The underlying premise of Convergent Cross Mappings (CCM) is that the interactions within the system are governed by an underlying dynamical system. In other words, we are dealing with a deterministic continuous or discrete time system. A continuous time dynamical system can be defined by a set of ordinary differential equations that relate the rate of change of a variable to any system variables that influence it. For instance, in a system consisting of two variables x and y, the instantaneous rate of change of each variable can be written as a function of all influencing variables:

$$\begin{aligned} \frac{dx}{dt} &= f_x(x, y) \\ \frac{dy}{dt} &= f_y(x, y) \end{aligned} \tag{13}$$

Or, in the discrete case, an iterative map can be used to describe the state of the system at times t:

$$\begin{aligned} x(t) &= f_x(x(t-1), y(t-1)) \\ y(t) &= f_y(x(t-1), y(t-1)) \end{aligned} \tag{14}$$

Where f_x and f_y in both systems can be linear or non-linear functions.

If X is not causal to Y then $f_y(x, y) \equiv f_y(y)$ i.e. x does not play a role in determining the next value or the change in y.

CCM is grounded in Takens embedding theorem (Takens, 1981). Takens theorem states that if a system has an attractor, then the dynamics of the full phase space can be reconstructed from a single time series from the system (Strogatz, 1995). In other words, all information in the system could potentially be recovered from observations of a single variable, given that the system is coupled. Using just the observations of a single variable and an appropriate embedding dimension d and time delay τ, we can construct a shadow manifold M_x that is topologically equivalent to the original system manifold M. A manifold is a topological space that geometrically describes the state of the system. It consists of the set of all trajectories of the system. For instance, in a bivariate system, it consists of the set $\{(x(t), y(t)) \text{ as } t \to \infty\}$ for some initial state $(x(0), y(0))$. Topologically equivalent means there is a one to one mapping between the shadow and the true manifold. A single time series can be viewed as a projection of the manifold on to its coordinate axis thus allowing us to reverse engineer the manifold by using time delays

of the observed time series. As such properties of the original manifold can be derived from the properties of its reconstructed shadow.

Similarly, the same can be done with Y. Sugihara argued that if X is causal to Y, then M_y contains unique information about X that can only be relayed via Y. Neighbouring points in the shadow manifold M_y around time t must have a corresponding neighbourhood in M_x.

Let M_x and M_y be the shadow manifold of the true manifold M. Without any information about the system dynamics, we attempt to build a shadow manifold from time series X and time series Y:

$$\begin{aligned} M_x &= [x(t), x(t-\tau), x(t-2\tau), .., x(t-(m-1)\tau)] \\ M_y &= [y(t), y(t-\tau), y(t-2\tau), .., y(t-(m-1)\tau)] \end{aligned} \tag{15}$$

Four different plausible causal relations can exist within a simple bivariate system: $X \rightarrow Y, X \leftarrow Y, X \longleftrightarrow Y, X \quad Y$. As a result, there are four different sets of governing equations assuming the system's evolution is in continuous time (Figure. 2).
If a variable is causal to another, then the variable appears in the differential equation of that variable (Sugihara et al., 2012; Rubenstein et al., 2016; Bellot et al., 2021).

Figure 3

$$\begin{cases} \frac{dx}{dt} = f_x(x,y) \\ \frac{dy}{dt} = f_y(x,y) \end{cases} \qquad \begin{cases} \frac{dx}{dt} = f_x(x) \\ \frac{dy}{dt} = f_y(x,y) \end{cases} \qquad \begin{cases} \frac{dx}{dt} = f_x(x,y) \\ \frac{dy}{dt} = f_y(y) \end{cases} \qquad \begin{cases} \frac{dx}{dt} = f_x(x) \\ \frac{dy}{dt} = f_y(y) \end{cases}$$

(a) (b) (c) (d)

Note: (a) If we have feedback between X and Y then X is causal to Y and Y is causal to X. This is a coupled system. (b) X is causal to Y but not vice versa. (c) Y is causal to X but not vice versa. (d) No causal relation between X and Y.

Longer time series will provide denser neighbourhoods and, thus, eventual convergence. If X is causal to Y then there is a 1-1 mapping from every neighbourhood in M_y to a neighbourhood in M_x. The converse is not true unless Y is also causal to X. If Y is not causal to X then the mapping is from M_x to M_y would be more sparse. The stronger the coupling in either direction, the denser the neighbourhood will be (Harnack et al., 2017).

CCM works best with weakly coupled dynamical systems. In strongly coupled systems, it is hard to distinguish between cause and effect.

4.2.2 Causal Kinetic Models

Much of the longitudinal data that is collected from different environments in various disciplines tends to belong to an underlying kinetic system. Learning the structure of such systems is important in order to conduct various inference tasks (Pfister et al., 2019). Some of the most prolific approaches include learning the underlying ODEs in a data-driven way when feasible and tractable. However, in many situations, this may not be possible when the system is complex or high dimensional, rendering the space and time complexity high. Given a d-dimensional system $\mathbf{X} = \{X_1, X_2, \ldots, X_d\}$, Pfister et al. propose a new framework CausalKinetiX, that learns a set of models $\mathbf{M}_i$ for each variable $X_i \mid i \in [1,d]$ by sourcing the data from different environments (Pfister et al., 2019). Simulation of different environments is done by taking a known system and intervening on it by either changing the initial conditions or fixing a variable to a certain value. In real world systems without a known underlying model, this can be done by sampling data from different environments having disparate data distributions. Then, a stable model that lies at the intersection of all generated models is chosen. A stable model is one that contains covariates that persist across models in $\mathbf{M}_i$ and maintain reasonable predictive power. Also, by learning a model for one variable at a time, they reduce the

computational complexity significantly. These models strictly deal with discrete time dynamical systems.

Causal Kinetic Models (Peters et al., 2022) were proposed to address the need for causal discovery techniques in continuous time dynamical systems. These models are closely related and can be thought of as an extension of Structural Causal Models. Since interventions are a key method for inferring causality, Peters et al. lay the road map for the type of interventions that can be performed in such systems, such as intervening on the initial value, fixing the rate of change or setting a variable to a specific value. They also account for any stochasticity in such systems in the form of either additive or driving noise. In terms of interpretability, systems defined by a set of ODEs are the gold standard (Schölkopf, 2019; Schölkopf et al., 2021). They offer the granularity that is necessary to understand not only the evolution of each variable but also the invariant mechanisms that generate the phase space. Consider a multivariate d-dimensional process $X_t = \{x_t^1, x_t^2, \ldots, x_t^d\}$ governed by a set of ordinary differential equations:

$$\begin{aligned} \frac{dx_t^1}{dt} &= f_{x^1}(Pa(x_t^1)) \\ &\vdots \\ \frac{dx_t^d}{dt} &= f_{x^d}(Pa(x_t^d)) \end{aligned} \tag{17}$$

With corresponding initial values $\{x^1(0) = \eta^1, x^2(0) = \eta^2, \ldots, x^d(0) = \eta^d\}$ where $\frac{dx_t^d}{dt}$ is the time derivative of x^d at time t. Unlike structural causal models, x^d can be in the set of parents $Pa(x^d)$ of x^d. The existence and uniqueness of a solution of the initial value problem is assumed and required. Since the initial values are not known and therefore may be chosen at random, this induces a probability distribution over the X. As mentioned before, noise can be additive:

$$\hat{x}_t = x_t + \epsilon_t \tag{18}$$

Where $\hat{x}_t$ is the observed value of x at time t and x_t is the true value. Or the noise can be a system driver:

$$\begin{aligned} \frac{dx_t^1}{dt} &= f_{x^1}(Pa(x_t^1)) + \epsilon_t^1 \\ &\vdots \\ \frac{dx_t^d}{dt} &= f_{x^d}(Pa(x_t^d)) + \epsilon_t^d \end{aligned} \tag{19}$$

However, such systems are considered to be stochastic.

Chapter 5.

5. Learning Analytics in Computer Science Education

In Chapters 2, 3 and 4, we discussed the benefits of establishing a causal model in contrast to a statistical or machine learning one and how these models can be built using various causal discovery techniques. In this Chapter, we discuss the area in which we apply these techniques and the reasoning behind them.

With the advent of online learning platforms and the prevalent use of learning management systems, the accumulation of learner data has become commonplace. The collection and analysis of such data for the purpose of improving student learning and experience is referred to as learning analytics (Baker et al., 2014). Such data can be analyzed to understand learner behaviour and gain insight into the intricacy of the learning process in order to offer learners tailored recommendations and *interventions* which will ultimately improve learning (Hundhausen et al., 2017).

Computer science (CS) is a fairly new science in comparison to other sciences. There are not any true and tested ways for best practices when it comes to delivery. For instance, in mathematics, the mantra ‘drill and kill’ has been a long-standing approach to cementing certain concepts and maximizing knowledge retention. However, in CS, students usually begin their post-secondary journey with little pre-existing knowledge about computer languages, algorithm writing and, not to mention, best practices. As a result, the learning curve is steep and students are left with the difficult task of balancing between understanding theoretical components and learning the lexical structure of a new programming language and proper application.

To offset the material overload, CS students spend a significant portion of the term in the laboratory working on different assignments that are meant to bridge the gap between theory and application. This gives students an opportunity to apply what they are learning in creative ways by using a combination of problem-solving skills and the available language library and constructs. The solution space can grow exponentially since there are multiple pathways to a functional program. However, a functioning program does not necessitate robustness or optimality as a precursor nor does it guarantee either as an outcome. Once a student is able to produce the expected output little attention is paid to the quality of the written code (Keuning et al., 2017). Students spend a considerable amount of time trying to produce syntactically and semantically correct programs while paying little attention to best practices (Dietz et al., 2018). Such practices are necessary in industry where projects are more complex and multifaceted and the outcome should possess attributes such as reasonable complexity, maintainability, extendibility and security. These characteristics are the hallmark of quality-code and are all standard industry goals during the software development process. The ideal place to learn and apply such practices is in the laboratory either from the instructor or from the development environment. However, it is unfeasible for an instructor to provide such feedback in a timely fashion as the number of students increases and the solution space grows. Feedback *might* come after grading, however this can be problematic since it may not come in time before the next assignment and even then the instructor may be oblivious to all flaws that are present due to the intricate nature of a given program. The alternative would be to exploit the development environment for instant feedback so that students ascertain these best practices as quickly as possible and begin to use them effectively.

