Journal of Artificial Intelligence Research 80 (2024) 835-857 Submitted 10/2023; published 06/2024

Simulating Counterfactuals

Juha Karvanen JUHA.T.KARVANEN@JYU.FI
Santtu Tikka SANTTU.TIKKA@JYU.FI
Matti Vihola MATTI.S.VIHOLA@JYU.FI
Department of Mathematics and Statistics
University of Jyvaskyla, Finland

Abstract

Counterfactual inference considers a hypothetical intervention in a parallel world that shares some evidence with the factual world. If the evidence specifies a conditional distribution on a manifold, counterfactuals may be analytically intractable. We present an algorithm for simulating values from a counterfactual distribution where conditions can be set on both discrete and continuous variables. We show that the proposed algorithm can be presented as a particle filter leading to asymptotically valid inference. The algorithm is applied to fairness analysis in credit-scoring.

1. Introduction

A counterfactual distribution is the probability distribution of a random variable under a hypothetical scenario that differs from the observed reality. "What would have been the outcome for this individual if they had received a different treatment?" is an example of a counterfactual question. Here the personal data of the individual constitute the evidence that specifies the observed reality, and the interest lies in the distribution of the outcome under a hypothetical treatment.

Counterfactual questions belong to the third and highest level in the causal hierarchy (Shpitser & Pearl, 2008) and are in general more difficult than associational (first level) or interventional (second level) questions. Algorithms for checking the identifiability of counterfactual queries from observational and experimental data have been developed (Shpitser & Pearl, 2007; Shpitser & Sherman, 2018; Correa, Lee, & Bareinboim, 2021) and implemented (Tikka, 2023). In many practical cases, the queries may be non-identifiable (Wu, Zhang, Wu, & Tong, 2019).

Counterfactuals are often linked with questions about fairness, guilt, and responsibility. Notably, the fairness of prediction models has become a major concern in automated decision-making (Wachter, Mittelstadt, & Russell, 2018; Pessach & Shmueli, 2022) where the requirement for fairness often arises directly from the legislation. A classic example of the fairness requirement in decision-making is credit-scoring where the bank's decision to lend must not depend on sensitive variables such as gender or ethnicity (Bartlett, Morse, Stanton, & Wallace, 2022).

Several definitions and measures of fairness have been proposed (Mehrabi, Morstatter, Saxena, Lerman, & Galstyan, 2021; Caton & Haas, 2023; Carey & Wu, 2022). Among these proposals, counterfactual definitions of fairness (Kusner, Loftus, Russell, & Silva, 2017; Nabi & Shpitser, 2018; Chiappa, 2019; Wu et al., 2019; Richens, Beard, & Thompson, 2022) are intuitively appealing as they compare whether an individual decision would remain the same in a hypothetical counterfactual world. Note that the term "counterfactual explanations" (Guidotti, 2022) is sometimes used in the literature on explainable artificial intelligence (XAI) and interpretable machine learning (Burkart & Huber, 2021) in contexts where the term "contrastive explanations" proposed by Karimi, Schölkopf, and Valera (2021) would be more appropriate.

We consider the problem of simulating observations from a specified counterfactual distribution. Given a structural causal model (SCM) and a counterfactual of interest, the counterfactual distribution can be derived in three steps (Pearl, 2009). First, the distribution of the latent background variables is updated given the evidence expressed as a set of conditions on the observed variables. Second, the causal model is modified according to the hypothetical intervention. Third, the counterfactual distribution of the target variable is calculated in the model that represents the counterfactual scenario under the updated distribution. These three steps require the full knowledge of the causal model in the functional form, i.e. the structural equations and the distributions of the background variables must be known.

The problem of simulating counterfactuals is similar to the problem of simulating observations from a given distribution. The challenges are related to the first step of counterfactual inference which requires determining a multivariate conditional distribution. This can be done analytically only in special cases where, for instance, all the variables are either discrete or normally distributed. Evidence that includes continuous variables leads to a conditional distribution concentrated on a manifold, which is generally difficult to simulate.

In this paper, we present an algorithm for simulating counterfactual distributions. The algorithm is applicable in settings where the causal model is fully known in a parametric form and the structural equations for continuous variables have an additive error term or, more generally, are strictly monotonic with respect to an error term. The algorithm is essentially tuning-free. Unlike Karimi et al. (2021) and Javaloy, Sánchez-Martín, and Valera (2023), we allow the SCM to contain background variables that affect two or more observed variables, which makes it significantly harder to simulate the counterfactual distribution.

The algorithm processes the conditioning variables one by one in a topological order and obtains an approximate sample from the updated distribution of the background variables (step 1). The challenge of continuous conditioning variables is overcome by using binary search to find solutions that satisfy the conditions and then applying sequential Monte Carlo to calibrate the distribution of these solutions. Discrete conditioning variables are processed by resampling observations that satisfy the condition. Next, the causal model is modified (step 2) and a sample from the counterfactual distribution is acquired by simulating the modified causal model with the obtained sample of the background variables (step 3).

We show that the conditional simulation at step 1 can be interpreted as a particle filter/sequential Monte Carlo (e.g., Gordon, Salmond, & Smith, 1993; Doucet, Godsill, & Andrieu, 2000; Del Moral, 2004; Cappé, Moulines, & Rydén, 2005; Chopin & Papaspiliopoulos, 2020). Theoretical results from the particle filter literature guarantee good asymptotic properties of the sample. In particular, we state a mean square error convergence rate and a central limit theorem for samples obtained with the proposed algorithm.

In real-world applications, the full knowledge of the causal model may not be available. Despite this serious restriction, we argue that the simulation-based counterfactual inference may still have its role in fairness evaluation. Consider a prediction model that is used for decision-making in a situation where the underlying causal model is unknown. To ensure fairness in this situation, the prediction model should be fair under any reasonable causal model. This implies that an analyst evaluating the fairness of the prediction model could choose some reasonable causal models for the evaluation. Deviations from fairness under any of these causal models indicate that the prediction model is not fair in general.

We use the counterfactual simulation algorithm as the main component in a fairness evaluation algorithm of prediction models. We simulate data from specified counterfactual distributions and compare the predictions for these settings. We demonstrate with a credit-scoring example how the fairness evaluation algorithm can be applied to opaque AI models without access to real data. As a result, we find out if an AI model is fair in the tested setting, and if not, learn how large differences there are in the outcome under different counterfactual interventions.

The rest of the paper is organized as follows. The notation and the basic definitions are given in Section 2. In Section 3, the counterfactual simulation algorithm and the fairness evaluation algorithm are introduced, and the conditional simulation is presented as a particle filter. In Section 4, the performance of the simulation algorithm is tested in benchmark cases where the counterfactual distributions can be derived analytically. In Section 5, the fairness evaluation algorithm is applied to a credit-scoring example. Section 6 concludes the paper.

2. Notation and Definitions

We use uppercase letters to denote variables, lowercase letters to denote values, and bold letters to denote sets of variables or values. The primary object of counterfactual inference is a causal model (Pearl, 2009):

Definition 1 (SCM). *A structural causal model $\mathcal{M}$ is a tuple $(\mathbf{V}, \mathbf{U}, \mathbf{F}, p(\mathbf{u}))$, where*

(1) $\mathbf{V}$ *is a set of observed (endogenous) variables that are determined by other variables in the model.*

(2) $\mathbf{U}$ *is a set of background variables that are unobserved random variables.*

(3) $\mathbf{F}$ *is a set of functions $\{f_V \mid V \in \mathbf{V}\}$ such that each f_V is a mapping from $\mathbf{U} \cup (\mathbf{V} \setminus \{V\})$ to V and such that $\mathbf{F}$ forms a mapping from $\mathbf{U}$ to $\mathbf{V}$. Symbolically, the set of equations $\mathbf{F}$ can be represented by writing $V = f_V(\mathrm{Pa}^*(V))$, where $\mathrm{Pa}^*(V) \subset \mathbf{U} \cup (\mathbf{V} \setminus \{V\})$ is the unique minimal set of variables sufficient for representing f_V.*

(4) $p(\mathbf{u})$ *is the joint probability distribution of* $\mathbf{U}$*.*

An SCM is associated with a directed graph $\mathcal{G}$ where there is an edge $W \to V$ if and only if $W \in \mathrm{Pa}^*(V)$. In other words, $\mathrm{Pa}^*(V)$ denotes the (observed and unobserved) parents of V in graph $\mathcal{G}$. We will consider only recursive SCMs where the set $\mathbf{F}$ defines a *topological order* of the variables $\mathbf{V} \cup \mathbf{U}$, i.e., an order where $\mathrm{Pa}^*(Z) < Z$ for all $Z \in \mathbf{V} \cup \mathbf{U}$. The graph associated with a recursive SCM is a directed acyclic graph (DAG). Notation $\tau(V)$ refers to the variables that precede V in the topological order meaning variables $W \in \mathbf{V} \cup \mathbf{U}$ such that $W < V$. We assume that variables $\mathbf{U}$ precede $\mathbf{V}$ in the topological order. Notation $\mathrm{Pa}(V)$ is a shorthand notation for $\mathrm{Pa}^*(V) \cap \mathbf{V}$ meaning the observed parents. Similarly, $\mathrm{An}^*(V)$ denotes the ancestors of V in $\mathcal{G}$ and $\mathrm{An}(V) = \mathrm{An}^*(V) \cap \mathbf{V}$. Notations $\tau(v)$, $\mathrm{Pa}^*(v)$,

$\mathrm{Pa}(v)$, $\mathrm{An}^*(v)$ and $\mathrm{An}(v)$ refer to the values of the variables $\tau(V)$, $\mathrm{Pa}^*(V)$, $\mathrm{Pa}(V)$, $\mathrm{An}^*(V)$ and $\mathrm{An}(V)$, respectively.

Data can be simulated from an SCM by generating the values $\mathbf{u}$ of the background variables $\mathbf{U}$ from the distribution $p(\mathbf{u})$ and then applying the functions $\mathbf{F}$ in the topological order to obtain values $\mathbf{v}$ of $\mathbf{V}$. For a data matrix $\mathbf{D}$ with named columns (variables), the following notation is used: $\mathbf{D}[i,]$ refers to the ith row, $\mathbf{D}[C=c,]$ where C is a column name of $\mathbf{D}$ refers to rows where condition $C=c$ is fulfilled, $\mathbf{D}[,\mathbf{S}]$ refers to data on variables $\mathbf{S}$, i.e., columns whose variable names are in the set $\mathbf{S}$, and $\mathbf{D}[i,\mathbf{S}]$ refers to the ith row of variables $\mathbf{S}$. The notation is similar to many high-level programming languages.

