Identifying Causal Effects with the R Package causaleffect

Santtu Tikka
University of Jyvaskyla

Juha Karvanen
University of Jyvaskyla

Abstract

Do-calculus is concerned with estimating the interventional distribution of an action from the observed joint probability distribution of the variables in a given causal structure. All identifiable causal effects can be derived using the rules of do-calculus, but the rules themselves do not give any direct indication whether the effect in question is identifiable or not. Shpitser and Pearl (2006b) constructed an algorithm for identifying joint interventional distributions in causal models, which contain unobserved variables and induce directed acyclic graphs. This algorithm can be seen as a repeated application of the rules of do-calculus and known properties of probabilities, and it ultimately either derives an expression for the causal distribution, or fails to identify the effect, in which case the effect is non-identifiable. In this paper, the R package **causaleffect** is presented, which provides an implementation of this algorithm. Functionality of **causaleffect** is also demonstrated through examples.

Keywords: DAG, do-calculus, causality, causal model, identifiability, graph, C-component, hedge, d-separation.

A modification of (Tikka and Karvanen 2017) published in the Journal of Statistical Software.

1. Introduction

When discussing causality, one often means the relationships between events, where a set of events directly or indirectly causes another set of events. The aim of causal inference is to draw conclusions from these relationships by using available data and prior knowledge. Causal inference can also be applied when determining the effects of actions on some variables of interest. These types of actions are often called interventions and the results of the interventions are referred to as causal effects.

The causal inference can be divided into three sub-areas: discovering the causal model from the data, identifying the causal effect when the causal structure is known and estimating an identifiable causal effect from the data. Our contribution belongs to the second category, identification of causal effects. As a starting point, we assume that the causal relationships between the variables are known in a non-parametric form and formally presented as a probabilistic causal model (Pearl 1995). Part of the variables may be latent. The causal structure, i.e., the non-parametric causal relationships, can be described using a directed acyclic graph (DAG). A causal effect is called identifiable if it can be uniquely determined from the causal structure on basis of the observations only.

Do-calculus (Pearl 1995) consist of a set of inference rules, which can be used to express the interventional probability distribution using only observational distributions. The rules of do-calculus do not themselves indicate the order in which they should be applied. This problem is solved in the algorithm developed by Tian and Pearl (2003) and Shpitser and Pearl (2006b). The algorithm is proved to determine the interventional distribution of an identifiable causal effect. When faced with an unidentifiable effect, the algorithm provides a problematic graph structure called a hedge, which can be thought of as the cause of unidentifiability.

Other R packages for causal inference are summarized in Table 1. It can be seen that in addition to **causaleffect**, only **pcalg** (Kalisch *et al.* 2012) supports the identification of causal effects. **pcalg** supports the generalized back-door criterion but does not support the front-door criterion. Thus, according to our knowledge, **causaleffect** is the only R package that implements a complete algorithm for the identification of causal effects.

An algorithm equivalent to the one developed by (Shpitser and Pearl 2006b) has been implemented earlier by Lexin Liu in the **CIBN** software using **JavaBayes**, which is a graphical software interface written in Java by Fabio Gagliardi Cozman. In addition to causal effect identification **CIBN** also provides tools for creating and editing graphical models. **CIBN** is freely available from `http://web.cs.iastate.edu/~jtian/Software/CIBN.htm`. **DAGitty** (Textor *et al.* 2011) provides another free interface for causal inference and causal modeling. One of the main features of **DAGitty** is finding sufficient adjustment sets for the minimization of bias in causal effect estimation. **DAGitty** can also be used to determine instrumental variables, which is a feature currently not provided by **causaleffect**. However, **DAGitty** does not provide a complete criterion for identifiability.

Familiarity of Pearl's causal model, do-calculus and basic graph theory is assumed throughout the paper. These concepts are briefly reviewed in Appendix A. A more detailed description can be found in (Pearl 2009) and (Koller and Friedman 2009). Notation similar to that of (Shpitser and Pearl 2006b) is also utilized repeatedly in this paper. Capital letters denote variables and small letters denote their values. Bold letters denote sets which are formed of the previous two. The abbreviations $Pa(\mathbf{Y})_G, An(\mathbf{Y})_G$, and $De(\mathbf{Y})_G$ denote the set of observable parents, ancestors and descendants of the node set $\mathbf{Y}$ while also containing $\mathbf{Y}$ itself. It should also be noted that the shorthand notation of bidirected edges is used to represent the direct effects of an unobserved confounding variable on the two variables at the endpoints of the bidirected edge.

A motivating example is presented in Section 2. The identification algorithm is presented in Section 3 and the details of its R implementation are described in Section 4. Section 5 showcases the usage of **causaleffect** in R with some simple examples, and describes some curious special cases arising from the nature of the algorithm itself. Section 6 concludes this paper by providing some examples of similar algorithms, where the work of this paper could be applicable.

2. Example on do-calculus

Consider identification of causal effect $P_x(y)$ in the graph G of Figure 1. We show how this causal effect can be identified by applying do-calculus (Pearl 2009) manually. Later the same example is reconsidered using the identification algorithm.

First, the rules of do-calculus are shortly reviewed. The purpose of do-calculus is to represent

Packages for specific applications	
ASPBay	Bayesian inference on causal genetic variants using affected sib-pairs data (Dandine-Roulland 2015)
cin	Causal inference for neuroscience (Luo *et al.* 2011)
mwa	Causal inference in spatiotemporal event data (Schutte and Donnay 2015)
qtlnet	Causal inference of QTL networks (Neto and Yandell 2014)
Packages for estimation of causal effects from data	
CausalGAM	Estimation of causal effects with generalized additive models (Glynn and Quinn 2010)
InvariantCausalPrediction	Invariant causal prediction (Meinshausen 2015)
iWeigReg	Improved methods for causal inference and missing data problems (Tan and Shu 2013)
pcalg	Methods for graphical models and causal inference
SVMMatch	Causal effect estimation and diagnostics with support vector machines (Ratkovic 2015)
wfe	Weighted linear fixed effects regression models for causal inference (Kim and Imai 2014)
Packages for sensitivity analysis and other specific problems in causal inference	
causalsens	Selection bias approach to sensitivity analysis for causal effects (Blackwell 2013)
cit	Causal inference test (Millstein 2015)
ImpactIV	Identifying causal effect for multi-component intervention using instrumental variable method (Ding 2012)
inferference	Methods for causal inference with interference (Saul 2015)
MatchingFrontier	Computation of the balance – sample size frontier in matching methods for causal inference (King *et al.* 2015)
mediation	Causal mediation analysis (Tingley *et al.* 2014)
qualCI	Causal inference with qualitative and ordinal information on outcomes(Kashin *et al.* 2014)
SimpleTable	Bayesian inference and sensitivity analysis for causal effects from 2×2 and $2 \times 2 \times K$ tables in the presence of unmeasured confounding (Quinn 2012)
treatSens	Sensitivity analysis for causal inference (Carnegie *et al.* 2015)
Packages for causal discovery	
CAM	Causal additive model (CAM) (Peters and Ernest 2015)
D2C	Predicting causal direction from dependency features (Bontempi *et al.* 2015)
pcalg	Methods for graphical models and causal inference
Packages for identification of causal effects	
causaleffect	Deriving expressions of joint interventional distributions in causal models
pcalg	Methods for graphical models and causal inference

Table 1: R packages for causal inference.

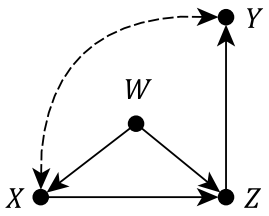


Figure 1: Graph G for the illustrative example.

the interventional distribution $P_{\mathbf{x}}(\mathbf{y})$ by using only observational probabilities. A causal effect is identifiable, if such an expression can be found by applying the rules of do-calculus repeatedly. This result follow directly from the definition of identifiability due to the fact that all observational distributions are assumed identical for the causal models that induce G.

Let $\mathbf{X}, \mathbf{Y}$ and $\mathbf{Z}$ be pairwise disjoint sets of nodes in the graph G induced by a causal model M. Here $G_{\overline{\mathbf{X}},\underline{\mathbf{Z}}}$ means the graph that is obtained from G by removing all incoming edges of $\mathbf{X}$ and all outgoing edges of $\mathbf{Z}$. Let P be the joint distribution of all observed and unobserved variables of M. Now, the following three rules hold (Pearl 1995):

1. Insertion and deletion of observations:

$$P_{\mathbf{x}}(\mathbf{y}|\mathbf{z},\mathbf{w}) = P_{\mathbf{x}}(\mathbf{y}|\mathbf{w}), \text{ if } (\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X},\mathbf{W})_{G_{\overline{\mathbf{X}}}}.$$

2. Exchanging actions and observations:

$$P_{\mathbf{x},\mathbf{z}}(\mathbf{y}|\mathbf{w}) = P_{\mathbf{x}}(\mathbf{y}|\mathbf{z},\mathbf{w}), \text{ if } (\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X},\mathbf{W})_{G_{\overline{\mathbf{X}},\underline{\mathbf{Z}}}}.$$

3. Insertion and deletion of actions:

$$P_{\mathbf{x},\mathbf{z}}(\mathbf{y}|\mathbf{w}) = P_{\mathbf{x}}(\mathbf{y}|\mathbf{w}), \text{ if } (\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X},\mathbf{W})_{G_{\overline{\mathbf{X}},\overline{Z(\mathbf{W})}}},$$

   where $Z(\mathbf{W}) = \mathbf{Z} \setminus An(\mathbf{W})_{G_{\overline{\mathbf{X}}}}$.

The rules of do-calculus can be shown to be true by using d-separation and the definition of the $do(\cdot)$-operator. Pearl presented proofs for these three rules (Pearl 1995). Do-calculus has also been shown to be complete, meaning that the expressions of all identifiable causal effects can be derived by using the three rules (Shpitser and Pearl 2006b; Huang and Valtorta 2006).