5.1 Integrated Development Environments

In CS education, there is growing interest in understanding learner behaviour, improving success rates and producing industry-ready programmers. Since students use

an integrated development environment (IDE) for their programming tasks and consequently spend a lot of time in the development environment, a rich body of learner data can be inconspicuously collected and leveraged to understand the coding process. Therefore, for CS students, there is no better place than the IDE to collect leaner data (Hundhausen et al., 2017). An IDE is a software application with an extensive amount of user-support and features where programmers can write, debug and test their code. They are intended to simplify the development process and offer programmers real-time feedback. In computer programming courses, students spend a copious amount of time in the IDE working on different applications and programming assignments. As such, much of the learning takes place in the IDE as it provides the means to apply what is learned in the classroom, making it an ideal candidate to monitor student activity and progress at a granular level. This means that, at relevant time steps, important attributes such as the number of lines of code, compilation errors and code deficiencies can be recorded. The dynamic collection of such data offers a unique insight into student practices and sheds light on the learning process as a whole.

While the main reason to use an IDE as opposed to a command line interface (CLI) is to streamline the development process, IDEs can also be extended through a standalone independent plug-in component. For instance, a broad range of information might be of interest, so a snapshot of the code alongside other features can be uploaded automatically via a version control system such as GitHub at regular or irregular time intervals. The data can be analyzed, and the text can be parsed to build a learning model. Then, the model can be used to create tailored features for coordinated *interventions* during the coding process. Consequently, IDEs can be instrumented to help students develop the necessary competency skills in order to ensure seamless migration into industry.

5.2 Assessing Student Competency

One way to assess a student's competency is to examine the quality of the source code the student writes. Quality code is imperative in both academia and industry. As a computer program's objectives grow and program length increases, the issues with low-quality code become more apparent as it reduces the ability to scale a project and maintain it (Dietz et al., 2018). Its relevance in industry is considerably higher since budget and understandability among teams of developers are at stake (Östlund et al., 2023; Kirk et al., 2020). Source code metrics such as number of lines of code and other characteristics such as readability, maintainability, extendability and security are often used to measure and assess the quality of code. A visual inspection of the code by a human examiner doesn't always uncover any violations of these criteria due to the intricacy and length of a given program. Consequently ed, IDEs are often instrumented to uncover any ineptness in the code in the form of competency violations. Not only can the IDE point to the issue, it can also provide students with consistent, uniform and timely feedback on the issue. As a result, a student can learn about an issue, why it is occurring and how it can be fixed.

Chapter 6.

6. A Coding Process

In order to effectively study and apply causal discovery techniques to data collected from an IDE, a set of mathematical definitions must be carefully curated. This will allow the construction of robust models that can be studied from a theoretical standpoint and facilitate inference tasks.

Assumptions

Data is committed (saved) to a version control system (VCS) at different timestamps. Several variables of interest can be the target of each timestamp, such as lines of code (LOC), number of issues and type of issue. In addition, another timeline can simultaneously exist where each timestamp registers whether the student consulted the system for feedback or not. Other consults, such as interactions with a lab instructor and the duration of that interaction, can be manually committed.

Definitions

Definition 6.0. Commit Timeline. Let $T' = \{t'_i\}_{i=1}^{i=n}$ be a sequence of times where $t'_i < t'_j$ for $i < j$. Furthermore, t'_1 represents the first timestamp at which the code was committed to the VCS and t'_n is the last timestamp the code was committed to the VCS.

Definition 6.1. System-Consult Timeline. Let $T'' = \{t''_i\}_{i=1}^{i=m}$ be a sequence of times where $t''_i < t''_j$ for $i < j$. Furthermore, t''_1 represents the first timestamp at which the

student consulted the system for feedback and t''_m is the last timestamp at which the student consulted the system for feedback.

Definition 6.2. System Timeline. This is an amalgamated timeline of the commit timeline and the system-consult timeline. Let T represent the system timeline such that $T = \{T' \cup T''\}$ where $T' \cup T'' = \{t \mid t \in T' \ \vee \ t \in T''\}$.

Definition 6.3. System Variable. A system variable is any variable that is tracked and registered by the IDE during the course of writing a computer program.

Definition 6.4. Coding Process. The coding process is the process of working on a programming task from time $t_1 = \min(T)$ to time $t_k = \max(T)$. Then, the *lifetime* of a coding process is the time interval $[t_1, t_k]$. The coding process entails one or more stochastic processes.

Let $\mathscr{P} = (\mathbf{X}, I, S)$ be a coding process defined by a multivariate stochastic process $\mathbf{X}$ and a set of interventions I that are indexed by time and can be *performed* on the coding process over the course of the coding lifetime and a set of states S, also indexed by times that are *reached* by the system over the lifetime of the coding process.
Let $\mathbf{X} = \{X_t^1, X_t^2, \ldots, X_t^d\}$ such that X_t^i for $i \in \{1,2,..,d\}$ is a collection of random variables pertaining to a single system variable indexed by time t. Each X_t^i process is representative of a variable of interest during the coding process. For instance, variables such as LOC, number of issues in code, human consults and system consults each represent a stochastic process belonging to $\mathbf{X}$.
Let $I = \{\mathrm{I}_t\}_{t=t_1}^{t=t_m}$ be the set of all interventions done on the system such that
$I_t = (I_t^1, I_t^2, \ldots, I_t^d)$ where $I_t^i \in \{x_t^i, \eta_t^i\}$ and η_i^t is a constant value that x_t^i is set to and I_t is the vector of assignments that are carried out at time t. In other words, at time t, one or more assignments $x_t^i = \eta_t^i$ are done on one or more system variables. All other variables

at time t remain the same. For instance, if X_t^3 represents consults and at time $t = 5$ the student is forced to consult (a tutor or the system), then the assignment $I_5^3 = \eta_5^3$ is carried out where $\eta_5^3 = 1$ and the remaining system variables at time $t = 5$ remain untouched rendering $\mathrm{I}_5 = (x_5^1, x_5^2, 1, ..., x_5^d)$. Here 1 means the student consulted at t and 0 otherwise. Then, I_t can be characterized as a perturbation of the system either in the form of help from a human tutor or a system consult where the student engages with either the human tutor or the system to understand what the issue is and how to fix it.

S is a set of states that the system passes through during the coding process. $S \in \mathbb{C}^{k \times d}$ where k is the number of states the system reaches over the coding lifetime and d is the number of system variables, then:

$$S = \begin{bmatrix} I_{t_1}^1 & I_{t_1}^2 & \dots & I_{t_1}^d \\ I_{t_2}^1 & I_{t_2}^2 & \dots & I_{t_2}^d \\ \vdots & \vdots & \ddots & \vdots \\ I_{t_k}^1 & I_{t_k}^2 & \dots & I_{t_k}^d \end{bmatrix}.$$

The rows of matrix S represent the different states of the system over the coding lifetime such that $S_t = (I_t^1, I_t^2, \dots, I_t^d)$ represents the state of the system at time t.

Note that if there are no forced interventions, then:

$$S = \begin{bmatrix} x_{t_1}^1 & x_{t_1}^2 & \dots & x_{t_1}^d \\ x_{t_2}^1 & x_{t_2}^2 & \dots & x_{t_2}^d \\ \vdots & \vdots & \ddots & \vdots \\ x_{t_k}^1 & x_{t_k}^2 & \dots & x_{t_k}^d \end{bmatrix}.$$

This means that S is a pure manifestation of the observational data.

At time $t = t_1$, all system variables are set to 0 and $t = t_k$ is the last recorded timestamp of each series.

Specific to the analysis conducted in Chapter 7, three processes were noted throughout: LOC, issues and consults. This, in turn, is a 3-dimensional system ($d = 3$).

Definition 6.5. LOC Time Series. If the stochastic process X_t^1 defines lines of code (LOC), then the *LOC time series* is a realization of this stochastic process. Let $\{x_t^1\}_{t=t_1}^{t=t_n}$ be a collection of random variables over time t where $t \in T$ and x_t^1 is the number of lines of code in the program at time t.

Definition 6.6. Issues Time Series. If the stochastic process X_t^2 defines the number of issues in the code, then the *issues time series* is a realization of this stochastic process. Let $\{x_t^2\}_{t=t_1}^{t=t_n}$ be a collection of random variables over time t where $t \in T'$ and x_t^2 is the number of issues in the code at time t.

Definition 6.7. Consult Time Series. If the stochastic process X_t^3 defines student consults, then the *consult time series* is a realization of this stochastic process. Let $\{x_t^3\}_{t=t_1}^{t=t_m}$ be a collection of random variables over time where $t \in T''$ and $x_t^3 \in \{0,1\}$ where 1 indicates that the system was consulted at time t for feedback and 0 otherwise. It is worth mentioning that, x_t^3 can also be registered as a duration as opposed to a binary value. This duration is the time in seconds the students spends interacting with the system to understand and rectify an issue.

Chapter 7.

7. Application of Causal Discovery Methods in IDE Data

In this chapter, we apply an appropriate causal discovery method on data collected from an IDE. In section 7.1 we discuss the data and its source. In section 7.2 we present our research question and in section 7.3-7.6 we discuss the applicable methods used and the results obtained from applying each method. We leave a detailed discussion for Chapter 8 and a list of recommendations for Chapter 9.