An intervention $\mathrm{do}(\mathbf{X}=\mathbf{x})$ targeting SCM $\mathcal{M}$ induces a *submodel* (Pearl, 2009), denoted by $\mathcal{M}_{\mathrm{do}(\mathbf{X}=x)}$, where those functions in $\mathcal{M}$ that determine the variables $\mathbf{X}$ are replaced by constant functions that output the corresponding values $\mathbf{x}$. We also use the subscript $\mathrm{do}(\mathbf{X}=x)$ to distinguish variables in the submodel from the original variables, e.g., $\mathbf{W}_{\mathrm{do}(\mathbf{X}=\mathbf{x})}$ is the set of variables $\mathbf{W}$ in $\mathcal{M}_{\mathrm{do}(\mathbf{X}=x)}$. The joint distribution of $\mathbf{V}_{\mathrm{do}(\mathbf{X}=x)}$ in the submodel $\mathcal{M}_{\mathrm{do}(\mathbf{X}=x)}$ is known as the *interventional distribution* or *causal effect*. The effects of interventions typically relate to interventional considerations such as the effect of a treatment on a response, but they also facilitate the analysis of counterfactuals that consider hypothetical actions that run contrary to what was observed in reality.

For an SCM where all variables are discrete, a counterfactual distribution can be evaluated by marginalizing over those values of the background variables that result in the specified evidence. More specifically, the probability distribution of a set of observed variables $\mathbf{W} \subset \mathbf{V} \setminus \mathbf{X}$ in the submodel $\mathcal{M}_{\mathrm{do}(\mathbf{X}=\mathbf{x})}$ conditional on a set of observations (the evidence) $\mathbf{C}=\mathbf{c}$ such that $\mathbf{C},\mathbf{X} \subseteq \mathbf{V}$ can be written as

$$p(\mathbf{W}_{\mathrm{do}(\mathbf{X}=\mathbf{x})} = \mathbf{w} \,|\, \mathbf{C}=\mathbf{c}) = \sum_{\{\mathbf{u} \,|\, f^{\circ}_{\mathbf{W}}(\mathbf{u})=\mathbf{w}\}} p(\mathbf{U}=\mathbf{u}|\mathbf{C}=\mathbf{c}),$$

where $f^{\circ}_{\mathbf{W}}$ denotes the functions of $\mathbf{W}_{\mathrm{do}(\mathbf{X}=\mathbf{x})}$ in the submodel $\mathcal{M}_{\mathrm{do}(\mathbf{X}=\mathbf{x})}$ expressed in terms of $\mathbf{U}$ (such functions always exist due to property (3) of Definition 1). The distribution can be also expressed as

$$p(\mathbf{W}_{\mathrm{do}(\mathbf{X}=\mathbf{x})} = \mathbf{w} \,|\, \mathbf{C}=\mathbf{c}) = \sum_{\mathbf{u}} I(f^{\circ}_{\mathbf{W}}(\mathbf{u})=\mathbf{w}) p(\mathbf{U}=\mathbf{u}|\mathbf{C}=\mathbf{c}).$$

This formulation generalizes to the scenario where the variables $\mathbf{U}$ are continuous as follows

$$p(\mathbf{W}_{\mathrm{do}(\mathbf{X}=\mathbf{x})} \in A \,|\, \mathbf{C}=\mathbf{c}) = \int I(f^{\circ}_{\mathbf{W}}(\mathbf{u}) \in A) p(\mathbf{U}=\mathbf{u}|\mathbf{C}=\mathbf{c}) \, \mathrm{d}\mathbf{u}.$$

where A is some event.

The submodel $\mathcal{M}_{\mathrm{do}(\mathbf{X}=\mathbf{x})}$ and the updated distribution $p(\mathbf{U}=\mathbf{u} \,|\, \mathbf{C}=\mathbf{c})$ together define a *parallel world* where the state of the background variables is shared with the non-interventional world. In contrast to interventional distributions, the sets of variables $\mathbf{C}$ and $\mathbf{X}$ defining the evidence $\mathbf{C}=\mathbf{c}$ and the intervention $\mathrm{do}(\mathbf{X}=\mathbf{x})$ need not be disjoint for a counterfactual distribution. If the sets are disjoint, a counterfactual distribution simply reduces to a conditional interventional distribution.

We follow the general procedure introduced by Pearl (2009) to evaluate counterfactual distributions.

Definition 2 (Evaluation of a counterfactual distribution)**.** *Let* $\mathcal{M} = (\mathbf{V}, \mathbf{U}, \mathbf{F}, p(\mathbf{u}))$ *be a recursive SCM and let* $\mathbf{C}, \mathbf{X} \subseteq \mathbf{V}$ *and* $\mathbf{W} \subseteq \mathbf{V} \setminus \mathbf{X}$*. The counterfactual distribution* $p(\mathbf{W}_{\mathrm{do}(\mathbf{X}=\mathbf{x})} = \mathbf{w} \,|\, \mathbf{C} = \mathbf{c})$ *can be evaluated using the following three steps.*

1. *Update* $p(\mathbf{U} = \mathbf{u})$ *by the evidence* $\mathbf{C} = \mathbf{c}$ *to obtain* $p(\mathbf{U} = \mathbf{u} \,|\, \mathbf{C} = \mathbf{c})$ *(Bayes' Theorem)*

2. *Construct the submodel* $\mathcal{M}_{\mathrm{do}(\mathbf{X}=\mathbf{x})}$ *corresponding to the counterfactual scenario.*

3. *Use the submodel* $\mathcal{M}_{\mathrm{do}(\mathbf{X}=\mathbf{x})}$ *and the updated distribution* $p(\mathbf{U} = \mathbf{u} \,|\, \mathbf{C} = \mathbf{c})$ *to compute* $p(\mathbf{W}_{\mathrm{do}(\mathbf{X}=\mathbf{x})} = \mathbf{w} \,|\, \mathbf{C} = \mathbf{c})$*.*

In practice, however, this procedure may not be directly applicable because it is often not possible to analytically compute $p(\mathbf{W}_{\mathrm{do}(\mathbf{X}=\mathbf{x})} = \mathbf{w} \,|\, \mathbf{C} = \mathbf{c})$ in the third step, for example when some of the variables in the conditioning set $\mathbf{C}$ are continuous. A simple illustration of counterfactual inference in a case where the distribution $p(\mathbf{W}_{\mathrm{do}(\mathbf{X}=\mathbf{x})} = \mathbf{w} \,|\, \mathbf{C} = \mathbf{c})$ can be derived analytically is presented in Online Appendix 1.

Our objective is to simulate observations from counterfactual distributions given an SCM in the general case where an analytical solution is not available. Before the evaluation of a counterfactual distribution, the SCM can be pruned similarly to what is done by causal effect identification algorithms (Tikka & Karvanen, 2018) by restricting our attention to only those variables that are relevant to the task. For this purpose, we define an ancestral SCM as follows:

Definition 3 (Ancestral SCM)**.** *Let* $\mathcal{M} = (\mathbf{V}, \mathbf{U}, \mathbf{F}, p(\mathbf{u}))$ *be a recursive SCM and let* $\mathbf{Z} \subseteq \mathbf{V} \cup \mathbf{U}$*. Then* $\mathcal{M}_{\mathbf{Z}} = (\mathbf{V}', \mathbf{U}', \mathbf{F}', p(\mathbf{u}'))$ *is the ancestral SCM of* $\mathcal{M}$ *with respect to* $\mathbf{Z}$ *where* $\mathbf{V}' = \mathbf{V} \cap (An(\mathbf{Z}) \cup \mathbf{Z})$*,* $\mathbf{U}' = \mathbf{U} \cap (An^*(\mathbf{Z}) \cup \mathbf{Z})$*, and* $\mathbf{F}'$ *is the subset of* $\mathbf{F}$ *that contains the functions for* $\mathbf{V}'$*.*

To evaluate a counterfactual distribution, it suffices to restrict our attention to the relevant ancestral SCM.

Theorem 1. *Let* $\mathcal{M} = (\mathbf{V}, \mathbf{U}, \mathbf{F}, p(\mathbf{u}))$ *be a recursive SCM and let* $\eta = p(\mathbf{W}_{\mathrm{do}(\mathbf{X}=\mathbf{x})} = \mathbf{w} \,|\, \mathbf{C} = \mathbf{c})$ *be a counterfactual distribution such that* $\mathbf{W} \cup \mathbf{X} \cup \mathbf{C} \subseteq \mathbf{V}$*. Then* η *evaluated via Definition 2 in* $\mathcal{M}$ *is equivalent to* η *evaluated via Definition 2 in the ancestral SCM* $\mathcal{M}_{\mathbf{W} \cup \mathbf{X} \cup \mathbf{C}}$*.*

Proof. Consider a variable $Y \notin \mathrm{An}^*(\mathbf{W} \cup \mathbf{X} \cup \mathbf{C})$. As Y is not an ancestor of $\mathbf{W}$, $\mathbf{X}$, or $\mathbf{C}$, it does not have any impact in steps 1–3 of Definition 2. □

This type of pruning is a useful operation because it often reduces the number of variables in the SCM and thus allows for more efficient simulation.

The main application of counterfactual simulation is counterfactual fairness for which different definitions have been proposed. Let S be a sensitive variable (sometimes referred to as a protected variable), Y the outcome of interest, and $\widehat{Y}$ an estimator of Y. Kusner et al. (2017) define $\widehat{Y}$ to be counterfactually fair if

$$p(\widehat{Y}_{\mathrm{do}(S=s)} \,|\, \mathbf{X} = \mathbf{x}, S = s) = p(\widehat{Y}_{\mathrm{do}(S=s')} \,|\, \mathbf{X} = \mathbf{x}, S = s) \qquad (1)$$

under any context $\mathbf{X} = \mathbf{x}$ and for any values s and s' of S. In other words, the evidence $\mathbf{X} = \mathbf{x}$ and $S = s$ has been observed and in this situation, the distribution of $\widehat{Y}$ should be the same under the intervention $\mathrm{do}(S = s)$ and the counterfactual intervention $\mathrm{do}(S = s')$.