To identify $P_x(y)$ in the causal model of Figure 1, we begin with the factorization

$$P_x(y) = \sum_{w,z} P_x(y|w,z)P_x(z|w)P_x(w). \tag{1}$$

Let us start by focusing on the first term in the sum. Because $(Y \perp\!\!\!\perp Z|X,W)_{G_{\overline{X},\underline{Z}}}$ rule 2 implies that

$$P_x(y|w,z) = P_{x,z}(y|w)$$

and by noting that $(Y \perp\!\!\!\perp X|Z,W)_{G_{\overline{Z},\overline{X}}}$ rule 3 allows us to write

$$P_{x,z}(y|w) = P_z(y|w).$$

By expanding the previous expression we get

$$P_z(y|w) = \sum_x P_z(y|w,x)P_z(x|w). \tag{2}$$

Rule 2 and the fact that $(Y \perp\!\!\!\perp Z|X,W)_{G_{\underline{Z}}}$ together imply

$$P_z(y|w,x) = P(y|w,x,z). \tag{3}$$

The condition $(X \perp\!\!\!\perp Z|W)_{G_{\overline{Z}}}$ and rule 3 allow us to write

$$P_z(x|w) = P(x|w). \tag{4}$$

Inserting (3) and (4) into (2) yields

$$P_z(y|w) = \sum_x P(y|w,x,z)P(x|w). \tag{5}$$

Focusing now on the second term of (1) we see that because $(Z \perp\!\!\!\perp X|W)_{G_{\underline{X}}}$ rule 2 implies that

$$P_x(z|w) = P(z|x,w). \tag{6}$$

Similarly, the third term simplifies by using rule 3 and the condition $(W \perp\!\!\!\perp X)_{G_{\overline{X}}}$ rule 3.

$$P_x(w) = P(w). \tag{7}$$

Finally, we combine the results above by inserting (5), (6) and (7) into (1) which yields the expression for the causal effect.

$$P_x(y) = \sum_{w,z} \left(\sum_x P(y|w,x,z)P(x|w) \right) P(z|x,w)P(w)$$

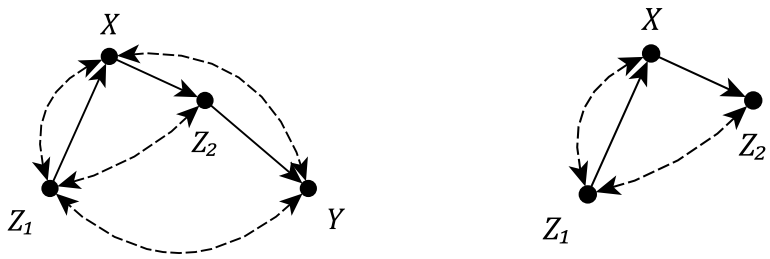
In Section 3.3 we will see how the causal effect can be identified by applying the algorithm of (Shpitser and Pearl 2006b). The previous result highly resembles the front-door criterion, which states that

$$P_{\mathbf{x}}(\mathbf{y}) = \sum_{\mathbf{s}} \left(\sum_{\mathbf{x}} P(\mathbf{y}|\mathbf{x},\mathbf{s})P(\mathbf{x}) \right) P(\mathbf{s}|\mathbf{x}),$$

whenever the set $\mathbf{S}$ blocks all directed paths from $\mathbf{X}$ to $\mathbf{Y}$, there are no unblocked back-door paths from $\mathbf{X}$ to $\mathbf{S}$ and $\mathbf{X}$ blocks all back-door paths from $\mathbf{S}$ to $\mathbf{Y}$. However, neither W, Z, or $\{W,Z\}$ satisfy the role of the set $\mathbf{S}$. The criterion would certainly hold if we removed W from the graph.

3. Identifiability algorithm

Even if a causal effect is identifiable, the rules of do-calculus themselves do not guarantee that they could be used to form an expression for the interventional distribution, and that it would contain only observed quantities. It is also not self-evident in which order the rules of do-calculus should be applied to reach the desired expression from the joint distribution of the observed variables $P(\mathbf{V})$.



(a) Graph G. (b) A subgraph of G induced by the set $\{X, Z_1, Z_2\}$.

Figure 2: An example illustrating the definition of an induced subgraph.

To overcome these limitations an identifiability algorithm has been developed by Shpitser and Pearl (2006b). This algorithm can be used to determine the identifiability of any causal effect, in addition of generating the expression for the interventional distribution in the case of an identifiable effect.

3.1. Definitions

Some graph theoretic definitions are necessary in order to present the algorithm. The notation mostly follows that of (Shpitser and Pearl 2006b) with some slight alterations for the benefit of the reader.

Definition 1 (Induced Subgraph)**.** Let $H = \langle \mathbf{W}, \mathbf{F} \rangle$ and $G = \langle \mathbf{V}, \mathbf{E} \rangle$ be graphs such that $\mathbf{W} \subset \mathbf{V}$. If every pair of nodes $X, Y \in \mathbf{W}$ is connected by an edge in graph H precisely when they are connected by an edge of the same direction in graph G, then H is an *induced subgraph* induced by the set $\mathbf{W}$ and $H = G[\mathbf{W}]$.

Defining new graphs using only a set of nodes can easily be achieved using induced subgraphs. For example, the graph in Figure 2(b) is an induced subgraph induced by the nodes X, Z_1 and Z_2 from G in 2(a).

Perhaps the most important definition is C-component (confounded component).

Definition 2 (C-component, (Shpitser and Pearl 2006b) 3)**.** Let $G = \langle \mathbf{V}, \mathbf{E} \rangle$ be a graph. If there exists a set $\mathbf{B}$ such that $\mathbf{B} \subset \mathbf{E}$ and $\mathbf{B}$ contains only bidirected edges, and the graph $\langle \mathbf{V}, \mathbf{B} \rangle$ is connected, then G is a *C-component*.

Both graphs in Figure 2 are examples of C-components. Even if a graph is not a C-component, at least one of its subgraphs is guaranteed to be a C-component because every subgraph induced by a single node is always a C-component. It is often of greater interest to determine how a given graph can be partitioned in C-components that contain as many nodes as possible.

Definition 3 (Maximal C-component)**.** Let G be a graph and $C = \langle \mathbf{V}, \mathbf{E} \rangle$ a C-component such that $C \subset G$. C-component C is *maximal* (with respect to graph G) if $H \subset C$ for every bidirected path H of graph G which contains at least one node of the set $\mathbf{V}$.

Tian (2002) proved, that the joint probability distribution $P(\mathbf{V})$ of the observed variables of graph G can always be factorized in such a way, that each term of the resulting product corresponds to a maximal C-component. This property is in a fundamental role in the algorithm,

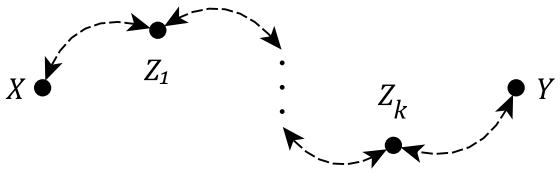


Figure 3: Path H.

since it can be used to recursively divide the expression of the interventional distribution into simpler expressions.

If a given graph G is not a C-component, it can still be divided into a unique set $C(G)$ of subgraphs, each a maximal C-component of G. This follows from the fact, that there exists a bidirected path between two nodes in G if and only if they belong in the same maximal C-component, which in turn follows from the definition of a maximal C-component. This means, that the bidirected paths of graph G completely define its maximal C-components.

C-trees are a special case of C-components. They are closely related to direct effects, which are causal effects of the form $P_{Pa(Y)}(Y)$.

Definition 4 (C-tree, (Shpitser and Pearl 2006b) 4)**.** Let G be a C-component such that every observed node has at most one child. If there is a node Y such that $G[An(Y)_G] = G$, then G is a *Y-rooted C-tree*.

Using only C-trees and C-components it is already possible to characterize identifiability of effects on a single variable. C-forest is the multivariate generalization of a C-tree in such a way that the *root set*, which is the set of nodes $\{X \in G \mid De(X)_G \setminus \{X\} = \emptyset\}$, contains one or more nodes.

Definition 5 (C-forest, (Shpitser and Pearl 2006b) 5)**.** Let G be a graph and $\mathbf{Y}$ its root set. If G is a C-component, and every observed node has at most one child, then G is *$\mathbf{Y}$-rooted C-forest*.

Both C-components in Figure 2 are also C-forests, because every observed node has at most one child in both graphs. In addition, their root sets consist only of a single node. There exists a connection between C-forests and general causal effects of the form $P_{\mathbf{x}}(\mathbf{Y})$. A graph structure formed by a pair of C-trees is used to determine such effects.

Shpitser and Pearl (2006b) proved, that if a graph G contains a hedge for $P_{\mathbf{x}}(\mathbf{y})$, then the effect is not identifiable.

Definition 6 (Hedge, (Shpitser and Pearl 2006b) 6)**.** Let $G = \langle \mathbf{V}, \mathbf{E} \rangle$ be a graph, and $\mathbf{X}, \mathbf{Y} \subset \mathbf{V}$ disjoint subsets. If there are two $\mathbf{R}$-rooted C-forests $F = \langle \mathbf{V}_F, \mathbf{E}_F \rangle$ and $F' = \langle \mathbf{V}_{F'}, \mathbf{E}_{F'} \rangle$ such that $\mathbf{V}_F \cap \mathbf{X} \neq \emptyset$, $\mathbf{V}_{F'} \cap \mathbf{X} = \emptyset, F' \subset F$, and $\mathbf{R} \subset An(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$, then F and F' form *hedge* for $P_{\mathbf{x}}(\mathbf{y})$ in G.

Hedges are a remarkable structure, since they generalize certain results regarding identifiability. One example of such a result is the condition for identification of a causal effect of the form $P_x(\mathbf{y})$ in (Tian and Pearl 2002). The result states that $P_x(\mathbf{y})$ is identifiable if and only if there are no bidirected paths between X and any of its children in $G[An(\mathbf{Y})_G]$. Consider the graph $H = \langle \mathbf{V}, \mathbf{E} \rangle$ in Figure 3 containing the nodes X and Y and a bidirected path

connecting them formed by the intermediary nodes $\{Z_1, \ldots, Z_k\}$. One can observe, that the C-forests H and $H[\mathbf{V} \setminus \{X\}]$ form a hedge for $P_x(Y, Z_1, \ldots, Z_k)$.

3.2. Algorithm

Using the previously presented definitions it is now possible to define Algorithm 1, which completely characterizes the identifiability problem of general causal effects. Shpitser and Pearl (2006b) showed, that the expression returned by Algorithm 1 for $P_{\mathbf{x}}(\mathbf{y})$ is always correct if the effect in question is identifiable. They also showed, that if the algorithm is interrupted on line five, then the original graph G contains a hedge, preventing the identifiability of the effect. The existence of a hedge is therefore equivalent with unidentifiability. This result also shows the completeness of do-calculus, because the algorithm only applies standard rules of probability manipulations and the three rules of do-calculus. All variables are assumed to be discrete, but the algorithm can also be applied in a continuous case, when the respective sums are replaced with integrals.

The algorithm is required to be able to iteratively process the nodes of the graph, which means that the nodes have to be ordered in some meaningful fashion. This ordering must be able to take the directions of the edges into account, and at least one such ordering must always exist for any given graph. Topological ordering has all of these prerequisite properties.

Definition 7 (Topological Ordering)**.** *Topological ordering* π of a DAG $G = \langle \mathbf{V}, \mathbf{E} \rangle$ is an ordering of its nodes, where either $X > Y$ or $Y > X$ for all pairs of nodes $X, Y \in \mathbf{V}$, $X \neq Y$ in G. In addition, no node can be greater than its descendants in π. In other words, if X is an ancestor of Y in G, then $X < Y$.