7.1 Data Source

The data was collected in a study done at a Canadian University in 2017 (Boulanger et al., 2017). The study was conducted as a randomized controlled trial where students were randomly selected from a Java programming course to participate. A total of 110 participants accepted the invite and were randomly assigned to the control or experimental group. All students were given three programming tasks that would take approximately five hours to complete. However, there was no restriction on when the tasks should be finalized. Consequently, tasks were completed in a range of $(4 - 141)$ days for the control group and $(7 - 190)$ days for the experimental group. The coding activities for both groups were tracked in the IDE through a series of commits, either manual (by the user) or automatic (by the system), to a version control system (VCS). A commit is the act of saving the current version of the code plus any other features to a VCS. However, the origin of the commit and when exactly it would happen were not clear from the data. With each commit, various variables of interest were registered (Table 7.1). This, in turn, generated a timeline for each student where, at each timestamp, several variables of interest were saved to the VCS (see definition 6.0). In addition, students were given a short quiz prior to running the experiment to determine their

respective coding levels and, as a result, assigned an appropriate task based on skill level. Students were also asked to report their ‘perceived’ coding level. We call the former ‘task level’ and the latter ‘reported level.’

For the experimental group, the IDE was extended with a plugin that would offer students real-time feedback on issues in their code. This, in turn, generated a new timeline where each timestamp represented when a student consulted the system for feedback (see definition 6.1). In total, 12 competence criteria were measured in the code (Table 7.2) (Boulanger et al., 2017). Each measurement was determined by a set of (10-30) rules for each competence measure. If the student was in violation of any of those rules in her code, the system would alert her of the competence measure and the rule within that is violated. Through a simple click, a student can see the competence measure that was violated, read about it and learn how to fix it.

Table 1

Name and Description of variables that were tracked in the study

Variable Name	Description	Source	Group
Coding (Commit) Timestamp	Unix timestamp at which the code was captured	VCS	Experimental/ Control
LOC	Lines of code at a given timestamp	VCS	Experimental/ Control
Issues	Number of issues at a given timestamp	VCS	Experimental/ Control
*Consult Timestamp	Unix timestamp at which a student engages the system for feedback	VCS	Experimental
Consult Category	Type of consult	VCS	Experimental
Consult Action	Type of issue found in the code	VCS	Experimental
Task Level	Determined task level after the student took a short quiz	Quiz	Experimental/ Control
Reported Level	A students self-perceived coding ability	Survey	Experimental/ Control

Table 2

Competence criteria that were used to assess the code

Competence Goal	Explanation
Clarity	Code that is easy to read and understand in terms of syntax; avoids use of ambiguous functions or confusing function/class names.
Cleanliness	Code that contains only those lines of code needed to do an assigned task (e.g. no unused variables, methods, and classes; no commented blocks or lines of code).
Conciseness	Code that avoids verbosity and redundancy. A program is considered concise that accomplishes as much as possible in the fewest lines of code. Examples of non-concise code include duplicate lines of code, calls to different functions that basically do the same action, redundant casts, etc.
Efficient Multi-Threading	Code that properly uses multi-threading techniques such as threads, run methods, locks, synchronization, etc.
Follows Standards	Code that uses proper formatting, white-space, naming conventions for variables, methods, and classes, conventional method signatures, etc.
Good Practices	Code that adheres to a set of Java-specific “best coding practices” which dictate how certain Java classes and methods should or should not be used and in what context. These include: method overriding, use of threads, null checking, method return types and how to use them, etc.
Performance	Code that only uses those methods, classes and concepts that are absolutely needed to solve a problem (e.g. uses proper types to store different types of data, has methods returning only required data, etc.).
Proper Error-Handling	Code that uses proper Java constructs to handle errors that are or may have been thrown by the program. Relates to try/catch/finally, exceptions, etc.
Proper Java8 Concept Use	Code that makes proper use of Java8-specific features like lambdas, anonymous classes, try-with-resources, the Optional class, etc.

Reliability	Code that avoids possible pitfalls or bugs in the code, meaning items which may work now but risk causing issues down the road due to inherent flaw in using them in a certain manner (e.g. use of raw types with collections, use of variant “for” loop stop conditions, instantiating StringBuffer/StringBuilder with single character, etc.).
Simplicity	Code that is simple to read and understand and is computationally efficient (i.e. time complexity and the Big(O) notation). NOTE: The lower the complexity, the better. Examples of higher complexity include complex Boolean expressions, nested loops, many lines of code in switch statements, too many switch statements, etc.
Well Designed	Code that is properly designed conceptually, which includes proper use of return types, inheritance, proper use of constructors, proper use of the static keyword, etc.

At the end of the study, a total of $N = 29$ usable datasets remained, 16 of which were from the control group and 13 from the experimental group.

7.2 Research Question

Our goal was to determine whether there was a direct causal link between system consults and the number of issues present in the code once the student finished the coding task. Did we see competence growth in the experimental group over time? By competence growth, we mean, did we see a reduction in issues defined in Table 7.2 at the end of the coding process?

7.3 Methodology

Due to the nature of the data, the problem was approached in two different ways: statically and dynamically. For the former, the data was considered as a whole, where a single value for each student was calculated for each of the recorded variables. This approach utilizes both the experimental approach (RCT) to causal discovery and Pearl’s

inductive causation approach, such as the constraint-based methods mentioned in Chapter 3. For the dynamic approach, each student's timeline was used, resulting in several time series', three for each student in the experimental group: lines of code (LOC), issues and consults. This approach utilizes the definitions introduced in Chapter 6 and the methods mentioned in Chapter 4. Methods from the stochastic case, such as Granger's method and transfer entropy, were used, in addition to convergent cross mappings that are found under the deterministic methods.

7.3.1 Data Preprocessing

The registered variables served as a good starting point to gain insight into the coding process and understand its evolution. Each registered variable was, in turn, analyzed and used either directly or indirectly as part of an engineered new feature. Since the research questions were approached statically and dynamically, each methodology necessitated a different set of features. Table 7.3 provides a list of all pertinent features for each approach and the groups considered.

Table 3

List of applicable variables for each methodology

Method	Group	Feature	Feature Description	Feature Type
Randomized Controlled Trial	Control/ Experimental	Number of Issues (NI)	Number of issues present at the end of the coding timeline	Raw
		Issues per line (IPL)	The number of issues at the end of the coding timeline relative to the number of lines of code at the end of the coding timeline	Engineered
Inductive Causation	Experimental	Number of Issues (NI)	Number of issues present at the end of the coding timeline	Raw
		Lines per issue (LPI)	The number of lines at the end of the coding timeline relative to the number of issues at the end of the coding timeline.	Engineered

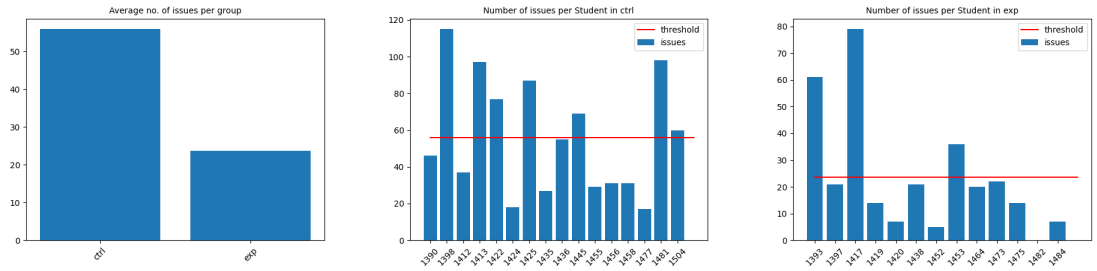
		Consult (C)	The type of consult	Raw
		Number of consults (NC)	Aggregate value which is the total number of times the student consulted the system over the course of the coding timeline. Each time the student looked at an issue, read a hint or looked at a regulation card the system timestamped that specific action. Each timestamped action incremented nConsults.	Engineered
		Number of consult episodes (NCE)	Consecutive consults addressing the same competency are amalgamated into one consult	Engineered
		ImprovementScore (IS)	This is measured by taking the ratio of the length of the unique unique list (one with no duplicate rules) divided by the length of the entire consult list. Eg. consult_lst = [Rule1, Rule5, Rule7, Rule4, Rule6, Rule5, Rule1,Rule4] and consult_lst_unique= [Rule1, Rule5, Rule7, Rule4, Rule6]. Therefore, the improvementScore = 5/8 = 0.6. Scores closer to 1 are good - meaning the student is learning from past mistakes (i.e. we do not see a lot of repetition in the competence rules)	Engineered
Granger, Transfer Entropy, Convergent Cross Mappings	Experimental	t'	Time stamp at which the code was saved or committed to the VCS	Raw
		t''	Time stamp at which a consultation took place	
		x_t^1	Lines of code (LOC) sat time t	Raw
		x_t^2	Number of issues at time t	Raw
		x_t^3	0 or 1 where 1 indicates that the system was consulted at time t and 0	Raw

7.4 Randomized Controlled Trial

Since the data was collected from a randomized trial and such trials are the gold standard for causal inference, we aimed to establish causality by first comparing the

means of both groups and then calculating the average causal effect (Figure 7.1). By randomization, we eliminate selection bias and any latent covariates since the groups will be statistically equivalent. Table 7.4 shows that the groups are relatively similar. Task level is synonymous to programming experience. In each group, the distribution between groups based on programming experience was relatively the same.

Figure 4



(a) (b) (c)

Note: (a) Difference in means (issues) between control and experimental group. (b) The number of issues at the end of the coding timeline for the control group. (c) The number of issues at the end of the coding timeline for the experimental group. The red line represents the average number of issues in that group (threshold).