Other authors (Nabi & Shpitser, 2018; Chiappa, 2019) have found the definition of Equation (1) too restrictive and have instead considered path-specific counterfactual inference where the sensitive variable may affect the decision via a fair pathway. For instance, even if gender affects education, it would be fair to use education in a recruitment decision. However, it would be discriminatory to base the recruitment decision on gender or its proxies such as the given name. The recognition of fair and unfair pathways requires an understanding of the underlying causal mechanisms. Along these lines, we will use the following definition.

Definition 4 (Counterfactual fairness)**.** *Let* $(\mathbf{V}, \mathbf{U}, \mathbf{F}, p(\mathbf{u}))$ *be an SCM where* $\mathbf{S} \subset \mathbf{V}$ *is the set of sensitive variables and* $\mathbf{Y} \subset \mathbf{V}$ *is the set of outcome variables. Define* $\mathbf{W} = \mathrm{Pa}(\mathbf{Y}) \setminus \mathbf{S}$ *and* $\mathbf{Z} = \mathbf{V} \setminus (\mathbf{Y} \cup \mathbf{W} \cup \mathbf{S})$*, and let the conditioning variables be* $\mathbf{C} = \mathbf{W} \cup \mathbf{Z} \cup \mathbf{S}$*. The decisions based on an estimator* $\widehat{\mathbf{Y}}$ *are counterfactually fair if*

$$p(\widehat{\mathbf{Y}}_{\mathrm{do}(\mathbf{S}=\mathbf{s}, \mathbf{W}=\mathbf{w})} \mid \mathbf{W} = \mathbf{w}, \mathbf{Z} = \mathbf{z}, \mathbf{S} = \mathbf{s}) = p(\widehat{\mathbf{Y}}_{\mathrm{do}(\mathbf{S}=\mathbf{s}', \mathbf{W}=\mathbf{w})} \mid \mathbf{W} = \mathbf{w}, \mathbf{Z} = \mathbf{z}, \mathbf{S} = \mathbf{s})$$

under any context $\mathbf{W} = \mathbf{w}$*,* $\mathbf{Z} = \mathbf{z}$*, and for any values* $\mathbf{s}$ *and* $\mathbf{s}'$ *of* $\mathbf{S}$*.*

Definition 4 fixes the non-sensitive parents $\mathbf{W}$ of the outcome to their observed values under any counterfactual intervention of the sensitive variables. In the recruitment example, this would mean that a counterfactual intervention on gender could change the given name, but education would remain unchanged because it has a direct effect on job performance. This kind of definition is mentioned by Wu et al. (2019) who call it "individual indirect discrimination".

3. Algorithms

We proceed to construct an algorithmic framework for counterfactual simulation and fairness evaluation. The general principle of the counterfactual simulation is straightforward: the proposed simulation algorithm randomly generates the values of some background variables and numerically solves the rest of the background variables from the conditions. The obtained candidate observations are iteratively resampled with unequal probabilities so that the selected sample is an approximate random sample from the conditional distribution of the background variables. The counterfactual distribution is approximated using this sample of background variables together with the intervened SCM.

This section is organized as follows. First, the assumptions and some basic results are presented in Section 3.1. Next, the formal algorithms are presented in Sections 3.2 and 3.3. Finally, in Section 3.4 we show that the simulation algorithm leads to asymptotically consistent inference and can be interpreted as a sequential Monte Carlo algorithm.

3.1 Assumptions and Basic Results

The starting point of counterfactual simulation is an SCM $\mathcal{M} = (\mathbf{V}, \mathbf{U}, \mathbf{F}, p(\mathbf{u}))$ where the functions $\mathbf{F}$ and the distribution $p(\mathbf{u})$ are fully known, and the objective is to simulate values from counterfactual distribution $p(\mathbf{W}_{\mathrm{do}(\mathbf{X}=\mathbf{x})} = \mathbf{w} \,|\, \mathbf{C} = \mathbf{c})$. In order to simulate counterfactuals in cases where the set of conditioning variables $\mathbf{C}$ contains continuous variables, we refine the definition of the SCM by imposing some restrictions on background variables. First, we define a special type of background variable:

Definition 5 (Dedicated error term)**.** *In an SCM $\mathcal{M} = (\mathbf{V}, \mathbf{U}, \mathbf{F}, p(\mathbf{u}))$, a background variable $U \in \mathbf{U}$ is a dedicated error term if it is independent of the other background variables and has exactly one child. If an observed variable $V \in \mathbf{V}$ has only one dedicated error term, it is denoted by $\overline{U}_V$, and the notation $\xi_V(u) = p_{\overline{U}_V}(u)$ is used for the density function of $\overline{U}_V$.*

With this definition, the background variables can be divided into two categories, $\overline{\mathbf{U}}$ and $\widetilde{\mathbf{U}}$, where $\overline{\mathbf{U}}$ is the set of dedicated error terms and $\widetilde{\mathbf{U}}$ is the set of other background variables which we call global background variables henceforth, and $\mathbf{U} = \overline{\mathbf{U}} \cup \widetilde{\mathbf{U}}$. Further, let $\overline{\mathbf{U}}_{\mathbf{C}}$ and $\overline{\mathbf{U}}_{\mathbf{V} \setminus \mathbf{C}}$ represent the dedicated error terms of variables in $\mathbf{C}$ and in $\mathbf{V} \setminus \mathbf{C}$, respectively. From the independence of dedicated error terms, it follows that

$$p(\overline{\mathbf{u}}) = \prod_{U \in \overline{\mathbf{U}}} p_U(u).$$

The variables $\widetilde{\mathbf{U}}$ act as unobserved confounders while each dedicated error term only affects the variation of one specific observed variable. We will often consider the dedicated error terms separately from other parents of a variable and for this purpose, we introduce the notation $\mathrm{Pa}^\circ(V) = \mathrm{Pa}^*(V) \cap (\mathbf{V} \cup \widetilde{\mathbf{U}}) = \mathrm{Pa}^*(V) \setminus \{\overline{U}_V\}$.

Our SCMs of interest have dedicated error terms with a monotonic effect in their respective functions.

Definition 6 (u-monotonic SCM)**.** *A recursive SCM $\mathcal{M} = (\mathbf{V}, \mathbf{U}, \mathbf{F}, p(\mathbf{u}))$ is u-monotonic with respect to a set of continuous variables $\mathbf{W} \subseteq \mathbf{V}$ if it has the following properties:*

(a) Each $V \in \mathbf{V}$ is univariate and has exactly one dedicated error term that is a continuous univariate random variable.

(b) For all $W \in \mathbf{W}$, it holds that given $\mathrm{Pa}^\circ(W)$, the value of W is determined as $w = g_W(\overline{u}_w) = g_W(\overline{u}_w; \mathrm{Pa}^\circ(w)) = f_W(\overline{u}_w, \mathrm{Pa}^\circ(w))$ where g_W is a continuous, differentiable and strictly increasing function of dedicated error term $\overline{u}_w$, and the parameters of the function are $\mathrm{Pa}^\circ(W)$.

The inverses (in their domain) and derivatives of the function g_W exist and they are denoted by g_W^{-1} and g'_W, respectively. Cases where g_W would be a strictly decreasing function can be accommodated to Definition 6 by redefining $\overline{u}_w^{\mathrm{new}} = -\overline{u}_w$ and $g_W^{\mathrm{new}}(u) = g_W(-u)$. The second property of Definition 6 is trivially fulfilled in the important special case, the additive noise model

$$w = f_W(\overline{u}_w, \mathrm{Pa}^\circ(w)) = f_W^*(\mathrm{Pa}^\circ(w)) + \overline{u}_w,$$

where f_W^* is some function that does not depend on $\overline{u}_w$. The formulation of Definition 6 permits $g_W(\overline{u}_w; \mathrm{Pa}^\circ(w))$ to be a complicated function, which allows for heteroscedastic noise models.

The first step of Definition 2 involves deriving the conditional distribution $p(\mathbf{U} = \mathbf{u} \,|\, \mathbf{C} = \mathbf{c})$. The following results characterize conditional distributions in a u-monotonic SCM.

Lemma 1. *Let W be a continuous variable in a u-monotonic SCM with respect to a set $\mathbf{W}$ such that $W \in \mathbf{W}$. The conditional density of $W \,|\, (\mathrm{Pa}^\circ(W) = \mathrm{Pa}^\circ(w))$ is*

$$p\big(w \,|\, \mathrm{Pa}^\circ(w)\big) = \begin{cases} \dfrac{\xi_W(\overline{u}_{W=w})}{g'_W(\overline{u}_{W=w}; \mathrm{Pa}^\circ(w))}, & \text{if } a < w < b, \\ 0, & \text{otherwise,} \end{cases}$$

where (a, b) is the domain of $g_W(\,\cdot\,; \mathrm{Pa}^\circ(w))$ and $\overline{u}_{W=w} = g_W^{-1}(w; \mathrm{Pa}^\circ(w))$. The notation $\omega_W(\overline{u}_{W=w}) = p\big(w \,|\, \mathrm{Pa}^\circ(w)\big)$ is used to assign a weight for $\overline{u}_{W=w}$.

Proof. The conditional density of $\overline{U}_W \mid \mathrm{Pa}^\circ(W)$ is, by independence, ξ_W, the unconditional density of $\overline{U}_W$. The conditional distribution $W \mid \mathrm{Pa}^\circ(W)$ is a point mass at $g_W(\overline{U}_W; \mathrm{Pa}^\circ(W))$. Therefore, by the standard transformation formula, W has the density

$$p_W(g^{-1}(w))(g^{-1})'(w) = \frac{\xi_W(g^{-1}(w))}{g'(g^{-1}(w))},$$

for all w in its domain, and zero elsewhere. □

Lemma 1 tells that the value of the dedicated error term $\overline{u}_{W=w}$ is determined by the value w and the values of other parents of W via the function $g_W^{-1}(w; \mathrm{Pa}^\circ(w))$. The parents $\mathrm{Pa}^\circ(w)$ may include global background variables $\widetilde{\mathbf{U}}$. When the parents $\mathrm{Pa}^\circ(w)$ are fixed, the distribution of W is obtained from the distribution $\overline{u}_{W=w}$ by the standard transformation formula. Note that for an additive noise model, we simply have $\omega_W(\overline{u}_{W=w}) = \xi_W(\overline{u}_{W=w})$. The next lemma describes the conditional distribution of the background variables.