There exists at least one topological ordering for any DAG, but in some cases there can be multiple orderings. One way to always construct an ordering for a given graph is to begin by determining all nodes without parents, and ordering them arbitrarily. Next, all nodes without parents excluding the nodes found in previous step are determined and again ordered arbitrarily. It is also assigned, that the largest node in the previous step is smaller than the smallest node in the current step. This process is iterated, until all nodes have been ordered.

Algorithm 1 is simple in a sense that at each recursion stage the computation proceeds to exactly one line only. This is easy to see from the fact that after a condition regarding any of the line has been checked, either a **return** or a **FAIL** command will be executed. If $\mathbf{x} = \emptyset$ on line one, then the marginal distribution $P(\mathbf{y})$ is computed instead of a causal effect. This can be achieved by marginalizing over the joint distribution $P(\mathbf{V})$. On line two, all non-ancestors of $\mathbf{Y}$ in G are eliminated. This is possible due to the fact that the input of the algorithm assumes that G is an I-map of G and thus all necessary conditional independences hold. On line three, interventions are added to the original causal effect, which is feasible due to the third rule of do-calculus, because $(\mathbf{Y} \perp\!\!\!\perp \mathbf{W} | \mathbf{X})_{G_{\overline{\mathbf{X}},\overline{\mathbf{W}}}}$.

It is possible to index the nodes of G and the nodes of any subgraph of G using the topological ordering. This property is utilized on lines four, six and seven. The notation $V_\pi^{(i-1)}$ refers to all nodes in G that are smaller than V_i in π. Any topological ordering of G is also a topological ordering for any subgraph of G. This means, that it is unnecessary to determine a new ordering for each subgraph of G. Instead, one can fix the ordering before applying the algorithm.

INPUT: Value assignments $\mathbf{x}$ and $\mathbf{y}$, joint distribution $P(\mathbf{v})$ and a DAG $G = \langle \mathbf{V}, \mathbf{E} \rangle$. G is an I-map of P.
OUTPUT: Expression for $P_{\mathbf{x}}(\mathbf{y})$ in terms of $P(\mathbf{v})$ or **FAIL**(F, F').

function ID$(\mathbf{y}, \mathbf{x}, P, G)$
1: **if** $\mathbf{x} = \emptyset$, **then**
  return $\sum_{v \in \mathbf{v} \setminus \mathbf{y}} P(\mathbf{v})$.
2: **if** $\mathbf{V} \neq An(\mathbf{Y})_G$, **then**
  return ID$(\mathbf{y}, \mathbf{x} \cap An(\mathbf{Y})_G, P(An(\mathbf{Y})_G), G[An(\mathbf{Y})_G])$.
3: **Let** $\mathbf{W} = (\mathbf{V} \setminus \mathbf{X}) \setminus An(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$.
  if $\mathbf{W} \neq \emptyset$, **then**
    return ID$(\mathbf{y}, \mathbf{x} \cup \mathbf{w}, P, G)$.
4: **if** $C(G[\mathbf{V} \setminus \mathbf{X}]) = \{G[\mathbf{S}_1], \ldots, G[\mathbf{S}_k]\}$, **then**
  return $\sum_{v \in \mathbf{v} \setminus (\mathbf{y} \cup \mathbf{x})} \prod_{i=1}^{k}$ **ID**$(\mathbf{s}_i, \mathbf{v} \setminus \mathbf{s}_i, P, G)$.
  if $C(G[\mathbf{V} \setminus \mathbf{X}]) = \{G[\mathbf{S}]\}$, **then**
5: **if** $C(G) = \{G\}$, **then**
    throw FAIL$(G, G[\mathbf{S}])$.
6: **if** $G[\mathbf{S}] \in C(G)$, **then**
    return $\sum_{v \in \mathbf{s} \setminus \mathbf{y}} \prod_{V_i \in \mathbf{S}} P(v_i | v_\pi^{(i-1)})$.
7: **if** $(\exists \mathbf{S}')\mathbf{S} \subset \mathbf{S}'$ such that $G[\mathbf{S}'] \in C(G)$, **then**
    return ID$(\mathbf{y}, \mathbf{x} \cap \mathbf{s}', \prod_{V_i \in \mathbf{S}'} P(V_i | V_\pi^{(i-1)} \cap \mathbf{S}', v_\pi^{(i-1)} \setminus \mathbf{s}'), G[\mathbf{S}'])$.

Algorithm 1: The causal effect of intervention $do(\mathbf{X} = \mathbf{x})$ on $\mathbf{Y}$.

The maximal C-components of $G[\mathbf{V} \setminus \mathbf{X}]$ are determined on line four and their factorization property is utilized. If more than one C-components were found, it is now necessary to calculate a new causal effect for every C-component. The algorithm proceeds to either line five, six or seven in the case if only one C-component was found.

If Algorithm 1 throws **FAIL**, then the original graph G contains a hedge formed by graph G and $G[\mathbf{S}]$ of the current recursion stage, due to which the original effect is not identifiable and computation terminates. If the algorithm continues, then it is necessary to determine whether $G[\mathbf{S}]$ is a maximal C-component of G. If this is the case, then the condition of line six has been satisfied. In the other case, the computation of the intervention can be limited to the intersection of sets $\mathbf{X}$ and $\mathbf{S}'$ on line seven.

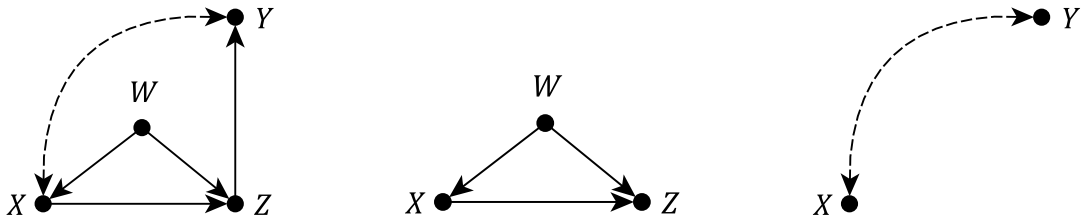
Identifiability of conditional interventional distributions is characterized by Algorithm 2. This algorithm is a generalization of Algorithm 1 and in fact it utilizes the function **ID** in the computation. It was constructed by Shpitser and Pearl (2006a) for identifying conditional causal effects i.e., causal effects of the form $P_{\mathbf{x}}(\mathbf{y}|\mathbf{z})$. They showed, that this algorithm is also sound and complete for identifying all such effects.

The primary focus of this paper however, is the implementation of Algorithm 1. The implementation of Algorithm 2 follows seamlessly from this implementation, because at the bottom of any recursive stack of **IDC** the function **ID** is ultimately called, which determines if the original conditional effect is identifiable. The only additional task is to determine whether a suitable node for the d-separation condition exists on line 1.

INPUT: Value assignments $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$, joint distribution $P(\mathbf{v})$ and a DAG $G = \langle \mathbf{V}, \mathbf{E} \rangle$. G is an I-map of P.
OUTPUT: Expression for $P_{\mathbf{x}}(\mathbf{y}|\mathbf{z})$ in terms of $P(\mathbf{v})$ or **FAIL**(F, F').

function IDC$(\mathbf{y}, \mathbf{x}, \mathbf{z}, P, G)$
1: **if** $\exists Z \in \mathbf{Z}$ such that $(\mathbf{Y} \perp\!\!\!\perp Z | \mathbf{X}, \mathbf{Z} \setminus \{Z\})_{G_{\overline{\mathbf{X}},\underline{\mathbf{Z}}}}$ **then**
return IDC$(\mathbf{y}, \mathbf{x} \cup \{z\}, \mathbf{z} \setminus \{z\}, P, G)$.
2: **else let** $P' =$ **ID**$(\mathbf{y} \cup \mathbf{z}, \mathbf{x}, P, G)$.
return $P' / \sum_{y \in \mathbf{y}} P'$

Algorithm 2: The causal effect of intervention $do(\mathbf{X} = \mathbf{x})$ on $\mathbf{Y}$ given $\mathbf{Z}$.



(a) Graph G (b) Subgraph $G[An(Z)_G]$. (c) Subgraph $G[\mathbf{S}']$.

Figure 4: Graph G and its subgraphs.

3.3. Application in practice

We return to the example presented in Section 2. The graph of the example along with some subgraphs are shown here in Figure 4. Let $G = \langle \mathbf{V}, \mathbf{E} \rangle$ be a graph such as in Figure 4(a) and a causal effect of interest $P_x(y)$, which is to be identified from the joint distribution $P(X, Y, Z, W)$. Only a single topological ordering exists for the nodes of G, and it is $W < X < Z < Y$. Clearly $\mathbf{x} \neq \emptyset, \mathbf{V} = An(Y)_G$ and $\mathbf{W} = \emptyset$, so the first three lines are ignored and line four is triggered, since

$$C(G[\mathbf{V} \setminus \{X\}]) = \{G[W], G[Z], G[Y]\}.$$

Because $\mathbf{v} \setminus (\{y\} \cup \{x\}) = \{w, z\}$, it is now necessary to identify three new causal effects in the following expression:

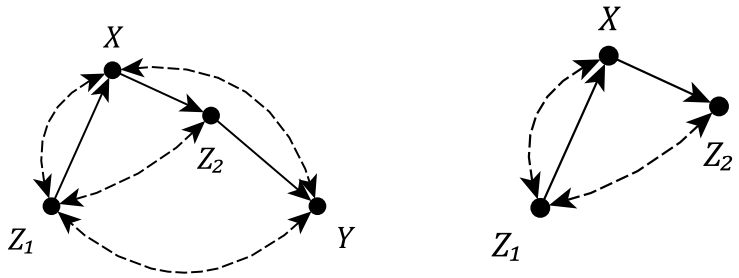
$$\sum_{w,z} P_{x,z,y}(w) P_{w,x,y}(z) P_{w,x,z}(y).$$

Consider the first term of the product. Because $\mathbf{V} \neq An(W)_G$, line two is triggered, and non-ancestors of W are ignored. This results in the first term simplifying to $P(w)$ because $An(W)_G = \{W\}$. Line two is also triggered when computing the second term, and

$$P_{w,x,y}(z) = P_{w,x}(z)$$

in a subgraph induced by ancestors of Z as in Figure 4(b). Observing that

$$C(G[An(Z)_G \setminus \{W, X\}]) = \{G[Z]\}$$



(a) Graph F. (b) Subgraph $An(Z_2)_F$.