Table 4

Task Distribution

Task Level	Group	
	Control	Experimental
Beginner	9	8
Intermediate	6	4
Advanced	1	1

Note: Task distribution after randomly assigning students to the control and experimental group. The task level was determined after each student completed a short quiz.

7.4.1 Difference In Means (Direct)

In Figure 7.1(a) we can see that the average number of issues (NI) at the end of the timeline was significantly higher for the control group than the experimental group. This is a strong indicator that the experimental group's exposure to system consults has led to this difference. However, to verify that this difference is not due to chance but due to the treatment (system consults) that the experimental group received, a t-test was conducted. Let $\mathscr{H}_0$ be the null hypothesis and and $\mathscr{H}_a$ be the alternate hypothesis:

$\mathscr{H}_0$: difference in means in NI is due to chance

$\mathscr{H}_a$: difference in means in NI is due to a fundamental difference between groups

In comparing the NI between both groups, the average number of issues in the control group ($\mu = 55.875$, $\sigma^2 = 983.98$) was significantly higher than that of the experimental group ($\mu = 23.62$, $\sigma^2 = 524.09$) with an average difference of 30 more issues ($t(27) = 3.197, p = 0.0035$). Using a significance value of $\alpha = 0.05$, we can reject the null hypothesis $\mathscr{H}_0$ since $p = 0.0035 < \alpha = 0.05$ and confidently state that the difference between the control and experimental group is not due to chance and that consulting the system does in fact reduce the number of issues at the end of the coding timeline.

7.4.2 Difference in Means (Indirect)

As the LOC increases, a larger number of issues will naturally manifest in the code. Consequently, we wanted to consider the number of lines in the code when determining

the number of issues at the end of the timeline. This resulted in a normalized metric which is the average number of issues per line (IPL) for the group. Again, in comparing the average IPL, the control group had a larger mean but similar standard deviation ($\mu = 0.11, \sigma^2 = 0.0023$) to the experimental group ($\mu = 0.065, \sigma^2 = 0.0065$) with an average of 0.045 issues more per line ($t(27) = 2.44, p = 0.021$). In other words, students in the experimental group on average had to write 6 more lines of code than their counterparts in the control group to see an issue. Again, using a significance value of $\alpha = 0.05$, we can reject the null hypothesis $\mathscr{H}_0$ since $p = 0.021 < \alpha = 0.05$ and confidently state that the difference between the control and experimental group is not due to chance and that consulting the system does in fact reduce the number of issues at the end of the coding timeline even when scaled by the number of lines.

Summary of Results

Students in the experimental group saw, on average, fewer issues than those in the control group (Table 7.5). The average causal effect, which is the difference between the outcome of the control and experimental group, is positive in both mean calculations. A positive result indicates that the number of issues (raw or normalized) was greater in the control group than in the experimental group. This indicates that there was a positive effect from consulting the system since the average number of issues in both cases was less for the experimental than the control group.

Table 5

	Control Group	Experimental Group	Average Causal Effect	Remarks
Mean (NI)	55.875	23.62	32.255	On average, students in the control group saw 32 more issues in their code at the end of the coding timeline.
Mean (IPL)	0.11	0.065	0.045	On average, students in the experimental group had 1 issues per 15 lines of code where students in the control group had on average of 1 issue per 9 lines of code.

Note: Results from the randomized controlled trial

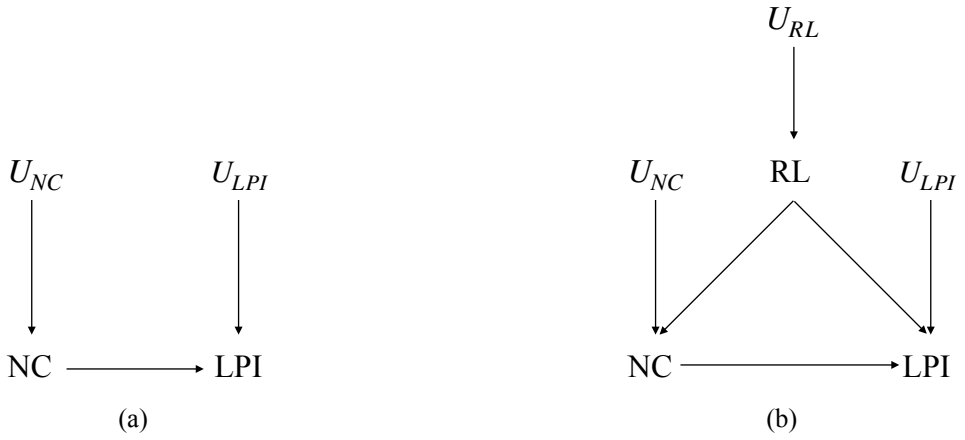
7.5 Causal Model Search (Static Case)

In this section, we begin our casual model search by considering only the experimental group. Among those who have access to system consults, we aimed to find evidence that an increase in system consults resulted in fewer issues at the end of the coding timeline while considering other variables in the system. This may appear to be a rudimentary question. However, we wanted to investigate whether students were passively consulting the system and not deeply contemplating the reason and the resolution to the issue they encountered.

Any causal model search begins with either a *causal claim* or a *causal discovery* algorithm. Causal claims are based on expert or prior knowledge about variable relations in the system. These claims manifest themselves into a pool of potential DAGs whose validity can be verified using conditional independence testing or any other suitable method. If a DAG is not Markovian, i.e. not supported by the observational data, then it is discarded from the pool of DAGs.

If a causal claim can not be made, an appropriate causal discovery method is chosen, taking into consideration the nature of the available observations.

Figure 5



Note: (a) Potential DAG with no confounding variables. U_{NC} and U_{LPI} are the unobserved (exogenous) variables that are influencing NC and LPI respectively. U_{NC} and U_{LPI} are independent i.e. $U_{NC} \perp\!\!\!\perp U_{LPI}$ (b) Potential DAG where RL is a common cause.

Causal Model (Prior Knowledge)

To initiate our causal model search, we began isolating variables of interest. Table 7.3 shows all system variables, some of which were obtained directly while others were engineered. If an engineered variable were utilized, it would replace any variable(s) involved in its construction in the final model. For instance, if LPI were considered, its use would eliminate the number of issues (NI) and lines of code (LOC) from the final model.

Since programming tasks were assigned based on skill level, program length differed based on the task. As a programming task increases in length and becomes more

intricate, a student will likely encounter more issues. Therefore, lines per issue (LPI) were utilized instead of number of issues (NI) in order to mitigate the increase in issues introduced by longer programming tasks. Following this process, three variables of interest remained: lines per issue (LPI), number of consults (NC) and reported level (RL). Two potential graphs were identified, given the provided data and prior knowledge (Figure 7.2). The first DAG (Figure 7.2a) is a simple two-node model where we assume the presence of exogenous variables acting on each node but were not measured in the study. For instance, variables such as motivation and discipline can influence NC but they are unobserved. The second DAG (Figure 7.2b) is a three-node model that is an extended version of the two-node model. Here, we assume the presence of a common cause RL. The reported level can be a potential common cause since it can convey the student's level of confidence and ability, thus affecting the number of times she consults and the number of issues she encounters.

7.5.1 Additive Noise Models (Two-Node DAG)

Calculating the conditional probabilities may appear to be a good starting point. However, data is passive, and conditional probabilities merely tell us what the probability of LPI is for a subset of the data, i.e., for a specific NC value. In the language of do-calculus (Pearl, 1995), we need to find the conditional probability of LPI when NC is manipulated, i.e. $P(LPI \mid do(NC))$. If $P(LPI \mid NC) = P(LPI \mid do(NC))$ then NC has no effect on LPI. For the two-node model, this would be trivial. An alternative approach for a bivariate system is additive noise models (ANM) (Hoyer et al., 2017). These models assume that the effect can be written as a function of the cause and an added noise term. These models fall under both constraint-based and score-based methods. They exploit both independence testing and function fitting. Two potential models emerge in our bivariate case:

$\text{LPI} = f(\text{NC}) + N_{LPI} \quad NC \perp\!\!\!\perp N_{LPI} \qquad \text{NC} = g(\text{LPI}) + N_{NC} \quad LPI \perp\!\!\!\perp N_{NC}$
$\text{NC} = N_{NC} \qquad \text{LPI} = N_{LPI}$

Model (a) Model (b)

The functions f and g can be linear or non-linear and the noise term in each model must be independent of the regressor. In model (a) LPI is regressed on NC and in model (b) NC is regressed on LPI. In the true model, the error term from the regression equation must be independent of the regressor. In other words, if model (a) was the correct model, then N_{LPI} must be independent of NC. Similarly, if model (b) was the correct model then N_{NC} must be independent of LPI. An independence test is done between these terms in each model and the one with lower degree of independence is rejected. The python package CDT (Kalainathan et al., 2019) was used to fit the data to an ANM. The algorithm returns a score on the interval $[-1,1]$ where scores on $(0,1]$ indicate that NC causes LPI and otherwise.

Summary of Results

If the data is truly generated by an additive noise model, then the true causal direction can be recovered from the data. An independence test is conducted between the regressor and the noise term in each model, yielding two scores (test statistics). The algorithm returns the difference between both scores (Table 7.6).

Table 6

ANM results

Model	Causal Direction	Independence Score	Results
Model a	$NC \rightarrow LPI$	0.423	0.1
Model b	$LPI \rightarrow NC$	0.342	-0.1

A positive result indicates the true causal direction. Therefore, system consults (NC) does have a causal effect on lines per issues (LPI) at the end of the coding timeline. However,

since results closer to 1 are stronger indicators of causality, it would be pragmatic to consider these results inconclusive. This could be due to several factors. For one, 4 out of the 13 students did not consult the system for feedback, grossly affecting the regression results. Other factors, such as the use of an inappropriate model, a small dataset or feedback between the variables, i.e. a bidirectional edge between NC and LPI could also affect model performance.