Lemma 2. *In a u-monotonic SCM, the conditional density of $\widetilde{\mathbf{U}}, \mathbf{U}_{\mathbf{V} \setminus \mathbf{C}} \,|\, (\mathbf{C} = \mathbf{c})$ satisfies:*

$$p(\widetilde{\mathbf{u}}, \mathbf{u}_{\mathbf{V}\setminus\mathbf{C}} \,|\, \mathbf{c}) \propto p(\widetilde{\mathbf{u}}) \left(\prod_{U \in \overline{\mathbf{U}}_{\mathbf{V}\setminus\mathbf{C}}} p(u) \right) \left(\prod_{C \in \mathbf{C}} \omega_C(\overline{u}_{C=c}) \right),$$

where terms $\omega_C(\overline{u}_{C=c}) = p\big(c \,|\, \mathrm{Pa}^\circ(c)\big)$ in the last product are defined in Lemma 1.

Proof. The expression on the right is the joint density of $(\widetilde{\mathbf{U}}, \overline{\mathbf{U}}_{\mathbf{V}\setminus\mathbf{C}}, \mathbf{C})$, written by the chain rule:

$$p(\widetilde{\mathbf{u}}) \left(\prod_{U \in \overline{\mathbf{U}}_{\mathbf{V}\setminus\mathbf{C}}} p(u \,|\, \widetilde{\mathbf{u}}, \tau(u) \cap \mathbf{u}_{\mathbf{V}\setminus\mathbf{C}}) \right) \left(\prod_{C \in \mathbf{C}} p(c \,|\, \widetilde{\mathbf{u}}, \overline{\mathbf{u}}_{\mathbf{V}\setminus\mathbf{C}}, \tau(c) \cap \mathbf{c}) \right).$$

The claim follows by applying the mutual independence of dedicated error terms to the first product and Lemma 1 to the second product. □

Combining the results above, the conditional distribution for all background variables can be written as follows:

Corollary 1. *The conditional distribution of* $(\widetilde{\mathbf{U}}, \overline{\mathbf{U}}, \mathbf{Y}) \,|\, (\mathbf{C} = \mathbf{c})$*, where* $\mathbf{Y} = \mathbf{V} \setminus \mathbf{C}$*, is:*

$$p(\widetilde{\mathbf{u}}, \overline{\mathbf{u}}_{\mathbf{V}\setminus\mathbf{C}} \,|\, \mathbf{c})\, \mathrm{d}\widetilde{\mathbf{u}}\, \mathrm{d}\overline{\mathbf{u}}_{\mathbf{V}\setminus\mathbf{C}} \left(\prod_{C\in\mathbf{C}} I(g_C^{-1}(c; \mathrm{Pa}^\circ(c)) \in \mathrm{d}u_c)\right)\left(\prod_{Y\in\mathbf{Y}} I(g_Y(u_Y; \mathrm{Pa}^\circ(y)) \in \mathrm{d}y)\right),$$

where $I(\cdot)$ *is an indicator function. In other words,* $\widetilde{\mathbf{u}}$ *and* $\overline{\mathbf{u}}_{\mathbf{V}\setminus\mathbf{C}}$ *have a density, and* $\overline{\mathbf{u}}_{\mathbf{C}}$ *and* $\mathbf{y}$ *depend on them deterministically as above.*

3.2 Algorithms for Conditional Simulation

Next, we will consider algorithms that simulate observations from a u-monotonic SCM under given conditions (step 1 of Definition 2). First, we will present simulation algorithms for the cases with a single conditioning variable. Algorithm 1 considers the case of a continuous conditioning variable and Algorithm 2 the case of a discrete conditioning variable. Algorithm 3 processes multiple conditioning variables and calls Algorithms 1 and 2. The full workflow of counterfactual inference is implemented later in Algorithm 4. Algorithm 5 applies Algorithm 4 in fairness evaluation.

3.2.1 Continuous Condition

We describe the operation of Algorithm 1. On lines 3 and 5, the procedure SimulateSCM is called. The unconditional call on line 5 first simulates n realizations of the background variables from $p(\mathbf{u})$ and then applies functions $\mathbf{F}$ to obtain the values of the observed variables $\mathbf{V}$. The conditional version on line 3 is similar, but the values of the variables in $\mathbf{D}_0$ are not generated but taken directly from $\mathbf{D}_0$. As a result of line 3 or line 5, the data matrix $\mathbf{D}$ contains n rows and the values of all variables in $\mathbf{V}$ and $\mathbf{U}$. In practice, it is not necessary to simulate the descendants of $\overline{U}_C$ on lines 3 or 5 because these variables will be updated later.

Algorithm 1 An algorithm for simulating n observations from a u-monotonic SCM $\mathcal{M} = (\mathbf{V}, \mathbf{U}, \mathbf{F}, p(\mathbf{u}))$ on the condition that the value of a continuous variable C is c. The optional argument $\mathbf{D}_0$ is an n-row data matrix containing the values of some variables $\mathbf{V}_0 \subset \mathbf{V}$, $\mathbf{U}_0 \subset \mathbf{U}$ that precede C in the topological order and have already been fixed.

```
1: function SimulateContinuousCondition($n$, $\mathcal{M}$, $C = c$, $\mathbf{D}_0$)
2:     if $\mathbf{D}_0$ exists then
3:         $\mathbf{D} \leftarrow$ SimulateSCM($n$, $\mathcal{M}$, $\mathbf{D}_0$)
4:     else
5:         $\mathbf{D} \leftarrow$ SimulateSCM($n$, $\mathcal{M}$, $\emptyset$)
6:     for $i = 1$ to $n$ do
7:         $\mathbf{D}[i, \overline{U}_C] \leftarrow$ FindRoot($\mathcal{M}$, $\mathbf{D}[i,]$, $\overline{U}_C$, $C$, $c$)
8:         $\boldsymbol{\omega}[i] \leftarrow \omega_C(\mathbf{D}[i, \overline{U}_C])$
9:     $\mathbf{D} \leftarrow$ Sample($\mathbf{D}$, $n$, $\boldsymbol{\omega}$)
10:    $\mathbf{D} \leftarrow$ SimulateSCM($n$, $\mathcal{M}$, $\mathbf{D}[, \mathbf{U}]$)
11:    return $\mathbf{D}$
```

Starting from line 6, the values of the error term $\overline{U}_C$ in the data matrix $\mathbf{D}$ are modified so that the condition $C = c$ is fulfilled. The procedure FindRoot on line 7 uses binary search to find the value u_i of $\overline{U}_C$ so that

$$f_C(u_i, \mathbf{D}[i, \mathrm{Pa}^\circ(C)]) = c, \tag{2}$$

where $f_C \in \mathbf{F}$ is the function that determines the value C in the u-monotonic SCM $\mathcal{M}$. Due to the monotonicity assumption, there is at most one value u_i that solves Equation (2), and if the solution exists it can be found by binary search. If a solution cannot be found, $\mathbf{D}[i, \overline{U}_C]$ is set as "not available". The binary search is not needed for additive noise models, where u_i can be solved analytically. On line 8, the sampling weights are calculated with function ω_C defined in Lemma 1. If $\mathbf{D}[i, \overline{U}_C]$ is not available, $\boldsymbol{\omega}[i]$ will be set as 0.

On line 9, a re-sample is drawn with the weights $\boldsymbol{\omega} = (\boldsymbol{\omega}[1], \ldots, \boldsymbol{\omega}[n])$ (it is assumed here that at least one of the weights is positive). The sampling is carried out with replacement which leads to conditionally independent but non-unique observations. Uniqueness could be achieved by adding low-variance noise to the background variables before line 10 but this would lead to a small divergence from the required condition $C = c$. On line 10, the observed variables of $\mathcal{M}$ are updated because the change in $\overline{U}_C$ affects its descendants.

3.2.2 Discrete Condition

Next, we consider the case of a single discrete conditioning variable and describe the operation of Algorithm 2. Lines 3 and 5 are similar to lines 3 and 5 of Algorithm 1, respectively. On line 6, n observations are sampled with replacement from those observations that fulfill the target condition $C = c$.

Algorithm 2 An algorithm for simulating n observations from causal model $\mathcal{M} = (\mathbf{V}, \mathbf{U}, \mathbf{F}, p(\mathbf{u}))$ on the condition that the value of discrete variable C is c. The optional argument $\mathbf{D}_0$ is an n-row data matrix containing the values of some variables $\mathbf{V}_0 \subset \mathbf{V}$, $\mathbf{U}_0 \subset \mathbf{U}$ that precede C in the topological order and have been already fixed.

```
1: function SimulateDiscreteCondition($n$, $\mathcal{M}$, $C = c$, $\mathbf{D}_0$)
2:     if $\mathbf{D}_0$ exists then
3:         $\mathbf{D} \leftarrow$ SimulateSCM($n$, $\mathcal{M}$, $\mathbf{D}_0$)
4:     else
5:         $\mathbf{D} \leftarrow$ SimulateSCM($n$, $\mathcal{M}$, $\emptyset$)
6:     $\mathbf{D} \leftarrow$ Sample($\mathbf{D}[C = c, ]$, $n$)
7:     return $\mathbf{D}$
```

3.2.3 Multiple Simultaneous Discrete and Continuous Conditions

When multiple conditions are present in the counterfactual, sequential calls of Algorithms 1 and 2 are needed. Algorithm 3 presents the required steps where the type of the conditioning variable decides whether Algorithm SimulateContinuousCondition or SimulateDiscreteCondition is called. On lines 3 or 5, the data are simulated according to the condition that is the first in the topological order. Starting from line 6, the remaining conditions are processed in the topological order. On line 7, the set $\mathbf{S}$ contains variables that have already been processed. On lines 9 or 11, the data are simulated according to each condition in the topological order taking into account that the variables in $\mathbf{S}$ have been already fixed in $\mathbf{D}$.