Figure 5: Graph F and its subgraph $F[An(Z_2)_F]$.

and

$$G[Z] \in C(G[An(Z)_G]) = \{G[X], G[W], G[Z]\},$$

the algorithm proceeds to line 6 and the second term simplifies again

$$P_{w,x}(z) = P(z|w,x).$$

The last term $P_{w,x,z}(y)$ triggers line four, because

$$C(G[\mathbf{V} \setminus \{W,X,Z\}]) = \{G[Y]\}.$$

$G[Y]$ is not a maximal C-component of G, but Y is a node of one of the maximal C-components of G: $\{Y\} \subset \{X,Y\} = \mathbf{S}'$. It holds for the set $\mathbf{S}'$, that

$$G[\mathbf{S}'] \in C(G) = \{G[\{X,Y\}], G[W], G[Z]\}.$$

So it is mandatory to compute $P_x(y)$ from $P(X|w)P(Y|X,w,z)$ in the graph corresponding to Figure 4(c). It should be noted, that this causal effect differs from the original effect $P_x(y)$, because the joint distribution $P(\mathbf{V})$ of observed variables of G is not the same as the distribution $P(X|w)P(Y|X,w,z)$ of the subgraph of the current recursion stage.

Line two is triggered next, and since Y has no observed ancestors in the graph corresponding to 4(c), it follows that

$$P_x(y) = \sum_x P(x|w)P(y|x,w,z).$$

An expression for the original causal effect is obtained by combining the previous results

$$P_x(y) = \sum_{w,z} P(z|w,x)P(w) \sum_x P(y|w,x,z)P(x|w).$$

The result agrees with the result derived in Section 2.

Algorithm 1 can also be used to detect unidentifiability. Let $F = \langle \mathbf{V}, \mathbf{E} \rangle$ be a graph of Figure 5(a) and a causal effect of interest $P_x(y)$, which is to be identified from $P(X,Y,Z_1,Z_2)$. Let the topological ordering of the nodes of F be $Z_1 < X < Z_2 < Y$.

The computation starts from line three

$$\mathbf{W} = (\mathbf{V} \setminus \mathbf{X}) \setminus An(\mathbf{Y})_{F_{\overline{\mathbf{X}}}} = (\{X,Y,Z_1,Z_2\} \setminus \{X\}) \setminus \{X,Z_2,Y\} = \{Z_1\} \neq \emptyset.$$

Z_1 is added to the original intervention, so $P_{z_1,x}(y)$ has to be identified. Line four is triggered next, because

$$C(F[\mathbf{V} \setminus \{Z_1, X\}]) = \{F[Z_2], F[Y]\}.$$

Since $\mathbf{v} \setminus (\{y\} \cup \{z_1, x\}) = \{z_2\}$, two new causal effects have to be identified following expression:

$$\sum_{z_2} P_{z_1,x,y}(z_2) P_{z_1,x,z_2}(y).$$

Consider the first term of the product. Clearly $\mathbf{V} \neq An(Z_2)_F$, so the algorithm proceeds to line two, which means that

$$P_{z_1,x,y}(z_2) = P_{z_1,x}(z_2)$$

in a subgraph formed by ancestors of Z_2 as in Figure 5(b). However, $P_{z_1,x}(z_2)$ is not identifiable, because

$$C(F[An(Z_2)_F \setminus \{Z_1, X\}]) = \{F[Z_2]\} \text{ and } C(F[An(Z_2)_F]) = \{F[An(Z_2)_F]\},$$

which trigger line five. In conclusion, F contains a hedge for $P_{z_1,x}(z_2)$ formed by C-forests $F[Z_2]$ and $F[\{Z_1, Z_2, X\}]$. Thus the original effect $P_x(y)$ is not identifiable.

4. Implementation using R

The programming language R (R Core Team 2015) was chosen for the implementation of Algorithm 1. The R packages **XML** (Temple Lang 2013), **igraph** (Csardi and Nepusz 2006) and **ggm** (Marchetti *et al.* 2015) are utilized repeatedly throughout the implementation.

4.1. Graph files

A graph G induced by the causal model is a crucial argument of Algorithm 1. Many file formats for visualizing graphs are available, each with their own strengths and weaknesses. Some of these formats are very simple, and do not differentiate directed and undirected graphs. Some formats offer excessive features for describing causal models, or they might require handling complex syntax, which can be time consuming.

GraphML (Brandes *et al.* 2002) is a user-friendly file format for graphs. Its features include support for directed graphs and visualizations. GraphML is based on the extensible markup language XML (Maler *et al.* 2004), which makes processing of graphs files almost effortless. One can also include the names of the nodes within the GraphML file itself, so the user is not limited to having to input the node names themselves inside the R environment. Graphical editors for creating GraphML files are freely available for the user. A special function called `parse.graphml` has been developed for processing GraphML files. However, the implementation of Algorithm 1 is not limited to GraphML files alone. Any file format supported by the **igraph** package can be used, as long as the graph follows one of the following notations for bidirected edges.

Bidirected edges can be separated from unidirected edges by using graphical parameters. For this purpose, three distinct notations have been selected to describe bidirected edges, which correspond to unobserved nodes.

The available notations for bidirected edges are shown in Figures 6(a), 6(b) and 6(c). It should be noted, that notations 1 and 2 are almost identical. Because of their similarity, both

(a) Notation 1. (b) Notation 2. (c) Notation 3.

Figure 6: Notations for bidirected edges.

notations 1 and 2 are referred to as `standard` notation. Notation 3, as shown in Figure 6(c) differs from the previous two. It is apparent, that this notation cannot be used as such, because it induces loops in the graph which is not allowed in the context of DAGs. However, GraphML format enables the assignment of parameters for the edges, which in turn allows one to separate these edges from their unidirected counterparts. When using notation 3, one must define a parameter called `description` for the two unidirected edges corresponding to the bidirected edge, and assign its value to `"U"` (Unobserved). Notation 3 is used in the implementation itself, which is why it is referred to as the `internal` notation.

The process of importing GraphML files created by a graphical editor is handled by using the R package **XML**. This package contains the function `xmlParse`, which is utilized to import graph files into R objects. It should be noted, that these objects only reflect their internal C objects and are thus different from ordinary R objects. This means that the memory reserved by the XML objects has to be freed after the files have been imported. Normally R does this automatically.

Algorithm 1 requires only a small portion of the XML content, and the unnecessary content is removed in the process of searching for the important items. Items of importance are those that contain data about the node names, node count, edge count and the values of the `description` parameters of the edges. If notation 1 or 2 of Figures 6(a) and 6(b) was used for the bidirected arcs, it is converted to match the `internal` format of Figure 6(c). The XML search is implemented using the function `getNodeSet` of the **XML** package. This function uses XPath, which is a processing language for XML content search (Simpson 2002).

When the crucial information has been extracted, an **igraph** graph is formed from the remaining content. **igraph** is a tool for visualizing and processing graphs, and it can handle graphs which may contain millions of nodes due to its implementation in C. This package also offers many useful functions related to Algorithm 1, such as determining the ancestors of a node, constructing a topological ordering and generating induced subgraphs from a set of edges or nodes. One of the main goals of **igraph** is the effortless implementation of graph algorithms.

4.2. Distribution objects

An important question regarding the Algorithm 1 of Section 3.2, is how the probability distribution which changes at each recursive stage should be implemented. An intuitive solution is to construct a distribution object, which maintains the terms currently present in the expression. Distribution objects are recursive by construction as is the algorithm itself. In practice this means that when any of the lines four, six or seven is triggered, sub objects are formed, which correspond to the product terms of the expression. These sub objects can further branch into sub objects of their own and so forth. **causaleffect** implements an R class called `probability` to represent the distribution objects.

Multiple attributes have to be set for the distribution objects in order to present the probability distribution precisely. The string vectors `var` and `cond` are one of the most common attributes, because they enable the definition of a simple conditional distribution. A distribution is formed by the variables described in `var` conditioned on those of `cond`. For example, let `p` be a distribution object, and let the values of its attributes be `var = "Y"` and `cond = "X"`. Therefore object `p` represents the conditional distribution $P(Y|X)$.

When the distribution is a product, the individual terms are defined in a list of distribution objects called `children` and a logical variable `recursive` is set to `TRUE` to differentiate this object from those containing only a single term. For example, for a distribution which represents the distribution $P^* = P(Z|X)P(X|Y)P(Y)$ one has to set `children = list(a,b,c)`, where the objects `a`, `b` and `c` represent the distributions $P(Z|X), P(X|Y)$ and $P(Y)$ respectively.

For marginal distributions a string vector `sumset` has been defined. The contents of this vector correspond to the variables which the distribution is to be summed over in the discrete case, or integrated over in the continuous case. In simple situations this parameter is not needed, but often with more complex graphs one encounters instances, where the computation of conditionals is no longer straightorward. Suppose one had to compute the marginal distribution $P^*(X)$ of X from the joint distribution $P^*(X, Y, Z)$ of the previous example. To achieve this, one has to set `sumset = c("Y","Z")` for the matching distribution object, because $P^*(X) = \sum_{Y,Z} P(Z|X)P(X|Y)P(Y)$.

The level of complexity increases further when computing conditionals from distributions which consist of multiple product terms. The previously presented attributes are often insufficient to form an expression for the corresponding distribution object. Consider once more the joint distribution P^*. Computing the marginal conditional distribution $P^*(X|Y)$ results in

$$P^*(X|Y) = \frac{P^*(X,Y)}{P^*(Y)} = \frac{\sum_Z P(Z|X)P(X|Y)P(Y)}{\sum_{X,Z} P(Z|X)P(X|Y)P(Y)} = \frac{P(X|Y)\sum_Z P(Z|X)}{\sum_X P(X|Y)\sum_Z P(Z|X)} = P(X|Y).$$

The implementation is able to handle similar situations, where the expression can easily be simplified using the following procedure. Any term which does not depend on the summation index, will be placed outside of the sum. Next, it is checked whether any expressions can be simplified by changing the order of summation. Corresponding terms are subtracted if possible.

These simplification rules are not sufficient to handle every situation. For example, the expression $\sum_X P(Y|X)P(X)$ cannot be simplified using the procedure above. One cannot remove any terms from within the sum and the summation order is clearly fixed. In situations, where the denominator is necessary in order to correctly form the expression, one needs to include additional attributes called `divisor` and `fraction`. These attributes are similar to the attributes `children` and `recursive` in a sense that `divisor` contains the distribution object that represents the denominator and `fraction` is set to `TRUE` when it is necessary to represent the expression as a fraction.

4.3. Maximal C-components

In Section 3.1 it was shown, that for every causal diagram G there exists a unique set $C(G)$ of maximal C-components of G. To construct this set, one has to begin by determining all bidirected edges of G. Afterwards, a subgraph containing only bidirected edges is formed. This subgraph will contain one or more *components*, which are connected subgraphs of G. Because these components are disjoint and every pair of nodes within a component is connected by a bidirected path, it follows that they must be the maximal C-components of G. The *adjacency matrix* of G is utilized to find the bidirected edges of G.