7.5.2 Inductive Causation (Three-Node DAG)

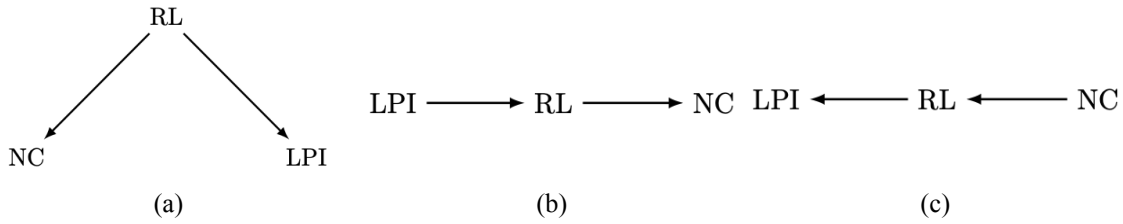
For the three-node DAG (Figure 7.2b), some variables were first transformed into categorical (Figure 7.4). With categorical data, calculating the average causal effect became simple since any assumptions or predictions about the underlying distribution of the variables was not required. If students consulted the system more, then NC is greater than the threshold and, therefore, has a categorical value of 1 and 0 otherwise. If students wrote a substantial amount of code before encountering any issues then LPI $=$ 1 and 0 otherwise. The RL variable was already a categorical variable, so no transformation was required.

To verify our causal claim, we performed the following conditional independence test:

$$P(\mathrm{NI} \cap \mathrm{LPI} \mid \mathrm{RL}) \stackrel{?}{=} P(NI \mid RL)\, P(LPI \mid RL)$$

By d-separation, if LPI is independent of NC conditioned on RL then the reported level is a common cause (confounding variable). This means that even though NC and LPI are dependent, knowledge of RL abolishes that dependency. Conditional independence testing returns a family of graphs up to a certain equivalence class. The following subgraphs are possible:

Figure 6



Note: (a) fork (b) chain (c) chain

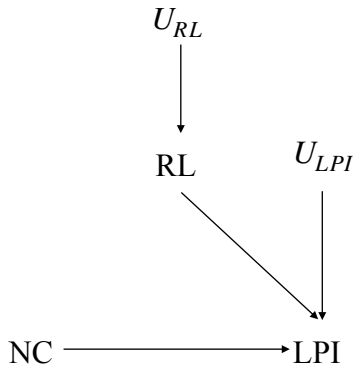
However, the reported level is reported by the student and cannot be manipulated by the remaining observed variables. In other words, it is unlikely that we would see RL influenced by NC or LPI. Therefore, Figure (7.3a) can be the only possible structure that exists between the 3 variables if the conditional independence test LPI ⫫ NC | RL passes.

Figure 7

$$\text{NC} = \begin{cases} 0 & \text{NC} < \text{threshold} \\ 1 & \text{NC} > \text{threshold} \end{cases} \qquad \text{LPI} = \begin{cases} 0 & \text{LPI} < \text{threshold} \\ 1 & \text{LPI} > \text{threshold} \end{cases} \qquad \text{RL} = \begin{cases} 0 & \text{Beginner} \\ 1 & \text{Intermediate} \\ 2 & \text{Advanced} \end{cases}$$

Note: Each variable was transformed into a category based on a specific threshold. For each variable, both the mean and median of the group were considered. The results showed little difference using the two different thresholds.

Figure 8



Note: Manipulated model after performing $do(NC)$. Any arrows into NC are severed. Since NC is now fixed we are generating a new distribution for the system.

In order to estimate the causal effect of NC on LPI, we must find $P(\text{LPI} = 1 \mid do(\text{NC=1}))$ which is the probability that $LPI = 1$ if the student consulted the system more, i.e. $NC = 1$. $do(NC)$ means that we are fixing NC to some value and seeing what happens to LPI. The DAG in Figure 7.3a becomes the DAG in Figure 7.5. The estimated effect of NC on LPI in the new model is just the conditional probability $P(\text{LPI} = 1 \mid do(\text{NC=1}))$, which can be calculated from the observational distribution using Pearl's adjustment formula (Pearl 2009a):

$$P(\text{LPI=1} \mid do(\text{NC=1})) = \sum_{\text{RL}} P(\text{LPI=1} \mid \text{NC=1, RL})\, P(RL).$$

All probabilities on the right-hand side can be computed from the observational distribution. Then, the average causal effect (ACE) is just the difference between $P(\text{LPI=1} \mid do(\text{NC=1}))$ and $P(\text{LPI=1} \mid do(\text{NC=0}))$.

Summary of Results

To estimate the average causal effect of system consults (*NC*) on lines per issue (*LPI*), we use Pearl's adjustment formula and the dichotomized data:

$$\text{ACE} = P(\text{LPI} \mid do(\text{NC=1})) - P(\text{LPI} \mid do(\text{NC=0})),$$

where

$$P(\text{LPI=1} \mid do(\text{NC=1})) = \sum_{\text{RL}} P(\text{LPI=1} \mid \text{NC=1, RL})\, P(RL), \text{ and}$$

$$P(\text{LPI=1} \mid do(\text{NC=0})) = \sum_{\text{RL}} P(\text{LPI=1} \mid \text{NC=0, RL})\, P(RL).$$

The results returned an $ACE = 0.415$. This means that the reported level can be a possible common cause between NC and LPI. However, these results remain inconclusive due to the limited size of the dataset.

7.6 Causal Model Search (Dynamic Case)

Temporal precedence is an important aspect of causality: the cause always proceeds the effect (Granger, 2004). Since the variables LOC, NI and C (see Table 7.3) were recorded at certain time stamps, this gave each variable temporal structure and an unprecedented opportunity to monitor its change during the coding process. Such temporal structure can be leveraged to make causal connections by treating the coding process as a dynamic system whose interactions are observed at different time steps.

Let $\mathbf{X} = \{X_t^1, X_t^2, X_t^3\}$ be a multivariate stochastic process such that X_t^1 is the process representing lines of code (LOC), X_t^2 is the process representing issues, and X_t^3 is the process representing consults (see definitions 6.5, 6.6, and 6.7 in Chapter 6).

We had two sets of timestamps, one reflecting a change in LOC and/or issues (T'), and another reflecting the time at which the student consulted the system for feedback (T'') (see definitions 6.0 and 6.1 in Chapter 6). Before we could commence our test for causality, synchronization of the time steps of the issues time series $\{x_t^2\}_{t \in T'}$ and the consult time series $\{x_t^3\}_{t \in T''}$ was required. This means that we want to have the same timestamps for all three series so that $\{x_t^1\}_{t \in T}$, $\{x_t^2\}_{t \in T}$ and $\{x_t^3\}_{t \in T}$ (see definition 6.2 in Chapter 6). This synchrony had already existed between X_t^1 and X_t^2 since they were both

timestamped according to the commit timeline. Much consideration had gone into this synchronization process. One method considered was taking the timestamps from the issues timeline and engineering an appropriate corresponding value using the consult timeline and the consult time series. An 'appropriate' value for consults at timestamp t'_j from the commit timeline i.e. $x^3_{t'_j}$ would be an aggregate representing the number of consults that occurred on the interval $(t'_i, t'_j]$ where $i < j$. In other words, any timestamp t''_i from the consult timeline such that $t''_i \in (t'_i, t'_j]$ would contribute a single increment $(+1)$ to this aggregate value. However, this resulted in an imbalanced consult timeline with an overwhelming amount of zeros that would hinder the performance of any algorithm. Instead, the two timelines were merged to reflect all timestamps relevant to the time spent in the IDE, whether it was writing code or consulting the system for feedback. Therefore, the timelines of each time series increased in length to $|T|$ in order to include all relevant timestamps in the system, generating a new timeline called the system timeline (see definition 6.2 in Chapter 6). As a result, each time series underwent interpolation and possible extrapolation with suitable values so that each new timestamp had a corresponding value. For instance, each random variable at a given time $t \in T$ in the consult time series is either 0 (no consult occurred) or 1 (consult occurred). As a result, any timestamp added to the consult timeline was given a 0 value for consulting, reflecting the fact that the student did not consult the system at this timestamp. Similarly, the coding timeline and LOC timeline were given, for each added timestamp, a value from the closest (previous) timestamp.

We can then think of each time step as the state S of the system at time t denoted by S_t. Then, $S = \{\mathrm{S}_t\}_{t \in T}$ such that $S \in \mathbb{C}^{|T| \times 3}$ where each $\mathrm{S}_t = (x^1_t, x^2_t, x^3_3)$ is a row in S. Then, S_t is a vector representing the status of each variable at time t. For instance, if at time $t = 5$, LOC $= x^1_5 = 25$, issues $= x^2_5 = 4$, and consult $= x^3_5 = 1$ then $\mathrm{S}_5 = (25, 4, 1)$.

Since there were no forced interventions on this system, $\mathrm{I}_t^i = x_t^i \ \forall \ i$ where $i \in \{1,2,3\}$. This means that S is based purely on observational data.
Then, $\mathscr{P} = (\mathbf{X}, \mathrm{S}, \mathrm{I})$ defines the coding process.

7.6.1 Granger Causality, Transfer Entropy, Convergent Cross Mappings

Granger's method is a statistical hypothesis test that seeks to determine if one time series granger-causes another. Granger believed that information about the driver variable was uniquely contained in the response variable. He hypothesized that including information about the cause in the prediction of the effect will reduce the prediction error. Autoregressive models are used to model time series data by regressing a future value on past values of the series itself up to a certain time lag τ. According to Granger, if the inclusion of another variable in the model reduces the regression error, then the two variables must be causally connected. If consultations at time $t - \tau$ does effect the number of issues at time t then its inclusion in the prediction of the number of issues at time t should reduce the prediction error.