Algorithm 3 An algorithm for simulating n observations from a u-monotonic SMC $\mathcal{M} = (\mathbf{V}, \mathbf{U}, \mathbf{F}, p(\mathbf{u}))$ with respect to all continuous variables in $C_1, \ldots, C_K$ under the conditions $\mathcal{C} = (C_1 = c_1) \wedge \cdots \wedge (C_K = c_K)$. The topological order of the variables in the conditions is $C_1 < C_2 < \cdots < C_K$. The batch size $n^* = n$ is used in Algorithm 2.

```
1:  function SimulateMultipleConditions($n$, $\mathcal{M}$, $\mathcal{C}$)
2:      if $C_1$ is continuous then
3:          $\mathbf{D} \leftarrow$ SimulateContinuousCondition($n$, $\mathcal{M}$, $C_1 = c_1$)
4:      else
5:          $\mathbf{D} \leftarrow$ SimulateDiscreteCondition($n$, $\mathcal{M}$, $C_1 = c_1$, $n$)
6:      for $k = 2$ to $K$ do
7:          $\mathbf{S} \leftarrow (C_1 \cup \mathrm{An}^*(C_1)) \cup \cdots \cup (C_{k-1} \cup \mathrm{An}^*(C_{k-1}))$
8:          if $C_k$ is continuous then
9:              $\mathbf{D} \leftarrow$ SimulateContinuousCondition($n$, $\mathcal{M}$, $C_k = c_k$, $\mathbf{D}[, \mathbf{S}]$)
10:         else
11:             $\mathbf{D} \leftarrow$ SimulateDiscreteCondition($n$, $\mathcal{M}$, $C_k = c_k$, $n$, $\mathbf{D}[, \mathbf{S}]$)
12:     return $\mathbf{D}$
```

3.3 Algorithms for Counterfactual Inference and Fairness

Next, we present high-level algorithms that use Algorithm 3 to simulate data from a multivariate conditional distribution. Algorithm 4 simulates observations from a counterfactual distribution. The input consists of a u-monotonic SCM, conditions that define the situation considered, an intervention to be applied, and the number of observations. The algorithm has three steps that correspond to the three steps of the evaluation of counterfactuals in Definition 2. On line 2, data are simulated using Algorithm 3. On line 3, an intervention $\mathrm{do}(\mathbf{X} = \mathbf{x})$ is applied. In practice, the functions of variables $\mathbf{X}$ in the SCM are replaced by constant-valued functions. On line 4, counterfactual observations are simulated from the intervened SCM with the background variables simulated on line 2.

Algorithm 4 An algorithm for simulating n observations from a counterfactual distribution under the conditions $\mathcal{C} = (C_1 = c_1) \wedge \cdots \wedge (C_K = c_K)$ in an SCM $\mathcal{M} = (\mathbf{V}, \mathbf{U}, \mathbf{F}, p(\mathbf{u}))$ that is u-monotonic with respect to all continuous variables in the set $\{C_1, \ldots, C_K\}$. The topological order of the variables in the conditions is $C_1 < C_2 < \cdots < C_K$.

```
1:  function SimulateCounterfactual($\mathcal{M}$, $\mathcal{C}$, do($\mathbf{X} = \mathbf{x}$), $n$)
2:      $\mathbf{D}_0 \leftarrow$ SimulateMultipleConditions($n$, $\mathcal{M}$, $\mathcal{C}$)
3:      $\mathcal{M}_{\mathbf{x}} \leftarrow$ Intervene($\mathcal{M}$, do($\mathbf{X} = \mathbf{x}$))
4:      $\mathbf{D} \leftarrow$ SimulateSCM($n$, $\mathcal{M}_{\mathbf{x}}$, $\mathbf{D}_0[, \mathbf{U}]$)
5:      return $\mathbf{D}$
```

Algorithm 5 evaluates the fairness of a prediction model based on Definition 4. The input consists of a prediction model, an SCM, a set of sensitive variables, a case to be evaluated, and the size of the data to be simulated. The prediction model can be an opaque AI model where the user is unaware of the functional details of the model and has only access to the predictions.

Algorithm 5 An algorithm for evaluating the fairness of a prediction model $\widehat{Y}(\cdot)$ in a u-monotonic SCM $\mathcal{M}$ with sensitive variables $\mathbf{S}$ and response variables $\mathbf{Y}$. The case to be considered is defined by conditions $\mathcal{W}$ and $\mathcal{C}$ where $\mathcal{W} = (W_1 = w_1) \wedge \cdots \wedge (W_M = w_M)$ denotes the conditions for $\mathrm{Pa}(\mathbf{Y}) \setminus \mathbf{S}$ (the non-sensitive observed parents of the responses $\mathbf{Y}$), and $\mathcal{C} = (C_1 = c_1) \wedge \cdots \wedge (C_K = c_K)$ denotes the conditions for some other variables (which may include $\mathbf{S}$). The argument n defines the number of simulated counterfactual observations.

```
1: function EvaluateFairness($\widehat{Y}(\cdot)$, $\mathcal{M}$, $\mathbf{S}$, $\mathcal{W} \wedge \mathcal{C}$, $n$)
2:     for all $\mathbf{s}$ in dom($\mathbf{S}$) do
3:         $\mathbf{D_s} \leftarrow$ SimulateCounterfactual($n$, $\mathcal{M}$, $\mathcal{W} \wedge \mathcal{C}$, do($\mathbf{S} = \mathbf{s}, \mathcal{W}$))
4:         $\widehat{\mathbf{Y}}_\mathbf{s} \leftarrow \widehat{Y}(\mathbf{D_s})$
5:     return CheckResponse($\{\widehat{\mathbf{Y}}_\mathbf{s}\}$)
```

Algorithm 5 loops over the possible values of the sensitive variables (line 2). If some sensitive variables are continuous, the values to be considered must be specified by the user. On line 3, data are simulated for the intervention $\mathrm{do}(\mathbf{S} = \mathbf{s}, \mathcal{W})$ that intervenes the sensitive variables but keeps the parents (outside of $\mathbf{S}$) of the responses fixed to their original values. On line 4, the prediction model is applied to the simulated data. On line 5, the fairness of the obtained predictions is evaluated, i.e., it is checked that all the predictions are the same regardless of the intervention applied to the sensitive variables. Algorithm 5 is usually called many times with different conditions $\mathcal{C}$. Algorithms 1–5 are implemented in the R package `R6causal` (Karvanen, 2024) which contains `R6` (Chang, 2021) classes and methods for SCMs.

Algorithms for alternative counterfactual definitions of fairness could be implemented similarly. For instance, the definition given in Equation (1) emerges if the intervention $\mathrm{do}(\mathbf{S} = \mathbf{s}, \mathcal{W})$ is replaced by the intervention $\mathrm{do}(\mathbf{S} = \mathbf{s})$ in Algorithm 5.

3.4 Conditional Simulation as a Particle Filter

We show that Algorithm 3 can be interpreted as a particle filter. This interpretation allows us to study the theoretical properties of the simulated sample. The notation used in this section is conventional for particle filters and differs from the rest of the paper. More specifically, we assume without loss of generality the topological order $V_1 < V_2 < \cdots < V_J$ which implies $\mathrm{Pa}(V_j) \subseteq \{V_1, \ldots, V_{j-1}\}$. We also assume that $\overline{U}_j$, $j = 1, \ldots, J$ is the dedicated error term of V_j. We introduce a shortcut notation $V_{1:j} = \{V_1, \ldots, V_j\}$ and use V_j^i to denote the ith observation of V_j. We will first review a basic particle filter algorithm and then show how it corresponds to Algorithm 3.

Particle filters are sequential Monte Carlo algorithms, which use sequentially defined 'proposal distributions' consisting of the initial distribution $M_0(\mathrm{d}z_0)$ on Z_0 and transition probabilities $M_j(\mathrm{d}z_j \mid z_{1:j-1})$ to Z_j, and non-negative 'potential functions' $G_j(z_{0:j})$, where j is the time index (cf. Del Moral, 2004). Algorithm 6 summarizes the basic particle filter algorithm, which outputs (approximate and dependent) samples from the probability distribution $\pi_J(B) = \gamma_J(B)/\gamma_J(\mathsf{Z}_J)$, where $B \subseteq \mathsf{Z}_J$ and the unnormalized 'smoothing

Algorithm 6 PARTICLEFILTER($M_{0:J}, G_{1:J}, n$)

```
1: Draw $Z_0^i \sim M_0(\,\cdot\,)$ and set $\mathbf{Z}_0^i = Z_0^i$ for $i = 1{:}n$
2: for $j = 1, \dots, J$ do
3:     Draw $Z_j^i \sim M_j(\,\cdot\mid \mathbf{Z}_{j-1}^i)$ for $i = 1{:}n$
4:     Calculate weights $W_j^i = \frac{\widetilde{W}_j^i}{\sum_{j=1}^n \widetilde{W}_j^j}$ where $\widetilde{W}_j^i = G_j(\mathbf{Z}_{j-1}^i, Z_j^i)$ for $i = 1{:}n$
5:     Draw $A_j^{1:n} \sim \mathrm{Categorical}(W_j^{1:n})$ and set $\mathbf{Z}_j^i = (\mathbf{Z}_{j-1}^{A_j^i}, Z_j^i)$ for $i = 1{:}n$
6: output $\mathbf{Z}_J^{1:n}$
```

distribution' γ_J is defined as follows:

$$\gamma_J(B) = \int M_0(\mathrm{d}z_0) \prod_{j=1}^{J} M_j(\mathrm{d}z_j \mid z_{1:j-1}) G_j(z_{0:j}) I(z_{0:J} \in B).$$

For the reader's convenience, we state a mean square error convergence and central limit theorem for particle filter estimates (Chopin & Papaspiliopoulos, 2020, Propositions 11.2 and 11.3).