Definition 8 (adjacency matrix)**.** An *adjacency matrix* of a graph $G = \langle \mathbf{V}, \mathbf{E} \rangle$ is a $n \times n$ matrix $A = [a_{ij}]$, where n is the number of nodes of G, $\mathbf{V} = \{V_1, V_2, \ldots, V_n\}$ and a_{ij} is the number of edges from V_i to V_j.

Because G is directed, its adjacency matrix is not necessarily symmetric. When notation 3 of Figure 6(c) is used to describe the bidirected edges, it is easy to confirm that two nodes V_i and V_j are connected by at least one bidirected edge if and only if $a_{ij} \geq 1$ and $a_{ji} \geq 1$. Thus all bidirected edges can be determined by comparing A to its transpose $A^\top$, and by choosing only those edges which correspond to indices with $a_{ij} \geq 1$ and $a_{ji} \geq 1$.

The subgraph of G containing only bidirected edges is constructed by using the function `subgraph.edges` of the **igraph** package. This function retains all nodes of the input graph, but removes all the edges that were not given as input. The subgraph returned by this function is further divided into components by using the function `decompose.graph` which is also provided by **igraph**.

4.4. Implementation

All necessary preparations have been presented to implement Algorithm 1. Any probability distribution can be represented with a corresponding distribution object, and the adjacency matrix provides a method to determine the maximal C-components of G. Other important methods are provided by the **igraph** package, such as constructing subgraphs and determining the ancestors of a given set of nodes. In this implementation, the input of Algorithm 1 consists of the sets `x` and `y` including the graph G, and returns a `probability` object, which is a list structure that describes the expression of the causal distribution $P_{\mathbf{x}}(\mathbf{y})$ in terms of $P(\mathbf{V})$. The returned object can be further parsed into a character representation.

The R function of Algorithm 1 is called `id`. This function takes five parameters as input: a string vector `y`, a string vector `x`, a distribution object `P`, an **igraph** graph `G` and a string vector `to`. The first four parameters correspond to their mathematical counterparts, namely the vectors $\mathbf{x}$, $\mathbf{y}$, P and G. The last parameter `to` is a string vector representing some topological ordering of the nodes of G. All required set theoretic operations are included in R as the functions `intersect`, `setdiff` and `union`.

The observed portion of `G` is saved as `G.obs`. This graph contains all the observed nodes of G and the edges between them. In addition, the observed nodes are saved into vector `v`, and the ancestors of `y` are saved into vector `anc`. The implementation of each line of Algorithm 1 is presented next.

1: **if** $\mathbf{x} = \emptyset$, **then**
   return $\sum_{v \in \mathbf{v} \setminus \mathbf{y}} P(\mathbf{v})$.

The truth value of the expression $\mathbf{x} = \emptyset$ is determined on line 1. This is done by computing

the length of `x`. If the length is zero, then `id` combines the difference of the sets `v` and `y` with the `sumset` of `P` and returns `P`.

2: **if** $\mathbf{V} \neq An(\mathbf{Y})_G$, **then**
return $\mathbf{ID}(\mathbf{y}, \mathbf{x} \cap An(\mathbf{Y})_G, P(An(\mathbf{Y})_G), G[An(\mathbf{Y})_G])$.

The truth value of the condition on line 2 is determined by computing the length of the vector `setdiff(v, anc)`. If the length is not zero, then `id` is called with the arguments `id(y, intersect(x, anc), P, anc.graph, to)`, where `anc.graph` is the induced subgraph $G[An(\mathbf{Y})_G]$, which is constructed by using the `induced.subgraph` function of the **igraph** package. This function takes a set of nodes and a graph as input, and constructs a subgraph, which retains all of the nodes given as input, and all of the edges between them in the original graph.

3: **let** $\mathbf{W} = (\mathbf{V} \setminus \mathbf{X}) \setminus An(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$.
if $\mathbf{W} \neq \emptyset$, **then**
return $\mathbf{ID}(\mathbf{y}, \mathbf{x} \cup \mathbf{w}, P, G)$.

To construct a vector `w` which represents the node set $\mathbf{W}$, one must first construct the subgraph $G_{\overline{\mathbf{X}}}$. To accomplish this, all incoming edges of $\mathbf{X}$ have to be determined. A useful operator is provided by the **igraph** package to accomplish this. The operator `%->%` can be used to find incoming or outgoing edges of a node. In this case, one finds the incoming nodes of `x` with the command `E(G) [1:length(E(G)) %->% x]`, where `E` is a function that returns all edges of `G`. When the subgraph has been constructed, `w` can also be constructed. If the length of `w` is not zero, then `id` is called with the arguments `id(y, union(x, w), P, G, to)`.

4: **if** $C(G[\mathbf{V} \setminus \mathbf{X}]) = \{G[\mathbf{S}_1], \ldots, G[\mathbf{S}_k]\}$, **then**
return $\sum_{v \in \mathbf{v} \setminus (\mathbf{y} \cup \mathbf{x})} \prod_{i=1}^{k} \mathbf{ID}(\mathbf{s}_i, \mathbf{v} \setminus \mathbf{s}_i, P, G)$.

The set $C(G[\mathbf{V} \setminus \mathbf{X}])$ can be found with the function `c.components`. This function determines the node set of every maximal C-component of the input graph, and returns them as a list `s`. If the length of this list is larger than one, then `id` returns a new distribution object with `sumset = setdiff(v, union(y, x)), recursive = TRUE, children = productlist`, where every object in `productlist` is determined by a new recursive call for every C-component $G[\mathbf{S}_i]$, $i = 1, \ldots, k$ that was found. These components are constructed by calling `id` with the arguments `id(s[[i]], setdiff(v, s[[i]]), P, G, to)`, $i = 1, \ldots, k$.

If the algorithm did not proceed to any of the previous lines, then the additional condition $C(G[\mathbf{V} \setminus \mathbf{X}]) = \{G[\mathbf{S}]\}$ must be true. The node set of the single C-component $G[\mathbf{S}]$ is now saved in the vector `s`, which was previously a list. This means that `s` is replaced by `s[[1]]`.

5: **if** $C(G) = \{G\}$, **then**
throw $\mathbf{FAIL}(G, G[\mathbf{S}])$.

The function `c.components` is utilized again in order to find the maximal C-components of G. If in addition to having only a single C-component this C-component is G itself, then line five is triggered. This is checked by comparing `s` and `v`. If they are equal, then the computation is interrupted by the `stop` function and an error message is produced. The error message describes the C-forests which form the problematic hedge structure for the causal effect of the current recursion stage.

6: **if** $G[\mathbf{S}] \in C(G)$, **then**

return $\sum_{v \in \mathbf{s} \setminus \mathbf{y}} \prod_{V_i \in \mathbf{S}} P(v_i | v_\pi^{(i-1)})$.

If the single C-component found on line four is one of the maximal C-components of G, then the function `id` returns a new distribution object. The `sumset` of this object is set to `setdiff(s, y)`. The distribution is a product so it must also be set, that `recursive = TRUE` for this new object. The objects in the list `children` are determined by new recursive calls for every node V_i in $\mathbf{S}$. The conditioning nodes are the ones that precede V_i in the topological ordering `to`.

7: **if** $(\exists \mathbf{S}')\mathbf{S} \subset \mathbf{S}'$ such that $G[\mathbf{S}'] \in C(G)$, **then**
return $\mathbf{ID}(\mathbf{y}, \mathbf{x} \cap \mathbf{s}', \prod_{V_i \in \mathbf{S}'} P(V_i | V_\pi^{(i-1)} \cap \mathbf{S}', v_\pi^{(i-1)} \setminus \mathbf{s}'), G[\mathbf{S}'])$.

If the single C-component found on line four is not one of the maximal C-components of G, then it must be a subgraph of some maximal C-component $G[\mathbf{S}']$. Vector `s` is replaced by a vector corresponding to the nodes of $\mathbf{S}'$, since the nodes of $\mathbf{S}$ are no longer required. The function `id` is called with the following attributes `id(y, intersect(x, s), probability(recursive = TRUE, children = productlist), s.graph, to)`, where `s.graph` is the induced subgraph $G[\mathbf{S}']$ and every distribution object in `productlist` is constructed by setting `var <- s[i]` and `cond <- v[0:(ind[i]-1)]` for every node V_i in $\mathbf{S}'$.

Algorithm 2 is also implemented in **causaleffect** as the function `idc`. This function iterates through the nodes `z` which it receives as input in addition to the parameters that were previously defined for the `id` function. The d-separation condition on line 1 is checked by using the function `dSep` from the **ggm** package.

5. Package causaleffect

The primary goal of the **causaleffect** package is to provide the implementation described in Section 4. The package also provides a means of importing GraphML files into R while retaining any attributes that have been set for the nodes or edges of the graph.

5.1. Using causaleffect in R

The primary function which serves as a wrapper for the functions `id` and `idc` is called `causal.effect`. This function can be called as

```
causal.effect(y, x, z = NULL, G, expr = TRUE)
```

where the parameters `y`, `x` and `G` are identical to those of `id`. The parameter `z` is optional and it is used to represent the conditioning variables of `idc`. The initial probability object `P` which is a parameter of `id` does not have to be specified by the user. In essence, `causal.effect` starts from an empty distribution object, and gradually builds the final expression if possible. Also, the topological ordering `to` of the function `id` is automatically generated by the `topological.sort` function of the **igraph** package. It is verified, that the vectors `y`, `x` and `z` actually contain nodes that are present in `G`. If `G` is not a DAG then `causal.effect` will also terminate. The last parameter `expr` is a logical variable. If assigned to `TRUE`, `causal.effect` will return the expression in LaTeX syntax. Otherwise, the `probability` object used internally by `id` is returned, which can be manually parsed by the user to gain the desired output. The function `get.expression` is also provided to get a string representation of a `probability` object. This function currently supports LaTeX syntax only.