Let $\bar{x}_t^2$ and $\tilde{x}_t^2$ be the predicted values of x^2 at time t, then:

$\bar{x}_t^2 = \sum_{i=1}^{\tau} \alpha_i x_{t-i}^2$ if past values of x^2 up to time lag τ are used to predict x_t^2, and

$\tilde{x}_t^2 = \sum_{i=1}^{\tau} \alpha_i x_{t-i}^2 + \beta_i x_{t-i}^3$ if past values of x^2 and x^3 up to time lag τ are used to predict x_t^2.

$\alpha_i \in \mathbb{R}$ and $\beta_i \in \mathbb{R}$ are the regression coefficients.

If $\left(\tilde{x}_t^2 - x_t^2\right)^2 < \left(\bar{x}_t^2 - x_t^2\right)^2$ then X_t^3 granger-causes X_t^2.

Before performing Granger's test, we had to ensure that each time series was stationary, i.e., the mean was independent of time, and variance was dependent only on time lag τ. Stationarity ensures the persistence of the results for any part of the timeline. From the

graphs in Table 7.6, the astute reader will notice that most of the timelines are not stationary. In particular, we can see a trend in LOC and NI. As a result, a transformation such as differencing was necessary to transform each time series so that the Granger test could be performed.

The Granger test in the Python package statsmodels (Seabold et al., 2010) and in the R package 'lmtest' (Zeileis et al., 2002) was used to conduct the Granger causality test. The Granger causality test was deployed in both directions:

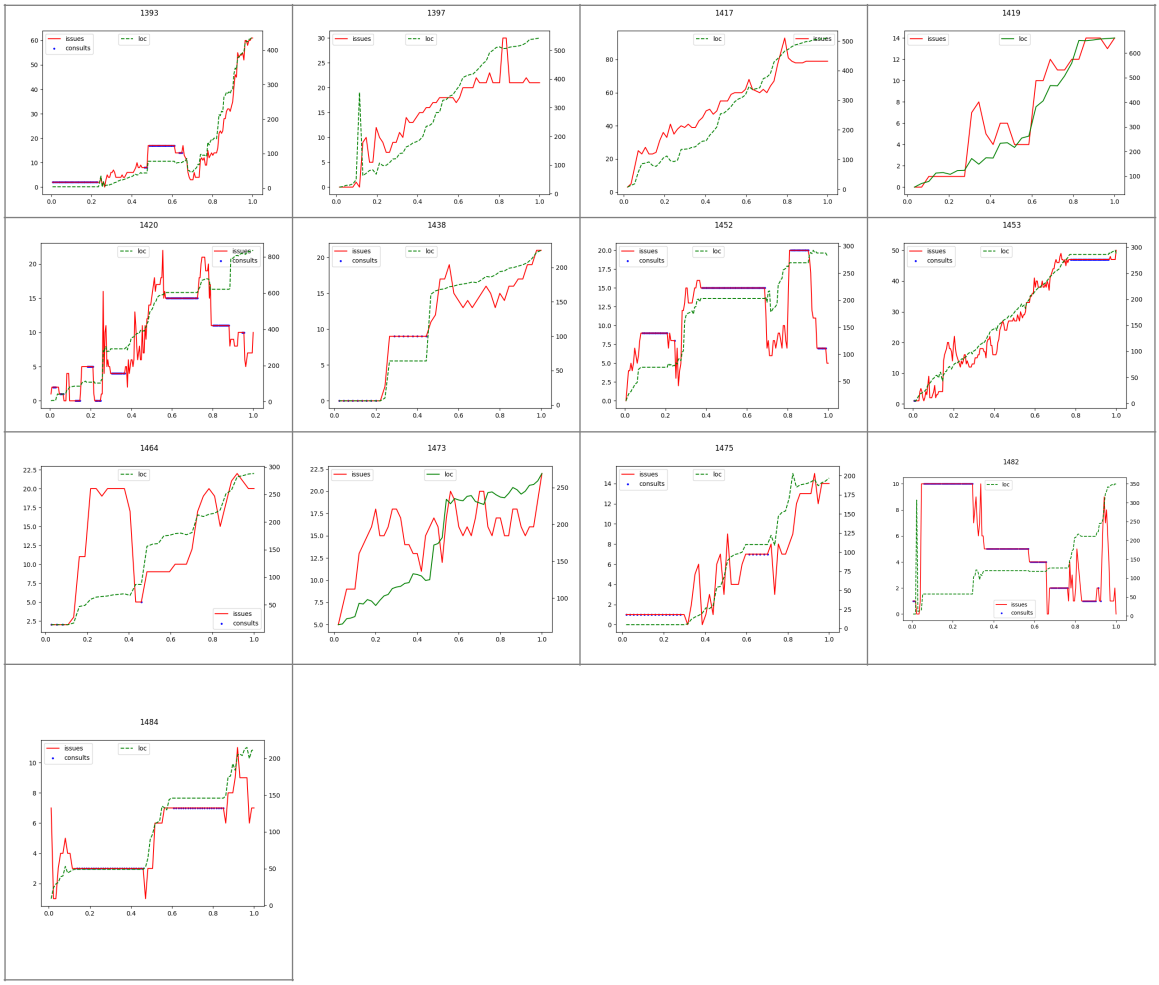
$$X_t^2 \rightarrow X_t^3, \text{ and}$$
$$X_t^3 \rightarrow X_t^2.$$

It is reasonable to think that the causal connection may be in reverse order. If a student is experiencing an issue, whether it is functional or competence-related, she may be inclined to consult the system. Such a possibility warrants a causality test in both directions. An appropriate time lag is difficult to establish in the Granger test and can be investigated by finding the order of the best-fitting autoregressive model. As a result, the time lags $\tau = \{1,..,10\}$ were tested. The Python module VAR found in statsmodels was used to fit each process using the above mentioned τ's. The τ whose model provided the lowest prediction error was used. Since some of the timelines were shorter, as the time lag increased the algorithm failed.

However, the Granger test used is only able to detect linear relationships. This may be problematic as real-world systems, such as learning processes, are complex and can not be easily modelled. Two particular impediments may be at play here. First, a non-linear relationship may exist between X_t^2 and X_t^3. Second, the system may not be separable. In order to account for non-linear systems, an information theoretic approach was deployed using the R package 'RTransferEntropy' (Behrendt et al., 2019) to investigate the presence of a non-linear causal relation. For systems that are not separable and possibly deterministic, an implementation of convergent cross mappings found in the R package rEDM (Sugihara et al., 2019) was used.

Table 7

Student Timelines



Note: The plots of the consult, issues and LOC time series for each student. Most of these timelines required de-trending before performing the Granger test. The x-axis represents the normalized times, the labels on the left y-axis represent the number of issues, and the labels on the right y-axis represent the lines of code.

Summary of Results

Table 7.7 shows the results from the Granger causality tests, transfer entropy tests and convergent cross mappings (CCM) tests. The test for causality was done in both directions as it is quite possible that consulting can cause a decrease in issues or the presence of an issue forces the student to consult. It is also possible to have feedback between both variables such that a bidirectional edge may exist between them. The cells highlighted in grey show a clear causal direction between the variables. Either the p-values are at or below the significance level of $\alpha = 0.05$ for the given test (Granger and transfer entropy), or the reconstructed manifold of one time series successfully predicts values from the other (convergent cross mappings). For these tests, we can stipulate that a causal direction does exist in the given direction.

Looking at the results from each timeline, we can see that both the Granger test and transfer entropy returned a clear causal connection from system consults to the number of issues. Following a number of consults, we can see a drop in issues. In some cases, the Granger tests indicated a causal connection from issues to consults. However, this was not conclusive as it did not appear to be the case in about half the students.

CCM returned mixed results, showing mainly a causal connection from system consults to number of issues. However, it also showed, in some instances, an influence from issues to system consults. This may indicate the presence of feedback between both variables or moderate coupling.

Table 8

Results from Dynamic Tests

Student	Total Consults	Timeline length	IS	TE $X_t^3 \to X_t^2$	TE $X_t^2 \to X_t^3$	GC $X_t^3 \to X_t^2$	GC $X_t^2 \to X_t^3$	CCM
1393	64	162	0.4	0.0	0.5	0.005	0.11	$X_t^3 \to X_t^2, X_t^2 \to X_t^3$
1420	99	218	0.5	0.6	0.5	0.03	0.03	$X_t^3 \to X_t^2$
1438	17	45	0.4	0	0.8	0	0.005	$X_t^3 \to X_t^2, X_t^2 \to X_t^3$
1452	95	163	0.6	0.04	0.9	0.002	0.13	$X_t^3 \to X_t^2, X_t^2 \to X_t^3$
1453	41	206	0.2	0.01	0.13	0.7	0.9	$X_t^3 \to X_t^2$
1464	4	37	0.1	0.04	0.19	0.02	0.0006	$X_t^3 \to X_t^2, X_t^2 \to X_t^3$
1475	23	57	0.4	0.003	0.74	0.03	0.04	$X_t^3 \to X_t^2, X_t^2 \to X_t^3$
1482	124	172	0.7	0.03	0.25	0.06	0.2	$X_t^3 \to X_t^2$
1484	54	87	0.6	0.01	0.82	0.01	0.0001	$X_t^3 \to X_t^2$

Note: Results from the Granger test, transfer entropy and convergent cross mappings. The results show the p-values from each test for each student's timeline. The cells highlighted in grey indicate a p-value below the significance level of $\alpha = 0.05$ or in the case of CCM, these cells indicate a direction of causality

Chapter 8.