Theorem 2. *Assume that the potential functions are bounded: $\|G_j\|_\infty = \sup_{\mathbf{z}} G_j(\mathbf{z}) < \infty$. Then, there exist constants $b, \sigma_h < \infty$ such that for any bounded test function $h : \mathsf{Z}_0 \times \cdots \times \mathsf{Z}_J \to \mathbb{R}$ the output of Algorithm 6 satisfies:*

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} h(\mathbf{Z}_J^i) - \pi_J(h)\right]^2 \le b\frac{\|h\|_\infty^2}{n}, \qquad \text{for all } n \ge 1,$$

$$\frac{1}{\sqrt{n}}\sum_{i=1}^{n}\left[h(\mathbf{Z}_J^i) - \pi_J(h)\right] \xrightarrow{n\to\infty} N(0, \sigma_h^2), \qquad \text{in distribution,}$$

where $\pi_J(h)$ stands for the expected value of $h(\mathbf{X})$ where $\mathbf{X}$ follows distribution π_J.

Theorem 2 is stated for bounded test functions only. It is possible to prove similar results for unbounded functions (e.g., Cappé et al., 2005; Rohrbach & Jack, 2022, and references therein).

In order to cast Algorithm 3 as a particle filter, we need the following ingredients:

(a) Initial particle states contain the global background variables $Z_0 = \widetilde{\mathbf{U}}$,

(b) Initial distribution $M_0(\mathrm{d}\widetilde{\mathbf{u}}) = p(\widetilde{\mathbf{u}})\mathrm{d}\widetilde{\mathbf{u}}$ on $\mathsf{Z}_0 = \mathrm{dom}(\widetilde{\mathbf{U}})$,

(c) The particle states for $j \ge 1$ contain the observed variables and their dedicated error terms $Z_j = (V_j, \overline{U}_j)$,

(d) If $V_j \in \mathbf{C}$ then the value of V_j is set to be v_j,

(e) Proposal distributions on $\mathsf{Z}_j = \mathbb{R}^2$:

(i) If V_j is continuous and $V_j \notin \mathbf{C}$:

$$M_j(\mathrm{d}v_j, \mathrm{d}\overline{u}_j \,|\, \widetilde{\mathbf{u}}, \overline{\mathbf{u}}_{1:j-1}, \mathbf{v}_{1:j-1}) = p(\overline{u}_j)\mathrm{d}\overline{u}_j \, I(g_{V_j}(\overline{u}_j; \mathrm{Pa}^\circ(v_j)) \in \mathrm{d}v_j),$$

(ii) If V_j is continuous and $V_j \in \mathbf{C}$:

$$M_j(\mathrm{d}v_j, \mathrm{d}\overline{u}_j \,|\, \widetilde{\mathbf{u}}, \overline{\mathbf{u}}_{1:j-1}, \mathbf{v}_{1:j-1}) = I(g^{-1}_{V_j}(v_j; \mathrm{Pa}^\circ(v_j)) \in \mathrm{d}u_j) I(v_j \in \mathrm{d}v_j),$$

(iii) If V_j is discrete:

$$M_j(\mathrm{d}v_j, \mathrm{d}\overline{u}_j \,|\, \widetilde{\mathbf{u}}, \overline{\mathbf{u}}_{1:j-1}, \mathbf{v}_{1:j-1}) = p(\overline{u}_j)\mathrm{d}\overline{u}_j,$$

(f) Potentials:

(i) If $V_j \notin \mathbf{C}$: $G_j \equiv 1$,

(ii) If $V_j \in \mathbf{C}$ and V_j is continuous:

$$G_j(\widetilde{\mathbf{u}}, \overline{\mathbf{u}}_{1:j}, \mathbf{v}_{1:j}) = \begin{cases} \dfrac{\xi_{V_j}(u_j)}{g'_{V_j}(u_j; \mathrm{Pa}^\circ(v_j))}, & \text{if } v_j \in \mathrm{dom}\big(g_{V_j}(u_j; \mathrm{Pa}^\circ(v_j))\big), \\ 0, & \text{otherwise.} \end{cases}$$

(iii) If $V_j \in \mathbf{C}$ and V_j is discrete:

$$G_j(\widetilde{\mathbf{u}}, \overline{\mathbf{u}}_{1:j}, \mathbf{v}_{1:j}) = \begin{cases} 1, & \text{if } I(v_j \in \mathrm{d}v_j), \\ 0, & \text{otherwise.} \end{cases}$$

We conclude that Algorithm 3 is a particle filter algorithm and therefore Theorem 2 is applicable:

Corollary 2. *Let $\mathcal{M} = (\mathbf{V}, \mathbf{U}, \mathbf{F}, p(\mathbf{u}))$ be a u-monotonic SCM and obtain a sample $\mathbf{D}$ by calling $\mathbf{D} = \textsc{SimulateMultipleConditions}(n, \mathcal{M}, \mathcal{C})$, where $\mathcal{C}$ is a set of conditions on $\mathbf{C} \subset \mathbf{V}$. Further, let $h : \mathrm{dom}(\mathbf{V}) \to \mathbb{R}$ be a bounded function and let π_J be the conditional distribution of $\mathbf{V}$ given $\mathcal{C}$. If the conditional density of $V_j \,|\, \mathrm{Pa}^\circ(V_j) = \mathrm{Pa}^\circ(v_j)$ is bounded for all $V_j \in \mathbf{C}$ then the mean square error and central limit theorem as in Theorem 2 hold for $h(\mathbf{D})$ and π_J.*

Proof. According to Theorem 2, the potential functions must be bounded. The potential functions for Algorithm 1 are $\xi_{V_j}(u_j)/g'_{V_j}(u_j; \mathrm{Pa}^\circ(v_j))$ which according to Lemma 1 give the conditional density of $V_j \,|\, \mathrm{Pa}^\circ(V_j) = \mathrm{Pa}^\circ(v_j)$. $\square$

We conclude the section with some remarks about possible extensions and elaborations of the algorithm. Algorithm 6 is similar to the seminal algorithm by Gordon et al. (1993), which can be elaborated in a number of ways so that the algorithm remains (asymptotically) valid. The resampling indices $A_j^{1:n}$ in step 5 need not be independent, as long as they form an unbiased 'sample' from the categorical distribution (e.g., Douc & Cappé, 2005; Chopin, Singh, Soto, & Vihola, 2022, and references therein). Furthermore, resampling can be (sometimes) omitted, and replaced by cumulation of the weights, and this can be done in an adaptive fashion (Liu & Chen, 1995). As a special case of adaptive resampling, we could omit the resampling on line 9 of Algorithm 1 entirely and return a weighted sample instead. In Algorithm 3, the weights would be multiplied after processing each condition and the resampling would be carried out only as the final step. However, this approach may not work well with multiple conditions because, on line 9 of Algorithm 3, the current data $\mathbf{D}[,\mathbf{S}]$ would mostly contain observations with low weights. Resampling after each condition makes sure that new data are simulated only for relevant cases.

The particle filter is known to be often efficient, if we are interested in the variables Z_J only, or $Z_{J-\ell}$ for moderate lag ℓ. However, as we care also about Z_0, large J often causes the samples to contain only a few or one unique value for Z_0. This is a well-known problem, which is referred to as 'sample degeneracy' (Li, Sun, Sattar, & Corchado, 2014). Algorithm 3 remains effective for an SCM with a high total number of variables, as long as the number of conditioning variables J remains moderate.

4. Simulations

The critical part of Algorithm 4 is the quality of the sample returned by Algorithm 3. In this section, the performance of Algorithm 3 is studied using randomly generated linear Gaussian SCMs. Importantly, we can analytically derive the true distribution for comparison in this particular scenario, see Online Appendix 2 for the details.

In the simulation experiment, we generate linear Gaussian SCMs with random graph structure and random coefficients, apply Algorithm 3, and compare the simulated observations with the true conditional normal distribution. The parameters of the simulation and the performance measures are explained in Online Appendix 3.

The key results are summarized in Table 1. The sample sizes in the simulation were 1000, 10000, 100000, and 1000000. For each setting, there were 1000 simulation rounds. Case A with five observed variables and one condition is expected to be an easy task. Cases B and C with ten observed variables represent moderate difficulty, and cases D and E with fifty observed variables are expected to be challenging.

It can be seen that the percentage of unique observations decreases when the setting becomes more challenging. The repeated observations originate from the resampling on line 9 of Algorithm 1. In all cases, $\overline{\overline{z}}$, the mean of sample means was unbiased, and the variation of sample means between simulation rounds decreased as the sample size increased. The mean of sample standard deviations was close to the true value 1 in all cases except in case E where the variation was too small in average. This was also reflected in the Kolmogorov–Smirnov statistics that indicated major differences from the true distribution. The correlation coefficients were unbiased or almost unbiased in all cases. The average runtime per a simulation round in case E with $n = 1000000$ was 121 seconds. The simulation code is available at `https://github.com/JuhaKarvanen/simulating_counterfactuals`.

5. Application to Fairness in Credit-Scoring

In this section, we show how Algorithm 5 can be used in the fairness analysis of a synthetic scenario where the details of the prediction model are unknown and real data are not available. We consider credit-scoring where the decision to grant a bank loan to a consumer

Simulation cases and their parameters

Case	$\|\mathbf{V}\|$	conditions	mean neighbors	mean $\|\widetilde{\mathbf{U}}\|/\|\mathbf{V}\|$
A	5	1	3	0
B	10	4	5	1
C	10	9	5	1
D	50	2	5	1
E	50	9	7	1