First, **causaleffect** is loaded to demonstrate the usage of the package.

```
R> library("causaleffect")
```

The `causal.effect` function can be utilized without first importing a graph file. One can utilize the **igraph** package to construct graphs within R itself. This is demonstrated by replicating some of the graphs of Section 3.3. The graph of Figure 1 is created as follows.

```
R> library("igraph")
R> fig1 <- graph.formula(W -+ X, W -+ Z, X -+ Z, Z -+ Y, X -+ Y, Y -+ X,
+    simplify = FALSE)
R> fig1 <- set.edge.attribute(graph = fig1, name = "description",
+    index = c(5,6), value = "U")
R> ce1 <- causal.effect(y = "Y", x = "X", z = NULL, G = fig1, expr = TRUE)
R> ce1

[1] "\\left(\\sum_{W,Z}P(W)P(Z|W,X)\\left(\\sum_{X}P(Y|W,X,Z)P(X|W)\\right)\\right)"
```

Here `X -+ Z` denotes a directed edge from X to Z. The argument `simplify = FALSE` allows the insertion of duplicate edges for the purposes of forming bidirected arcs. Recalling the `internal` notation from Section 4.1 we must denote the unidirected edges that correspond to a bidirected edge with a special `description` parameter, and assign its value to `"U"`. This can be done with the `set.edge.attribute` function of the **igraph** package. Finally, the expression for the interventional distribution is obtained by using the `causal.effect` function. Usually one needs to apply the standard R function `cat` to obtain the expression with only singular slash symbols.

```
R> cat(ce1)

\left(\sum_{W,Z}P(W)P(Z|W,X)\left(\sum_{X}P(Y|W,X,Z)P(X|W)\right)\right)
```

To observe unidentifiability, the graph of Figure 5(a) is also constructed and an attempt is made to identify $P_x(y)$.

```
R> fig5 <- graph.formula(Z_1 -+ X, X -+ Z_2, Z_2 -+ Y, Z_1 -+ X, X -+ Z_1,
+    Z_1 -+ Z_2, Z_2 -+ Z_1, Z_1 -+ Y, Y -+ Z_1, X -+ Y, Y -+ X,
+    simplify = FALSE)
R> fig5 <- set.edge.attribute(graph = fig5, name = "description",
+    index = 4:11, value = "U")
R> causal.effect(y = "Y", x = "X", z = NULL, G = fig5, expr = TRUE)

Error: Graph contains a hedge formed by C-forests of nodes:
  {Z_1,X,Z_2} and {Z_2}.
```

The identification fails in this case due to a hedge present in the graph.

Another function provided by **causaleffect** is `parse.graphml` which can be called as

```
parse.graphml(file, format = c("standard", "internal"), nodes = c(),
  use.names = TRUE)
```

Parameter `file` is the path to the GraphML file the user wishes to convert into an **igraph** graph. Parameter `format` should match the notation that is used to denote bidirected edges in the input graph. The vector `nodes` can be used to give names to the nodes of the graph if they have not been specified in the file itself or alternatively, to replace them. Finally, `use.names` is a logical vector indicating whether the names of the nodes should be read from the file or not.

We provide an example GraphML file in the replication materials to demonstrate the use of the `parse.graphml` function. The file `g1.graphml` contains the graph of Figure 1 in `standard` notation. This means that we do not have to provide names for the nodes or set the unidentified edges manually. First, we read the file into R. This produces several warnings which can be ignored because they are related to the visual attributes created by the graphical editor that was used to produce `g1.graphml`. These attributes play no role in the identification of $P_x(y)$. We omit these warnings from the code for clarity.

```
R> gml1 <- parse.graphml("g1.graphml", format = "standard")
R> ce2 <- causal.effect(y = "Y", x = "X", z = NULL, G = gml1, expr = TRUE)
R> cat(ce2)

\left(\sum_{W,Z}P(W)P(Z|W,X)\left(\sum_{X}P(Y|W,X,Z)P(X|W)\right)\right)
```

We see that the result agrees with the one derived from the manually constructed graph.

For conditional causal effects, we simply utilize the parameter `z` of the `causal.effect` function. For example, we can obtain the formula for $P_x(z|w)$ in the graph of Figure 1.

```
R> cond1 <- causal.effect(y = "Z", x = "X", z = "W", G = gml1, expr = TRUE)
R> cat(cond1)

\frac{P(Z|W,X)}{\left(\sum_{Z}P(Z|W,X)\right)}
```

In mathematical notation the result reads

$$\frac{P(z|w,x)}{\sum_z [P(z|w,x)]}.$$

This is a typical case where the resulting expression is slightly awkward due to the incompleteness of the simplification rules. However, in this case it is easy to see that the expression can be simplified into $P(z|w,x)$.

5.2. A complex expression

The conditional distributions $P(v_i|v_\pi^{(i-1)})$ that are computed on line 6 can sometimes produce difficult expressions when causal effects are determined from complex graphs. This is a result of the simplification rules which were described in the previous section, and their inability to handle every situation. The graph G of Figure 7 serves to demonstrate this phenomenon. An attempt is made to identify $P_x(z_1, z_2, z_3, y)$ in this graph.

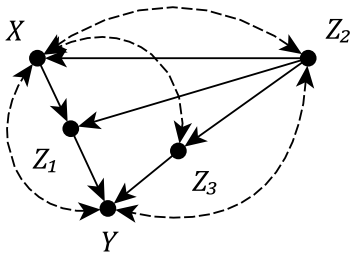


Figure 7: An example of a graph, where an identifiable causal effect results in a complex expression.

Tian (2002) proved this effect to be identifiable, and showed that its expression is

$$P_x(z_1, z_2, z_3, y) = P(z_1|x, z_2) \sum_x P(y, z_3|x, z_1, z_2) P(x, z_2).$$

When applying Algorithm 1 to this causal effect, it is necessary to compute a conditional distribution $P^*(Y|Z_2, Z_3)$, where

$$P^*(y, z_2, z_3) = \sum_x P(y|z_2, x, z_3, z_1) P(z_3|z_2, x) P(x|z_2) P(z_2)$$

and P is the joint distribution of the observed variables of G. Now, the function `causal.effect` is applied as follows.

```
R> fig7 <- graph.formula(X -+ Z_1, Z_1 -+ Y, Z_3 -+ Y, Z_2 -+ X,
+    Z_2 -+ Z_1, Z_2 -+ Z_3, X -+ Y, Y -+ X, X -+ Z_3, Z_3 -+ X,
+    X -+ Z_2, Z_2 -+ X, Y -+ Z_2, Z_2 -+ Y, simplify = FALSE)
R> fig7 <- set.edge.attribute(graph = fig7, name = "description",
+    index = 7:14, value = "U")
R> ce3 <- causal.effect(y = c("Z_1", "Z_2", "Z_3", "Y"), x = "X",
+    z = NULL, G = fig7, expr = TRUE)
R> cat(ce3)
```

This results in the expression

$$P(z_1|z_2, x) \frac{(\sum_x P(y|z_2, x, z_3, z_1) P(z_3|z_2, x) P(x|z_2) P(z_2))}{\left(\sum_{x,y} P(y|z_2, x, z_3, z_1) P(z_3|z_2, x) P(x|z_2) P(z_2)\right)} \times \left(\sum_{x,z_3,y} P(y|z_2, x, z_3, z_1) P(z_3|z_2, x) P(x|z_2) P(z_2)\right) P(z_3|z_2)$$

This result is clearly more cumbersome than the one determined by Tian. However, it can be shown that this expression is correct by using do-calculus. Because the set $\{X, Z_2\}$ d-separates all paths from Z_1 to Z_3, it follows that $(Z_3 \perp\!\!\!\perp Z_1|X, Z_2)_G$, so

$$\begin{aligned} & P(z_1|z_2, x) \sum_x P(y|z_2, x, z_3, z_1) P(z_3|z_2, x) P(x|z_2) P(z_2) \\ = & P(z_1|z_2, x) \sum_x P(y|z_2, x, z_3, z_1) P(z_3|z_2, x, z_1) P(x, z_2) \\ = & P(z_1|z_2, x) \sum_x P(y, z_3|z_2, x, z_1) P(x, z_2), \end{aligned}$$

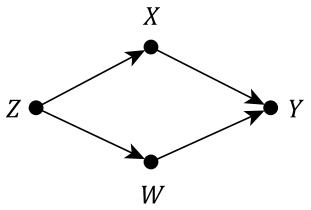


Figure 8: An example of a graph with additional conditional independences.

where the second equality is due to the conditional independence of Z_1 and Z_3 given X and Z_2. The last line is equivalent with Tian's expression up to the ordering of terms. It can be shown, that the remaining terms are subtracted from the expression.

$$\begin{aligned}&\frac{P(z_3|z_2)\sum_{x,z_3,y}P(y|z_2,x,z_3,z_1)P(z_3|z_2,x)P(x|z_2)P(z_2)}{\sum_{x,y}P(y|z_2,x,z_3,z_1)P(z_3|z_2,x)P(x|z_2)P(z_2)}\\ =&\frac{P(z_3|z_2)P(z_2)}{\sum_{x,y}P(y|z_2,x,z_3,z_1)P(z_3|z_2,x)P(x,z_2)}.\end{aligned}$$

By applying the same logic to the denominator, it follows that

$$\frac{P(z_3|z_2)P(z_2)}{\sum_{x,y}P(y|z_2,x,z_3,z_1)P(z_3|z_2,x)P(x,z_2)}=\frac{P(z_3|z_2)P(z_2)}{\sum_{x,y}P(y,z_3|z_2,x,z_1)P(x,z_2)}.$$

By using the conditional independence of Z_1 and Z_3 given X and Z_2 one gets

$$\begin{aligned}&\frac{P(z_3,z_2)}{\sum_x P(z_3|z_2,x,z_1)P(x,z_2)}=\frac{P(z_3,z_2)}{\sum_x P(z_3|z_2,x)P(x,z_2)}\\ =&\frac{P(z_3,z_2)}{\sum_x P(z_3|z_2,x)P(x,z_2)}=\frac{P(z_3,z_2)}{\sum_x P(z_3,z_2,x)}=\frac{P(z_3,z_2)}{P(z_3,z_2)}=1.\end{aligned}$$

The expression produced by `causal.effect` is correct despite its complexity.

5.3. d-separation

Algorithm 1 does not utilize every possible independence property of a given graph G. For example, the conditional distribution of line six is conditioned on all nodes preceding V_i in the topological ordering π, even though at least some nodes on paths preceding V_i are often d-separated by some sets of nodes. In these cases, the nodes that are d-separated with V_i could be excluded from the expression, because they are conditionally independent from V_i in G. This situation is demonstrated by determining the expression of $P_{x,w}(y)$ in the graph G of Figure 8.

The function `causal.effect` is utilized

```
R> fig8 <- graph.formula(z -+ x, z -+ w, x -+ y, w -+ y)
R> ce3 <- causal.effect(y = "y", x = c("x", "w"), z = NULL, G = fig8,
+    expr = TRUE)
R> cat(ce3)
```

The function returns $P(y|x,w)$ even though Algorithm 1 would return $P(y|z,x,w)$. This is possible because $(Y \perp\!\!\!\perp Z|X,W)_G$. This means that our implementation is able to simplify the expression into $P(y|x,w)$.

6. Discussion

We have introduced R package **causaleffect** for deriving expressions of joint interventional distributions in causal models. The task is a specific but important part of causal inference. We believe that our implementation has two practical use cases. First, **causaleffect** can be simply used to derive expressions of interventional distributions for complex causal models or to check manual derivations. This is an important step in the estimation of causal effects in complicated settings (Karvanen 2015). Second, **causaleffect** can be used as a building block in simulation studies and automated systems where identifiability needs to be checked for a large number of causal models. An example of this kind usage is already given by Hyttinen *et al.* (2015).