8. Discussion

Looking at the data from the controlled experiment, a clear causal effect between system consults and the number of issues at the end of the coding timeline was present. Students who did make use of the feedback provided by the system upon consultation were able to reduce the number of issues at the end of the coding timeline regardless of program length or complexity. Two-thirds of the students from the experimental group opted to use the system and, as a result, contributed to a difference in (number of issues) NI means at the end of the coding timeline between both groups. However, in both analyses, direct and indirect (see 7.4), the mean of NI and IPL was almost double for the control group than that of the experimental group. However, 4 out of the 13 students in the experimental group did not consult the system for feedback, increasing the calculated means (NI and IPL) of this group. We suspect that this difference in means would be even more notable had the entirety of the experimental group made use of system consultations and feedback and or if students used the system more effectively. In addition, the sparseness of the consult timelines for the experimental group signifies that students were not always willing to use the available help or that optimality was not a primary goal so long as a program was functional.

Our causal discovery search within the experimental group yielded inconclusive results in the static models but auspicious results in the dynamic models. In the static two-node models, a causal effect was present under the assumption that a common cause was not present (see 7.5.1). However, the results were not definitive since the independence tests between the noise terms and the corresponding regressors in both models ‘a’ and ‘b’ produced values close to zero. Model ‘a’ did yield a higher score (0.1) than model ‘b’ (-0.1), indicating a weak possibility of a causal connection between the number of consults and issues at the end of the timeline. These are still galvanizing results since we

are looking at aggregate values for consults in our static models, and 1/3 of the experimental group did not consult the system for feedback, and 1/2 of those that did consult the system used it sporadically ($< 40\,\%$ of the time). The number of consultations ranged from 0 to 124, with the highest number of consultations leading to 0 errors at the end of the timeline. However, more consultations did not always mean fewer issues at the end of the timeline. This may signify that students are not learning from past mistakes, and the same pattern of errors is continuously being repeated.

In the static three-node models, conditional independence testing concluded that RL was determined to be a potential confounder. However, the results remain inconclusive. The pc algorithm did not return any results. The edges in the graph were undirected, indicating the evidence was too ambiguous to establish a strong causal connection in the presence of RL. However, this could be due to the small dataset, which could affect the behaviour of the algorithm. Statistical testing, such as correlations and conditional independence tests, are grossly hampered by small datasets. In addition, other possible confounders should also be considered, such as age, gender, socioeconomic background and past experience. These variables are strong contenders as they affect both a student's willingness to consult the system as well as the number of issues they would experience during the programming task.

In the dynamic models, we found that a causal effect was present in most timelines. In the Granger causality tests, 90% of the timelines showed a causal connection from consults to issues ($X_t^3 \to X_t^2$), one of which was close to the significance level $p = 0.05$. In addition, 80% of this subset showed a causal connection in the opposite direction as well ($X_t^2 \to X_t^3$). However, for the causal connections from consults to issues, the time lag τ had an average and median value of 4, whereas causal connections from issues to consults, τ had an average value of 7 and a median value of 8. This implies that it takes about 4-time steps, each signifying an interaction with the system on a specific issue, in order to see a reduction in issues. Viewing precise hints, viewing regulation cards or rule explanations are some of these interactions. In addition, it signifies a more immediate effect of consultations on the number of issues. On the other hand, it takes about 7 time

steps, each signifying a snapshot of the code at a given (random) time, for a student to check the state of issues in the code. It is unclear whether students perform these checks out of sheer curiosity about the competence level of the code or because they are experiencing functional issues and hope to gain insight into why these issues are occurring. Unlike Granger's method, transfer entropy can detect non-linear relationships. Checking for a causal relationship from consults to issues ($X_t^3 \rightarrow X_t^2$), transfer entropy returned a causal relation in 90% of the students in the experimental group. In this subset, all but two of the cases overlap with the Granger causality results. In addition, a time lag of $\tau = 1$ was the predominant value required to model the transfer of information. This implies a more immediate effect of consultations on the number of issues in the code. This also corroborates the results from the Granger tests but also indicates that the relation is nonlinear since a *shorter* time lag than the Granger tests was able to conclude a causal effect from consults to issues. However, transfer entropy returned only negative results in the causal direction from issues to consults. This may not be entirely true since the algorithm failed for longer time lags (only time lags up to $\tau = 3$ were tested), and the Granger tests returned a causal effect from issues to consults for an average time lag of $\tau = 7$. Convergent cross mapping was able to show a causal connection from consults to issues in every timeline, 60% of which showed a causal connection from issues to consults as well. It seemed that the algorithm did converge in most cases. However, the correlation values ρ to which the algorithm converged were lower in the causal direction from issues to consults. Similarly, this ratifies the results produced by the Granger method and transfer entropy. A common thread is present between all three algorithms. One can definitively stipulate the existence of a causal connection between system consults X_t^3 to issues X_t^2. In addition, a weaker, less evident, causal connection appears to be present from X_t^2 to X_t^3. This may indicate a more immediate effect of consults on the number of issues and a more distant effect of issues on the number of consults. This implies that the coupling is stronger in one direction than the other.

In summary, the three causal discovery methods yielded a notable causal effect of system consults on the number of issues at the end of the system timeline. Looking at the results

from Granger and Transfer Entropy, we can see that the time lags ranged between 1 (Transfer Entropy) and 4 (Granger). We also argued that the relation between consults and issues is non-linear since any real-world system is likely to be complex. This signifies that the effect of a consult is almost immediate since we see a reduction in issues in the near future (next timestamp).

Chapter 9.

9. Recommendations

9.1 At the Data Collection Level

Much work is also required at the data collection level. The time to complete the tasks should be limited and the same for all participants. For instance, making the plugin available to all students during the semester will foster this goal. It will automatically enforce a time limit on each task, which will be the time between the assignment release and the assignment due date. Participants from the same cohort should be given the same programming task, eradicating the need to normalize the number of issues per task level or at least making the raw timeline an option to analyze. In addition, commits to the VCS should not be irregular and controlled solely by the system to ensure regularly timed commits reporting the number of issues and current number of lines of code. This would allow for more consistent timelines. This would also provide other useful information, such as pique productivity times for each student, which can be leveraged to improve learner experience and provide timely feedback. The time stamping of the consults would remain the same since it solely depends on student engagement and curiosity. However, if the student has access to a human tutor, then these consults may be forced interventions on the coding process and, therefore, must be labelled as such i.e. forced and not voluntary. In addition, another process that reports on functional issues can be added to the coding process. A set of labels defining functional milestones can be curated beforehand. Thus, consultations in this regard would be exclusive to the human tutor. Therefore, differentiating between both consults (system and human) may be desired to distinguish between them.

9.2 At the Consult Level

In addition to reducing the number of issues, the goal is also to improve students' understanding of the violations that are occurring. As such, mindlessly consulting the system without putting thought into an issue and why it is occurring negates the purpose of the system. Timely feedback on the issues in their code is critical, but we also want students to learn from past mistakes so that best practices are cemented. As such, we recommend the introduction of an engineered metric called the improvement score (Table 7.3). The consult timeline is checked for duplicate occurrences of a specific rule. If a rule appears often, then the student will be alerted and warned of potential system misuse. This will force the student to take a step back and reflect on the issue. The student is then encouraged to engage further with the rule explanations and examples provided to understand the issue and its occurrence better. The improvement score is inversely proportional to the number of duplicates. A higher number of duplicates means a lower improvement score.

9.3 At the Assignment Level

As we saw in the given timelines, 4 out of 13 students from the experimental group did not make use of system consults. As an incentive, a portion of the grade can be allocated to the improvement score and another portion to the number of issues present at the end of the commit timeline. The former will encourage students to use the system effectively and properly in order to solidify best practices. The latter will simply encourage its use.

10. Other Applications

Indisputably, the improvement of learner experience and outcomes is universally advocated. New approaches are constantly being tested, and old approaches are challenged in order to reach a wide range of learners and cater to their individual styles. As such, there is a persistent desire to determine the effect of a new teaching strategy or the cause of an upswing in student performance not just in computer science but across other disciplines as well.

It is important to note that the analysis and methodology presented herein can be applied in any discipline where learner data is collected either statically or dynamically. For instance, in automated essay scoring (Ke et. al. 2019), the predicted score of an essay is based on several features such as grammatical accuracy, lexical diversity and word count. However, it might be desirable to inspect the impact or the effect of a feature on the overall score of an essay in order to prescribe a plan to improve the outcome. In addition, seemingly uncorrelated features might have an influence on one another, and thus, improving one might improve the other, rendering a positive impact on the final score. Similarly, in automated short answer grading in mathematics (Zhang et. al. 2022), or any other discipline for that matter, it also might be desirable to find the impact of one feature on another or the impact a feature might have on the overall quality of the answer. We can deploy either our static or dynamic approach. In the essay scoring scenario, aggregate values of any features of interest can be registered and preprocessed appropriately and then an applicable causal discovery approach can be used (section 7.5). More interestingly, if data can be collected longitudinally where the essay writing process is monitored via a version control system, we can deploy our dynamic approach. The latter provides the means to conduct the analysis at a granular level by observing the dynamics of the system over time. How the variables change and interact with one another offers a rare glimpse into the learning process and student progress over time. In addition, it may

be a more appropriate approach when the number of participants is smaller and presents an obstacle in the static case.

11. Final Notes

Causality is a fascinating topic that has garnered attention for millennia. Yet, due to its fundamental intricacy, its definition remains disputed. The ability to isolate cause from effect or reason from consequence has allowed humanity to adapt and progress. Data with temporal structure like time series data offers a unique setting to disentangle cause from effect since we are able to monitor the change in variables over time. We argue that cause and effect form a system whose dynamics remain consistent over time and are best modelled and disentangled using approaches and methods from dynamical system theory. The world around us consists of many systems such as social, biological, economical and many others such as educational. Just like in these systems, the governing body of a cause and effect system depends on the parts at play and the nature of the interactions between them. The behaviour of such systems can not be explained by a single part or broken down into a sum of parts. For instance, in biological systems, the growth or decline of a predators population over time can not be explained by its past populace alone but also the number of prey and other factors such as disease and environment. Similarly, a change in the effect can only be explained by the dynamics of the system which includes all influencing factors. The size and complexity of these systems is unknown as one can never, with conviction, quantify all driving forces of a single component. However, as people of science, we often aspire to find appropriate models for these systems even if principled approaches or base equations do not exist yet. In the language of do-calculus, we can simulate interventions on these systems in three different ways: 1- fixing a specific variable, 2- bifurcating the system by modifying its parameters or 3- perturbing the system by some value ϵ. Then we can study the behaviour and conduct a stability analysis on the system to find the effect of the intervention.