Simulation results

Case	n	uniq. %	$\overline{\overline{z}}$	$\overline{z}$: min,max	$\overline{s}_z$	s_z: min,max	K-S	diff. cor.
A	1000	49	0.00	$(-0.32, 0.49)$	1.00	$(0.00, 1.13)$	0.05	0.00
A	10000	49	-0.00	$(-0.84, 0.09)$	1.00	$(0.74, 1.10)$	0.02	0.00
A	100000	49	0.00	$(-0.02, 0.04)$	1.00	$(0.97, 1.03)$	0.00	-0.00
A	1000000	49	0.00	$(-0.02, 0.01)$	1.00	$(0.99, 1.01)$	0.00	-0.00
B	1000	17	-0.01	$(-3.01, 2.13)$	0.95	$(0.00, 1.56)$	0.17	0.00
B	10000	18	-0.00	$(-1.24, 1.03)$	0.99	$(0.02, 1.48)$	0.06	0.00
B	100000	17	-0.00	$(-0.47, 0.25)$	1.00	$(0.60, 1.29)$	0.02	0.00
B	1000000	16	-0.00	$(-0.17, 0.09)$	1.00	$(0.93, 1.06)$	0.01	-0.00
C	1000	18	-0.00	$(-1.58, 1.11)$	0.97	$(0.07, 1.74)$	0.16	—
C	10000	19	0.00	$(-1.19, 0.86)$	1.00	$(0.55, 1.40)$	0.06	—
C	100000	19	0.00	$(-0.19, 0.15)$	1.00	$(0.90, 1.24)$	0.02	—
C	1000000	20	0.00	$(-0.04, 0.09)$	1.00	$(0.88, 1.04)$	0.01	—
D	1000	14	-0.02	$(-2.41, 2.14)$	0.92	$(0.00, 1.63)$	0.19	-0.02
D	10000	13	0.00	$(-1.01, 1.53)$	0.98	$(0.00, 1.67)$	0.07	-0.00
D	100000	14	-0.00	$(-0.99, 0.59)$	1.00	$(0.33, 1.27)$	0.02	-0.00
D	1000000	13	-0.00	$(-0.22, 0.07)$	1.00	$(0.89, 1.08)$	0.01	0.00
E	1000	4	0.02	$(-4.04, 4.05)$	0.38	$(0.00, 1.66)$	0.60	-0.05
E	10000	4	-0.01	$(-3.63, 2.96)$	0.67	$(0.00, 1.97)$	0.41	-0.01
E	100000	4	0.01	$(-2.98, 2.74)$	0.84	$(0.00, 2.21)$	0.25	0.01
E	1000000	4	-0.02	$(-1.76, 2.10)$	0.95	$(0.00, 1.70)$	0.12	0.01

Table 1: The results of the simulation experiment. The first panel shows the SCM parameters: the number of observed variables $|\mathbf{V}|$, the number of conditions, the expected number of neighbors for the observed variables, and the expected number of global background variables per observed variable. The second panel reports the performance measures for cases A–E with different sample sizes n. The measures are the percentage of unique observations out of the total of n observations (column 'uniq. %', ideally 100%), the mean of sample means of standardized variables ($\overline{\overline{z}}$, ideally 0) and its minimum and maximum, the mean of sample standard deviations ($\overline{s}_z$, ideally 1) and its minimum and maximum, the average of Kolmogorov-Smirnov statistics (K-S, ideally 0), and the average difference between the true and the sample correlation coefficients (diff. cor., ideally 0).

depends on personal data and the amount of the loan. The credit decision is made by an automated prediction model that can be accessed via an application programming interface (API) but is otherwise unknown to the fairness evaluator. The outcome of the prediction model is the default risk. We consider three models:

A) The prediction model has no restrictions for the predictors.

B) The prediction model does not directly use sensitive variables.

C) The prediction model uses only non-sensitive variables that have a direct causal effect on the outcome.

It is expected that the evaluation should indicate that only prediction model C is fair according to Definition 5.

For the fairness evaluation, we constructed an SCM with the causal diagram shown in Figure 1. We consider variables that are similar to those typically present in credit-scoring datasets, such as Statlog (German Credit Data, Hofmann, 1994). Here, *default* is the outcome variable that indicates whether the customer will repay the loan or not. The annual income (in euros, continuous), the savings (in euros, continuous), the credit amount (in euros, continuous), type of housing (categorical), level of education (ordinal), the type of job (categorical), the length of employment (in months, discrete) and ethnicity (categorical) have a direct effect on the probability of default. The marital status (categorical), the number of children (discrete), age (in years, continuous), and gender (categorical) have only an indirect effect on the risk of default. Ethnicity and gender are the sensitive variables. The address does not have a causal effect on the risk of default but is included here as a potential proxy of ethnicity. The causal diagram also has five unobserved confounders $U_1, \ldots, U_5$. The dedicated error terms are not displayed. The detailed structure of the SCM is explained in Online Appendix 4 and the R code for the example is available in the repository `https://github.com/JuhaKarvanen/simulating_counterfactuals` in the file `fairness_example.R`.

To set up the example, we simulated training data from an SCM corresponding to the causal diagram of Figure 1 and fitted prediction models A, B, and C for the default risk using XGBoost (Chen & Guestrin, 2016; Chen, He, Benesty, Khotilovich, Tang, Cho, Chen, Mitchell, Cano, Zhou, Li, Xie, Lin, Geng, Li, & Yuan, 2023). These models are opaque AI models for the fairness evaluator who can only see the probability of default predicted by the models.

Algorithm 5 was applied to prediction models A, B, and C in 1000 cases that were again simulated from the same SCM. In the algorithm, $\widehat{Y}(\cdot)$ was one of the prediction models, $\mathcal{M}$ was the SCM whose causal diagram is depicted in Figure 1, sensitive variables $\mathbf{S}$ were gender and ethnicity, condition $\mathcal{C}$ contained all observed values of the case, and number of observations was $n = 1000$. For each case and each model, the counterfactual probability of default was estimated for all possible combinations of gender and ethnicity. For a fair prediction model, these probabilities should be the same regardless of gender and ethnicity as stated in Definition 4. Consequently, the difference between the largest and the smallest probability was chosen as a measure of fairness. For instance, if the estimated counterfactual probability of default was 0.09 for one combination of gender and ethnicity and 0.08 for all other combinations, the counterfactual difference would be $0.09 - 0.08 = 0.01$.

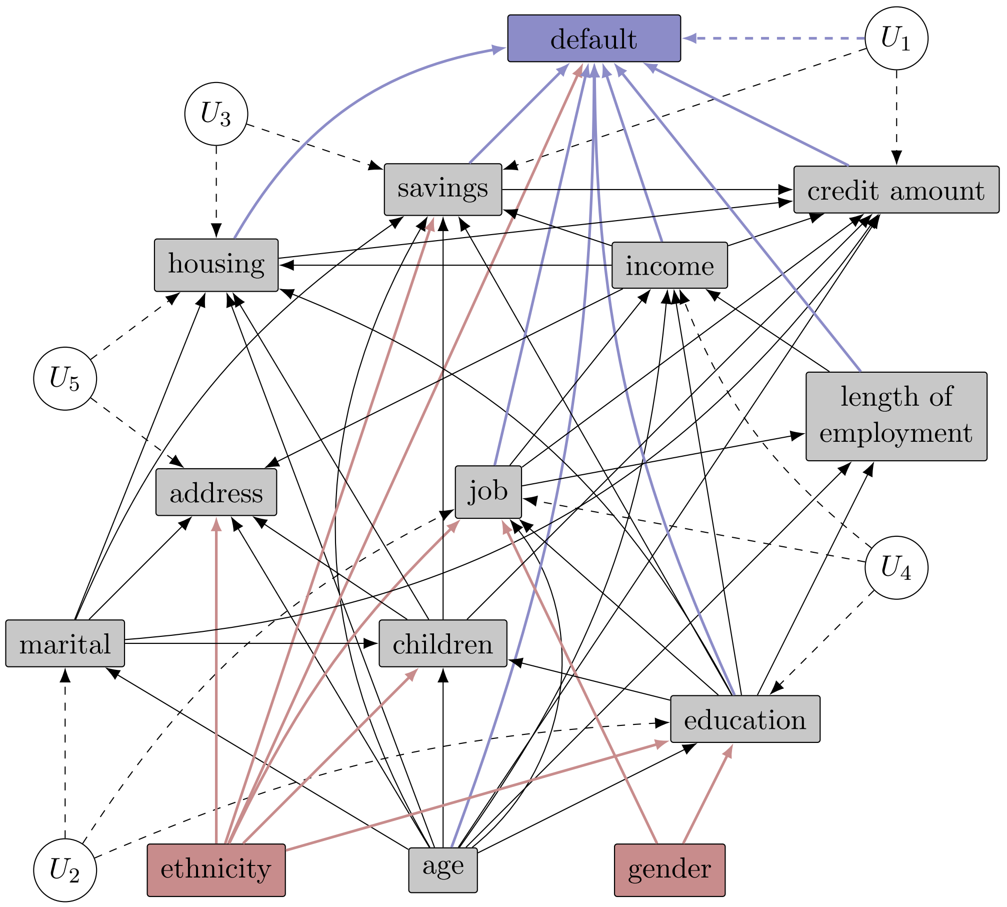

Figure 1: Causal model for the fairness of credit-scoring example. Ethnicity and gender (red nodes) are the sensitive variables, and the risk of default (blue node) is the outcome to be predicted. Gray nodes depict other observed variables and white nodes are unobserved confounders.

The fairness results based on the 1000 cases are presented in Table 2. Only prediction model C was evaluated to be fair. Even if the median counterfactual difference was close to zero for models A and B, there were cases where the difference was very large. Consequently, the use of models A and B in decision-making would potentially lead to illegal discrimination.

It would have been interesting to compare the results of Algorithm 5 with other methods of fairness evaluation but this was unattainable. Earlier works on counterfactual fairness (Kusner et al., 2017; Nabi & Shpitser, 2018; Chiappa, 2019; Wu et al., 2019; Richens et al., 2022) give some examples but do not provide a code that a user could directly apply to a new problem. On the other hand, `Fairlearn` (Weerts, Dudík, Edgar, Jalali, Lutz, & Madaio, 2023) is a general-purpose package for fairness evaluation but it does not currently include counterfactual definitions or metrics.

	Model		
Measure	A	B	C
Zero difference (%)	18	27	100
Difference < 0.01 (%)	90	94	100
Median difference	0.00031	0.00032	0.00000
Maximum difference	0.57	0.25	0.00

Table 2: The fairness results for the credit-scoring models. The first row indicates the percentage of cases where the counterfactual difference was exactly 0, i.e., the fairness criterion held exactly. The second row shows the percentage of cases where the difference was smaller than 0.01, i.e., the counterfactual probabilities were not necessarily the same but close to each other. The third and fourth rows report the median and maximum of the counterfactual difference, respectively.

6. Discussion

We presented an algorithm for simulating data from a specified counterfactual distribution (Algorithm 4). The algorithm receives a known SCM and a counterfactual of interest as inputs and simulates independent but non-unique observations from the counterfactual distribution. The algorithm is intended to be a tool for simulation studies on counterfactual inference and fairness evaluation. In Section 5, we demonstrate how the algorithm can be used in fairness analysis when the prediction models are unknown and real data are not available.