The efficiency of the presented implementation **causaleffect** could be analyzed further for example by simulation studies. However, an attempt to maximize performance was made by utilizing the most efficient packages available for the processing of graph files and for the objects corresponding to them. The existing simplification rules of the expressions could also be further improved, but it should be noted that sometimes the more complex expression can prove useful.

There have been many recent developments in the field of causality resulting in graph theoretic algorithms similar to **ID** and **IDC**. These include for example:

- Causal effect z-identifiability algorithm ID^{Z} (Bareinboim and Pearl 2012). z-identifiability deals with a situation, where it is possible to utilize a set $\mathbf{Z}$ that is disjoint from $\mathbf{X}$ to achieve identifiability.
- Causal effect transportability algorithm sID (Bareinboim and Pearl 2013a). Transportability means, that results obtained from experimental data can be generalized into a larger population, where only observational studies are applicable.
- Causal effect meta-transportability algorithm μsID (Bareinboim and Pearl 2013b). Meta-transportability is an extension of the concept of transportability, where the results are to be generalized from multiple experimental studies simultaneously.
- Counterfactual and conditional counterfactual identifiability algorithms ID* and IDC* (Shpitser and Pearl 2007).

The work presented in this paper could be utilized to implement these algorithms.

References

Bareinboim E, Pearl J (2012). "Causal Inference by Surrogate Experiments: z-identifiability." In N de Freitas, K Murphy (eds.), *Proceedings of the Twenty-Eight Conference on Uncertainty in Artificial Intelligence*, pp. 113–120. AUAI Press.

Bareinboim E, Pearl J (2013a). "A General Algorithm for Deciding Transportability of Experimental Results." *Journal of Causal Inference*, **1**, 107–134.

Bareinboim E, Pearl J (2013b). "Meta-Transportability of Causal Effects: A Formal Approach." In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 135–143.

Blackwell M (2013). ***causalsens**: Selection Bias Approach to Sensitivity Analysis for Causal Effects*. R package version 0.1, URL http://CRAN.R-project.org/package=causalsens.

Bontempi G, Olsen C, Flauder M (2015). ***D2C**: Predicting Causal Direction from Dependency Features*. R package version 1.2.1, URL http://CRAN.R-project.org/package=D2C.

Brandes U, Eiglsperger M, Herman I, Himsolt M, Marshall M (2002). "GraphML Progress Report Structural Layer Proposal." In P Mutzel, M Jünger, S Leipert (eds.), *Graph Drawing*, volume 2265 of *Lecture Notes in Computer Science*, pp. 501–512. Springer-Verlag. URL http://dx.doi.org/10.1007/3-540-45848-4_59.

Carnegie NB, Harada M, Hill J (2015). ***treatSens**: A Package to Assess Sensitivity of Causal Analyses to Unmeasured Confounding*. New York, NY. R package version 1.1, URL http://CRAN.R-project.org/package=treatSens.

Csardi G, Nepusz T (2006). "The **igraph** Software Package For Complex Network Research." *InterJournal*, **Complex Systems**, 1695. URL http://igraph.org.

Dandine-Roulland C (2015). ***ASPBay**: Bayesian Inference on Causal Genetic Variants using Affected Sib-Pairs Data*. R package version 1.2, URL http://CRAN.R-project.org/package=ASPBay.

Dawid AP (1979). "Conditional Independence in Statistical Theory." *Journal of the Royal Statistical Society B*, **41**, 1–31.

Ding P (2012). ***ImpactIV**: Identifying Causal Effect for Multi-Component Intervention Using Instrumental Variable Method*. R package version 1.0, URL http://CRAN.R-project.org/package=ImpactIV.

Glynn A, Quinn K (2010). ***CausalGAM**: Estimation of Causal Effects with Generalized Additive Models*. R package version 0.1-3, URL http://CRAN.R-project.org/package=CausalGAM.

Huang Y, Valtorta M (2006). "Pearl's Calculus of Intervention is Complete." In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, pp. 217–224. AUAI Press.

Hyttinen A, Eberhardt F, Järvisalo M (2015). "Do-calculus When the True Graph is Unknown." In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, pp. 395–404. AUAI Press.

Kalisch M, Mächler M, Colombo D, Maathuis MH, Bühlmann P (2012). "Causal Inference Using Graphical Models with the R Package **pcalg**." *Journal of Statistical Software*, **47**(11), 1–26. URL http://www.jstatsoft.org/v47/i11/.

Karvanen J (2015). "Study Design in Causal Models." *Scandinavian Journal of Statistics*, **42**(2), 361–377.

Kashin K, Glynn A, Ichino N (2014). ***qualCI**: Causal Inference with Qualitative and Ordinal Information on Outcomes*. R package version 0.1, URL http://CRAN.R-project.org/package=qualCI.

Kim IS, Imai K (2014). ***wfe**: Weighted Linear Fixed Effects Regression Models for Causal Inference*. R package version 1.3, URL http://CRAN.R-project.org/package=wfe.

King G, Lucas C, Nielsen R (2015). ***MatchingFrontier**: R Package for Computing the Matching Frontier*. R package version 1.0.0, URL http://CRAN.R-project.org/package=MatchingFrontier.

Koller D, Friedman N (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

Luo X, Small D, shan Li C, Rosenbaum P (2011). ***cin**: Causal Inference for Neuroscience*. R package version 0.1, URL http://CRAN.R-project.org/package=cin.

Maler E, Paoli J, Sperberg-McQueen CM, Yergeau F, Bray T (2004). "Extensible Markup Language (XML) 1.0 (Third Edition)." *Technical report*, W3C. URL http://www.w3.org/TR/2004/REC-xml-20040204.

Marchetti GM, Drton M, Sadeghi K (2015). ***ggm**: Functions for Graphical Markov Models*. R package version 2.3, URL http://CRAN.R-project.org/package=ggm.

Meinshausen N (2015). ***InvariantCausalPrediction**: Invariant Causal Prediction*. R package version 0.3-1, URL http://CRAN.R-project.org/package=InvariantCausalPrediction.

Millstein J (2015). ***cit**: Causal Inference Test*. R package version 1.4, URL http://CRAN.R-project.org/package=cit.

Neto EC, Yandell BS (2014). ***qtlnet**: Causal Inference of QTL Networks*. R package version 1.3.6, URL http://CRAN.R-project.org/package=qtlnet.

Pearl J (1995). "Causal Diagrams for Empirical Research." *Biometrika*, **82**, 669–688.

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

Peters J, Ernest J (2015). ***CAM**: Causal Additive Model (CAM)*. R package version 1.0, URL http://CRAN.R-project.org/package=CAM.

Quinn KM (2012). ***SimpleTable**: Bayesian Inference and Sensitivity Analysis for Causal Effects from 2 x 2 and 2 x 2 x K Tables in the Presence of Unmeasured Confounding*. R package version 0.1-2, URL http://CRAN.R-project.org/package=SimpleTable.

Ratkovic M (2015). ***SVMMatch**: Causal Effect Estimation and Diagnostics with Support Vector Machines*. R package version 1.1, URL http://CRAN.R-project.org/package=SVMMatch.

R Core Team (2015). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Saul B (2015). ***inferference**: Methods for Causal Inference with Interference*. R package version 0.4.61, URL http://CRAN.R-project.org/package=inferference.

Schutte S, Donnay K (2015). ***mwa**: Causal Inference in Spatiotemporal Event Data*. R package version 0.4.1, URL http://CRAN.R-project.org/package=mwa.

Shpitser I, Pearl J (2006a). "Identification of Conditional Interventional Distributions." In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence (UAI2006)*, pp. 437–444. AUAI Press.

Shpitser I, Pearl J (2006b). "Identification of Joint Interventional Distributions in Recursive Semi-Markovian Causal Models." In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, pp. 1219–1226. AAAI Press.

Shpitser I, Pearl J (2007). "What Counterfactuals Can Be Tested." In *Proceedings of Twenty Third Conference on Uncertainty in Artificial Intelligence*, pp. 352–359. Vancouver, Canada.

Simpson JE (2002). *XPath and XPointer: Locating Content in XML Documents*. O'Reilly & Associates, Sebastopol, CA, USA.

Tan Z, Shu H (2013). ***iWeigReg**: Improved Methods for Causal Inference and Missing Data Problems*. R package version 1.0.

Temple Lang D (2013). *XML: Tools for Parsing and Generating XML within R and S-PLUS*. R package version 3.98-1.1, URL http://CRAN.R-project.org/package=XML.

Textor J, Hardt J, Knüppel S (2011). "DAGitty: A Graphical Tool for Analyzing Causal Diagrams." *Epidemiology*, **22**, 745.

Tian J (2002). *Studies in Causal Reasoning and Learning*. Phd thesis, Department of Computer Science, University of California, Los Angeles.

Tian J, Pearl J (2002). "A General Identification Condition For Causal Effects." In *Proceedings of the 18th National Conference on Artificial Intelligence*, pp. 567–573. AAAI/The MIT Press.

Tian J, Pearl J (2003). "On the Identification of Causal Effects." *Technical report*, Department of Computer Science, University of California, Los Angeles. R-290-L.

Tikka S, Karvanen J (2017). "Identifying Causal Effects with the R Package causaleffect." *Journal of Statistical Software*, **76**(12), 1–30.

Tingley D, Yamamoto T, Hirose K, Keele L, Imai K (2014). "**mediation**: R Package for Causal Mediation Analysis." *Journal of Statistical Software*, **59**(5).

Verma TS (1993). "Graphical Aspects of Causal Models." *Technical report*, Department of Computer Science, University of California, Los Angeles. R-191.

A. Graphs, causal models and causal effects

A.1. Graphs

The definitions that are presented here follow those of (Koller and Friedman 2009). *Graph* is an ordered pair $G = \langle \mathbf{V}, \mathbf{E} \rangle$, where $\mathbf{V}$ and $\mathbf{E}$ are sets such that

$$\mathbf{E} \subset \{\{X, Y\} \mid X \in \mathbf{V}, Y \in \mathbf{V}, X \neq Y\}.$$

The elements of $\mathbf{V}$ are the nodes of G, and the elements of $\mathbf{E}$ are the edges of G. A graph $F = \langle \mathbf{V}', \mathbf{E}' \rangle$ is a *subgraph* of G if $\mathbf{V}' \subset \mathbf{V}$ and $\mathbf{E}' \subset \mathbf{E}$. This is denoted as $F \subset G$. A graph G is *directed* if the set $\mathbf{E}$ consists of ordered pairs (X, Y). In a directed graph, node V_2 is a *child* of node V_1 if G contains an edge from V_1 to V_2, which means that $(V_1, V_2) \in \mathbf{E}$. Respectively V_2 is a *parent* of V_1 if $(V_2, V_1) \in \mathbf{E}$. The child-parent relationship is often denoted as $V_1 \rightarrow V_2$, where V_1 is a parent of V_2 and V_2 is a child of V_1. This can also be notated as $V_2 \leftarrow V_1$.