In this thesis, we set forth the preliminary steps for defining a coding process and what it entails. We define any processes of interest that are involved, the types of interventions that are permissible and the different states the system can achieve. To the best of our

knowledge, at the time this thesis was written, this has not been done before. This paves the way for studying and analyzing a students coding activities in a principled way using mathematical concepts and notations.

References

Baker, R., & Siemens, G. (2014). Learning analytics and educational data mining. Cambridge handbook of the leaning sciences (2nd edn). Cambridge University Press: New York, NY, 253-272.

Behrendt, S., Dimpfl, T., Peter, F. J., & Zimmermann, D. J. (2018). RTransferEntropy: Measuring Information Flow Between Time Series with Shannon and Renyi Transfer Entropy. In *R Package Version 0.2. 7*.

Bellot, A., Branson, K., & van der Schaar, M. (2021). Consistency of mechanistic causal discovery in continuous-time using neural odes. *arXiv preprint arXiv:2105.02522*.

Boulanger, D., Seanosky, J., Guillot, R., Guillot, I., Guillot, C., Fraser, S., ... & Kinshuk, K. (2019, July). Assessing Learning Analytics Impact on Coding Competence Growth. In *2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT)* (Vol. 2161, pp. 170-172). IEEE.

Brockwell, P. J., & Davis, R. A. (2009). *Time series: theory and methods*. Springer science & business media.

Chan, K., Low, B. T., Lam, W., & Lam, K. P. (2005). Extracting causation knowledge from natural language texts. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2336, 555–560. https:// doi.org/ 10.1007/3-540-47887-6_55

Colombo, D., Maathuis, M. H., Kalisch, M., & Richardson, T. S. (2012). Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, 294-321.

Dash, D. (2005, January). Restructuring dynamic causal systems in equilibrium. In *International Workshop on Artificial Intelligence and Statistics* (pp. 81-88). PMLR.

Davison, A. C. (2003). *Statistical models* (Vol. 11). Cambridge university press.

Dietz, L. W., Manner, J., Harrer, S., & Lenhard, J. (2018). Teaching clean code. In *PROCEEDINGS of the 1st Workshop on Innovative Software Engineering Education*.

Granger, C. W. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, 424-438.

Granger, C. W. J., & Hatanaka, M. (2015). Spectral Analysis of Economic Time Series.(PSME-1) (Vol. 2066). Princeton university press.

Granger, C. W. J. (2004). Time series analysis, cointegration, and applications. *American Economic Review*, *94*(3), 421-425.

Guo, R., Cheng, L., Li, J., Hahn, P. R., & Liu, H. (2020). A survey of learning causality with data: Problems and methods. *ACM Computing Surveys (CSUR)*, *53*(4), 1-37.

Harnack, D., Laminski, E., Schünemann, M., & Pawelzik, K. R. (2017). Topological causality in dynamical systems. *Physical review letters*, *119*(9), 098301.

Heinze-Deml, C., Maathuis, M. H., & Meinshausen, N. (2018). Causal structure learning. *Annual Review of Statistics and Its Application*, *5*, 371-391.

Hoyer, P., Janzing, D., Mooij, J. M., Peters, J., & Schölkopf, B. (2008). Nonlinear causal discovery with additive noise models. *Advances in neural information processing systems*, *21*.

Hundhausen, C.D., Olivares, D.M., & Carter, A.S. (2017). IDE-Based Learning Analytics for Computing Education. ACM Transactions on Computing Education (TOCE), 17, 1 - 26.

Hwang, T. (2018). Computational power and the social impact of artificial intelligence. *arXiv preprint arXiv:1803.08971*.

Jordan, M. I. (2004). Graphical models.

Kalainathan, D., & Goudet, O. (2019). Causal discovery toolbox: Uncover causal relationships in python. *arXiv preprint arXiv:1903.02278*.

Ke, Z., & Ng, V. (2019, August). Automated Essay Scoring: A Survey of the State of the Art. In *IJCAI* (Vol. 19, pp. 6300-6308).

Keuning, H., Heeren, B., & Jeuring, J. (2017). Code quality issues in student programs. In Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE (Vol. Part F128680, pp. 110–115). Association for Computing Machinery. https://doi.org/10.1145/3059009.3059061

Kirk, D., Crow, T., Luxton-Reilly, A., & Tempero, E. (2020, February). On assuring learning about code quality. In *Proceedings of the Twenty-Second Australasian Computing Education Conference* (pp. 86-94).

Koller, D., Friedman, N., Getoor, L., & Taskar, B. (2007). Graphical models in a nutshell. *Introduction to statistical* relational learning, 43.

Lewis, D. (1974). Causation. *The journal of philosophy*, *70*(17), 556-567.

Östlund, L., Wicklund, N., & Glassey, R. (2023, March). It's Never too Early to Learn About Code Quality: A Longitudinal Study of Code Quality in First-year Computer Science Students. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (pp. 792-798).

Paluš, M., Krakovská, A., Jakubík, J., & Chvosteková, M. (2018). Causality, dynamical systems and the arrow of time. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *28*(7), 075307.

Pearl, J. (1995). Causal diagrams for empirical research. *Biometrika*, *82*(4), 669-688.

Pearl, J. (2009a). Causal inference in statistics: An overview.

Pearl, J. (2009b). *Causality*. Cambridge university press

Pearl, J. (2010). An introduction to causal inference. *The International Journal of Biostatistics*, *6*(2), 1-62.

Pearl, J., & Mackenzie, D. (2018). The book of why: the new science of cause and effect. Basic books.

Peters, J., Janzing, D., & Schölkopf, B. (2012). Causal Inference using the Algorithmic Markov Condition. In Advances in Neural Information Processing Systems (pp. 1410-1418).

Peters, J., Janzing, D., & Schölkopf, B. (2017). Elements of causal inference: foundations and learning algorithms (p. 288). The MIT Press.

Peters, J., Bauer, S., & Pfister, N. (2022). Causal models for dynamical systems. In *Probabilistic and Causal Inference: The Works of Judea Pearl* (pp. 671-690).

Peña, A., Sossa, H., & Gutiérrez, A. (2008). Causal knowledge and reasoning by cognitive maps: Pursuing a holistic approach. Expert Systems with Applications, 35(1-2), 2-18.

Pfister, N., Bauer, S., & Peters, J. (2019). Learning stable and predictive structures in kinetic systems. *Proceedings of the National Academy of Sciences*, *116*(51), 25405-25411.

Reichenbach, H. (1956). The direction of time. *Berkeley:* University of California.

Rubenstein, P. K., Bongers, S., Schölkopf, B., & Mooij, J. M. (2016). From deterministic ODEs to dynamic structural causal models. *arXiv preprint arXiv:1608.08028*.

Rubin, D., & Zell, E. (Eds.) (2018). . (Vols. 1-4). SAGE Publications, Inc., https://doi.org/10.4135/9781506326139

Schölkopf, B. (2019). Causality for Machine Learning. *arXiv e-prints*, arXiv-1911.

Schölkopf, B., Locatello, F., Bauer, S., Ke, N. R., Kalchbrenner, N., Goyal, A., & Bengio, Y. (2021). Towards causal representation learning 2021. *arXiv preprint arXiv:2102.11107*.

Schölkopf, B., & von Kügelgen, J. (2022). From statistical to causal learning. *arXiv preprint arXiv:2204.00607*.

Schreiber, T. (2000). Measuring information transfer. *Physical review letters*, *85*(2), 461.

Seabold, Skipper, and Josef Perktold. "Statsmodels: Econometric and statistical modeling with python." *Proceedings of the 9th Python in Science Conference.* 2010

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, *27*(3), 379-423.

Shojaie, A., & Fox, E. B. (2022). Granger causality: A review and recent advances. *Annual Review of Statistics and Its Application*, *9*, 289-319.

Spirtes, P., Glymour, C. N., & Scheines, R. (2000). *Causation, prediction, and search*. MIT press.

Sugihara, G., May, R., Ye, H., Hsieh, C. H., Deyle, E., Fogarty, M., & Munch, S. (2012). Detecting causality in complex ecosystems. *science*, *338*(6106), 496-500.

Sugihara, G., Park, J., Deyle, E., Saberski, E., Smith, C., & Ye, H. (2019). rEDM: An R Package for Empirical Dynamic Modeling and Convergent Cross Mapping.

Strogatz, S. H. (1994). Nonlinear dynamics and chaos: With applications to physics, biology, chemistry, and engineering. CRC press.

Takens, F. (1981). Detecting strange attractors in turbulence. *Lecture Notes in Mathematics, Berlin Springer Verlag*, *898*, 366.

Winship, C., & Morgan, S. L. (1999). The estimation of causal effects from observational data. *Annual review of sociology*, *25*(1), 659-706.

Wold, H. (1956). Causal inference from observational data: A review of end and means. *Journal of the Royal Statistical Society. Series A (General)*, *119*(1), 28-61

Zeileis A, Hothorn T (2002). "Diagnostic Checking in Regression Relationships." *R News*, **2**(3), 7–10. https://CRAN.R-project.org/doc/Rnews/.

Zhang, M., Baral, S., Heffernan, N., & Lan, A. (2022). Automatic short math answer grading via in-context meta-learning. *arXiv preprint arXiv:2205.15219*.