The proposed algorithm can be interpreted as a particle filter and it returns an approximate sample that yields asymptotically valid inference. The algorithm possesses multiple practical strengths. It is applicable to SCMs that may simultaneously have both continuous and discrete variables and may also have unobserved confounders. The algorithm solves the non-trivial problem of conditioning on continuous variables and operates rapidly with SCMs of sizes typically found in the literature. A documented open-source software implementation is available.

The counterfactual simulation algorithm has some requirements and limitations. The knowledge of the full causal model is a strong assumption in practical applications. However, counterfactual distributions cannot in general be simulated, for example, based on data alone, unless the counterfactual distribution in question is identifiable from the available data sources (Shpitser & Pearl, 2007; Tikka, 2023). Our approach could also be combined with a sensitivity analysis where the counterfactual simulation is repeated for different model parameters.

We presented the algorithm for SCMs which are u-monotonic with respect to continuous conditioning variables. We do not consider this as a serious restriction because the monotonicity requirement concerns only a subset of variables and offers flexibility for the functional form of the dedicated error terms' impact. Similar but more restrictive monotonicity assumptions have been also used for causal normalizing flows (Javaloy et al., 2023). In principle, the algorithms could be generalized to work with dedicated error terms that are *piecewise* monotonic. Then, the expression in Lemma 1 involves a sum of domain-restricted terms, and the dedicated error terms are no longer uniquely determined, and need to be simulated.

We note that our method can be generalized to multivariate observations (clustered variables, Tikka, Helske, & Karvanen, 2023) with multivariate dedicated error terms. Namely, if the multivariate transformation is a differentiable bijection, we may modify Lemma 1 using a multivariate transformation formula. However, in this case, the inverse transformation and the Jacobian must be available for point-wise evaluations.

Sample degeneracy is a well-known problem of particle filters that also affects the counterfactual simulation algorithm when the number of conditioning variables increases. The literature on particle filters proposes ways to address the degeneracy problem. One possibility is to run independent particle filters and weigh their outputs, which leads to consistent estimates (cf., Vihola, Helske, & Franks, 2018, Proposition 23). This is appealing because of its direct parallelizability but leads to a weighted sample. Another, perhaps even more promising approach, is to iterate the *conditional* particle filter (CPF, Andrieu, Doucet, & Holenstein, 2010), which is a relatively easy modification of the particle filter algorithm, and which defines a valid Markov chain targeting the desired conditional distribution.

When carefully configured, the CPF can work with other resampling algorithms (Chopin & Singh, 2015; Karppinen, Singh, & Vihola, 2022). CPFs with adaptive resampling have been suggested as well (Lee, 2011). We leave the practical efficiency of the CPF and its variants for future research.

The significance of counterfactual simulation emerges from the context of fairness evaluation. The fairness evaluation algorithm (Algorithm 5) uses simulated data, which extends the scope of evaluation to encompass data that could be potentially encountered (Cheng, Varshney, & Liu, 2021). We perceive the proposed fairness evaluation algorithm as complementary to methods that are based on real data. It is important to note that in addition to Definition 4, the simulation algorithm can also be applied to other definitions of counterfactual fairness. Other potential applications of the counterfactual simulation algorithm include counterfactual explanations and algorithmic recourse (Karimi et al., 2021). Furthermore, the conditional simulation (Algorithm 3) is applicable to Bayesian networks that lack a causal interpretation.

Acknowledgments

CSC – IT Center for Science, Finland, is acknowledged for computational resources. MV was supported by Research Council of Finland (Finnish Centre of Excellence in Randomness and Structures, grant 346311). ST was supported by Research Council of Finland (PREDLIFE: Towards well-informed decisions: Predicting long-term effects of policy reforms on life trajectories, grant 331817).

References

Andrieu, C., Doucet, A., & Holenstein, R. (2010). Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *72*(3), 269–342.

Bartlett, R., Morse, A., Stanton, R., & Wallace, N. (2022). Consumer-lending discrimination in the FinTech era. *Journal of Financial Economics*, *143*(1), 30–56.

Burkart, N., & Huber, M. F. (2021). A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, *70*, 245–317.

Cappé, O., Moulines, E., & Rydén, T. (2005). *Inference in Hidden Markov Models.* Springer Series in Statistics. Springer.

Carey, A. N., & Wu, X. (2022). The causal fairness field guide: Perspectives from social and formal sciences. *Frontiers in Big Data*, *5*, 892837.

Caton, S., & Haas, C. (2023). Fairness in machine learning: A survey. *ACM Computing Surveys*, *56*(7), 1–38.

Chang, W. (2021). *R6: Encapsulated Classes with Reference Semantics.* R package version 2.5.1.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794. Association for Computing Machinery.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., Li, M., Xie, J., Lin, M., Geng, Y., Li, Y., & Yuan, J. (2023). *xgboost: Extreme Gradient Boosting.* R package version 1.7.5.1.

Cheng, L., Varshney, K. R., & Liu, H. (2021). Socially responsible AI algorithms: Issues, purposes, and challenges. *Journal of Artificial Intelligence Research*, *71*, 1137–1181.

Chiappa, S. (2019). Path-specific counterfactual fairness. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pp. 7801–7808.

Chopin, N., & Papaspiliopoulos, O. (2020). *An Introduction to Sequential Monte Carlo.* Springer.

Chopin, N., & Singh, S. S. (2015). On particle Gibbs sampling. *Bernoulli*, *21*(3), 1855–1883.

Chopin, N., Singh, S. S., Soto, T., & Vihola, M. (2022). On resampling schemes for particle filters with weakly informative observations. *Annals of Statistics*, *50*(6), 3197–3222.

Correa, J., Lee, S., & Bareinboim, E. (2021). Nested counterfactual identification from arbitrary surrogate experiments. In *Advances in Neural Information Processing Systems*, Vol. 34, pp. 6856–6867.

Del Moral, P. (2004). *Feynman-Kac Formulae.* Springer.

Douc, R., & Cappé, O. (2005). Comparison of resampling schemes for particle filtering. In *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, pp. 64–69.

Doucet, A., Godsill, S., & Andrieu, C. (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, *10*, 197–208.

Gordon, N. J., Salmond, D. J., & Smith, A. F. M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings-F*, *140*(2), 107–113.

Guidotti, R. (2022). Counterfactual explanations and how to find them: Literature review and benchmarking. *Data Mining and Knowledge Discovery*, *http://dx.doi.org/10.1007/s10618-022-00831-6*.

Hofmann, H. (1994). Statlog (German Credit Data). UCI Machine Learning Repository. DOI: 10.24432/C5NC77.

Javaloy, A., Sánchez-Martín, P., & Valera, I. (2023). Causal normalizing flows: From theory to practice. arXiv preprint arXiv:2306.05415.

Karimi, A.-H., Schölkopf, B., & Valera, I. (2021). Algorithmic recourse: From counterfactual explanations to interventions. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pp. 353–362.

Karppinen, S., Singh, S. S., & Vihola, M. (2022). Conditional particle filters with bridge backward sampling. arXiv preprint arXiv:2205.13898.

Karvanen, J. (2024). *R6causal: R6 Class for Structural Causal Models.* R package version 0.8.3.

Kusner, M., Loftus, J., Russell, C., & Silva, R. (2017). Counterfactual fairness. In *Advances in Neural Information Processing Systems*, Vol. 30, pp. 4069–4079.

Lee, A. (2011). *On auxiliary variables and many-core architectures in computational statistics.* PhD thesis, University of Oxford.

Li, T., Sun, S., Sattar, T. P., & Corchado, J. M. (2014). Fight sample degeneracy and impoverishment in particle filters: A review of intelligent approaches. *Expert Systems with Applications*, *41*(8), 3944–3954.

Liu, J. S., & Chen, R. (1995). Blind deconvolution via sequential imputations. *Journal of American Statistical Association*, *90*(430), 567–576.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, *54*(6), 1–35.

Nabi, R., & Shpitser, I. (2018). Fair inference on outcomes. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pp. 1931–1940. AAAI Press.

Pearl, J. (2009). *Causality: Models, Reasoning, and Inference* (2nd edition). Cambridge University Press.

Pessach, D., & Shmueli, E. (2022). A review on fairness in machine learning. *ACM Computing Surveys (CSUR)*, *55*(3), 1–44.

Richens, J., Beard, R., & Thompson, D. H. (2022). Counterfactual harm. In *Advances in Neural Information Processing Systems*, Vol. 35, pp. 36350–36365.

Rohrbach, P. B., & Jack, R. L. (2022). Convergence of random-weight sequential Monte Carlo methods. arXiv preprint arXiv:2208.12108.

Shpitser, I., & Pearl, J. (2007). What counterfactuals can be tested. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pp. 352–359. AUAI Press.

Shpitser, I., & Pearl, J. (2008). Complete identification methods for the causal hierarchy. *Journal of Machine Learning Research*, *9*, 1941–1979.

Shpitser, I., & Sherman, E. (2018). Identification of personalized effects associated with causal pathways. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence*, pp. 530–539.

Tikka, S. (2023). Identifying counterfactual queries with the R package cfid. *The R Journal*, *15*(2), 330–343.

Tikka, S., Helske, J., & Karvanen, J. (2023). Clustering and structural robustness in causal diagrams. *Journal of Machine Learning Research*, *24*(195), 1–32.

Tikka, S., & Karvanen, J. (2018). Enhancing identification of causal effects by pruning. *Journal of Machine Learning Research*, *18*(194), 1–23.

Vihola, M., Helske, J., & Franks, J. (2018). Importance sampling type estimators based on approximate marginal MCMC. arXiv preprint arXiv:1609.02541v5. (version 5).

Wachter, S., Mittelstadt, B., & Russell, C. (2018). Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology*, *31*(2), 841–861.

Weerts, H., Dudík, M., Edgar, R., Jalali, A., Lutz, R., & Madaio, M. (2023). Fairlearn: Assessing and improving fairness of AI systems. *Journal of Machine Learning Research*, *24*(257), 1–8.

Wu, Y., Zhang, L., Wu, X., & Tong, H. (2019). PC-fairness: A unified framework for measuring causality-based fairness. In *Advances in Neural Information Processing Systems*, Vol. 32.