Let $n \geq 1$, $\mathbf{V} = \{V_1, \ldots, V_n\}$ and $V_i \neq V_j$ for all $i \neq j$. If $n > 1$, then the graph $H = \langle \mathbf{V}, \mathbf{E} \rangle$ is a *path* if

$$\mathbf{E} = \{\{V_1, V_2\}, \{V_2, V_3\}, \ldots, \{V_{n-1}, V_n\}\}$$

or if

$$\mathbf{E} = \{\{V_1, V_2\}, \{V_2, V_3\}, \ldots, \{V_{n-1}, V_n\}, \{V_n, V_1\}\}.$$

In the first case, H is a path from V_1 to V_n. In the second case H is a *cycle*. If $n = 1$, then $H = \langle \{V_1\}, \emptyset \rangle$ is also a path. A path H is a *directed path* if all of its edges are directed and point to the same direction, which means that either

$$\mathbf{E} = \{(V_1, V_2), (V_2, V_3), \ldots, (V_{n-1}, V_n)\}$$

or

$$\mathbf{E} = \{(V_1, V_2), (V_2, V_3), \ldots, (V_{n-1}, V_n), (V_n, V_1)\}.$$

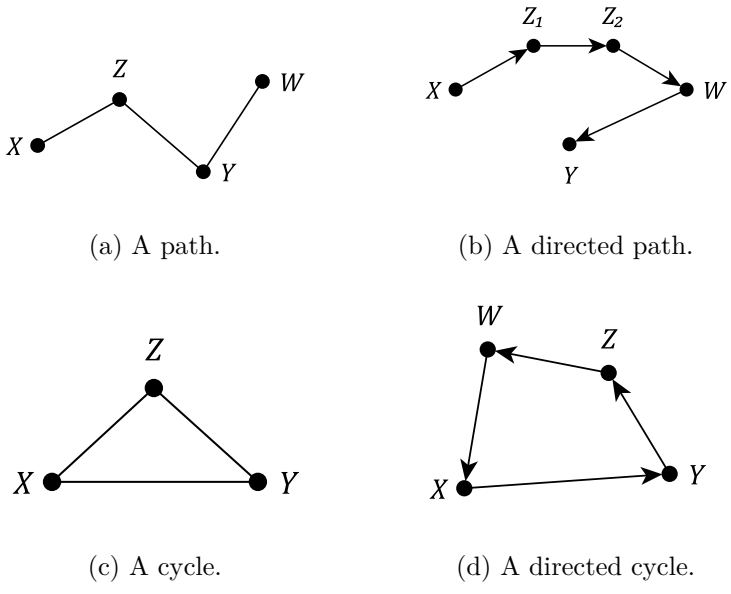
A node V_2 is a *descendant* of V_1 in G, if there exists a directed path H from V_1 to V_2 and $H \subset G$. Respectively, V_2 is an *ancestor* of V_1 in G, if there exists a directed path H from V_2 to V_1 and $H \subset G$. If a graph G does not contain any cycles, it is *acyclic*. A graph $G = \langle \mathbf{V}, \mathbf{E} \rangle$ is *connected* if there exists a path $H \subset G$ between every pair of nodes $V_i, V_j \in \mathbf{V}$. Examples of paths are cycles are presented in Figure 9.

If a graph is directed it is also possible to consider its subgraphs as undirected graphs, when all of the edges of the graph are regarded as undirected edges. For example, a directed graph contain paths, even if it does not contain any directed paths. The directed graph in Figure 10 contains a path connecting the nodes X and Y, even though they are not connected by a directed path.

Let $G = \langle \mathbf{V}, \mathbf{E} \rangle$ be a graph and $\mathbf{Y} \subset \mathbf{V}$. Assume that the nodes of $\mathbf{Y}$ correspond to some observed variables, and that the set $\mathbf{V}$ can also contain nodes, which in turn correspond to some unobserved variables. Then the abbreviations $Pa(\mathbf{Y})_G$, $An(\mathbf{Y})_G$, and $De(\mathbf{Y})_G$ denote the set of observable parents, ancestors and descendants of the node set $\mathbf{Y}$ while also containing $\mathbf{Y}$.

A.2. Causal model

Causal model can be used to describe the functional relationships between variables of interest. In addition, the model enables the formal treatment of actions or interventions on the



(a) A path. (b) A directed path.

(c) A cycle. (d) A directed cycle.

Figure 9: Directed and undirected paths and cycles.

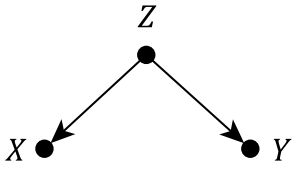


Figure 10: An undirected path in a directed graph.

variables of the model. Judea Pearl defined the deterministic causal model and its probabilistic counterpart (Pearl 2009, p. 203-205), which are presented in this section.

Definition 9 (Causal Model, (Pearl 2009) 7.1.1)**.** A *causal model* is a triple

$$M = \langle \mathbf{U}, \mathbf{V}, \mathbf{F} \rangle,$$

where:

1. $\mathbf{U}$ is a set of background variables that are determined by factors outside the model;
2. $\mathbf{V}$ is a set $\{V_1, V_2, \ldots, V_n\}$ of variables, called endogenous, that are determined by variables in the model – that is, variables in $\mathbf{U} \cup \mathbf{V}$; and
3. $\mathbf{F}$ is a set of functions $\{f_{V_1}, f_{V_2}, \ldots, f_{V_n}\}$ such that each f_{V_i} is a mapping from (the respective domains of) $\mathbf{U} \cup (\mathbf{V} \setminus V_i)$ to V_i, and such that the entire set $\mathbf{F}$ forms a mapping from $\mathbf{U}$ to $\mathbf{V}$. In other words, each f_i tells the value of V_i given the values of all other variables in $\mathbf{U} \cup \mathbf{V}$, and the entire set $\mathbf{F}$ has a unique solution $V(u)$. Symbolically, the set of equations $\mathbf{F}$ can be represented by writing

$$v_i = f_{V_i}(\mathbf{pa}_{V_i}, \mathbf{u}_{V_i}), \quad i = 1, \ldots, n,$$

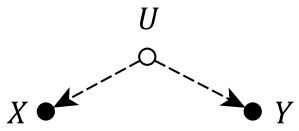


Figure 11: Example notation of unobserved edges.

Figure 12: Notation for bidirected edges.

where $\mathbf{pa}_i$ is any realization of the unique minimal set of variables $\mathbf{PA}_{V_i}$ in $\mathbf{V} \setminus V_i$ (connoting parents) sufficient for representing f_i. Likewise, $\mathbf{U}_{V_i} \subseteq \mathbf{U}$ stand for the unique minimal set of variables in $\mathbf{U}$ sufficient for representing f_i.

For each causal model M there is a corresponding graph $G = \langle \mathbf{W}, \mathbf{E} \rangle$. The node set $\mathbf{W}$ contains a node for each observed and unobserved variable of M. The edge set $\mathbf{E}$ is determined by the functional relationships between the variables of $\mathbf{V}$ and $\mathbf{U}$ in the causal model M. The set $\mathbf{E}$ contains an edge from X to Y if $X \in \mathbf{PA}_Y$, which means that there is an edge coming into V_i from every node required to uniquely define f_{V_i}. Likewise, the set $\mathbf{E}$ contains an edge from U to every node V_i such that $U \in \mathbf{U}_{V_i}$.

The definition of causal model does not set any limitations for the unobserved variables. Thus any unobserved node can be a parent of an arbitrary number of observed nodes. If every unobserved node is a parent of exactly two observed nodes, then the causal model is a *semi-Markovian causal model*. Verma (1993) showed, that for any causal model with unobserved variables one can construct a semi-Markovian causal model that encodes the same set of conditional independences. This is why only semi-Markovian models are considered in this paper.

The edges coming from unobserved variables are sometimes denoted as in Figure 11. However, it is common not to include the unobserved nodes in the visual representation of the graph, which serves to simplify the notation. Instead, it is said that there exists a *bidirected edge* between X and Y, which corresponds to the effect of the unobserved variable. Thus the notation of Figure 12 is utilized instead of the one in Figure 11.

This notation is used in (Huang and Valtorta 2006; Shpitser and Pearl 2006b; Tian 2002). It should be noted, that a bidirected edge is not the same as two directed edges between two nodes, as this would induce a cycle in the graph which is not allowed. Next, the definition of the causal model is expanded by defining a probability distribution for the unobserved variables.

Definition 10 (Probabilistic Causal Model, (Pearl 2009) 7.1.6)**.** A *Probabilistic causal model* is a pair

$$M = \langle M_D, P(\mathbf{U}) \rangle,$$

where M_D is a (deterministic) causal model and $P(\mathbf{U})$ is the joint distribution of the variables in $\mathbf{U}$.

Henceforth in the paper, the term causal model refers to a probabilistic semi-Markovian causal model without exception. Similarly, any graphs discussed will also refer to the graphs induced by these causal models. A graph G induced by a causal model is strongly related to the joint distribution P of all variables in the model, where $P = \prod_{i=1}^{n} P(v_i|pa^*(V_i)_G) \prod_{j=1}^{k} P(u_j)$, and $Pa^*(.)_G$ also contains all unobserved parents. If this relationship holds, then G is an *I-map* (independence map) of P. Independence properties of G and P are closely related through the following definition

Definition 11 (d-separation, (Pearl 2009) 1.2.3)**.** Let $H = \langle \mathbf{V}, \mathbf{E} \rangle$ be a path and a set $\mathbf{Z} \subset \mathbf{V}$. H is said to be *d-separated* by $\mathbf{Z}$ in G, if and only if either

1. H contains a chain $I \to M \to J$ or a fork $I \leftarrow M \to J$, where $M \in \mathbf{Z}$ and $I, J \in \mathbf{V}$., or
2. H contains an inverted fork $I \to M \leftarrow J$, where $De(M)_G \cap \mathbf{Z} = \emptyset$.

Disjoint sets $\mathbf{X}$ and $\mathbf{Y}$ are said to be d-separated by $\mathbf{Z}$ in G if every path from $\mathbf{X}$ to $\mathbf{Y}$ is d-separated by $\mathbf{Z}$ in G.

If $\mathbf{X}$ and $\mathbf{Y}$ are d-separated by $\mathbf{Z}$ in G, then $\mathbf{X}$ is independent of $\mathbf{Y}$ given $\mathbf{Z}$ in every P for which G is an I-map of P. The notation of (Dawid 1979) is used to denote this statement as $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})_G$.

A.3. Causal effects

Interventions on a causal model alter the functional relationships between its variables. Any intervention $do(\mathbf{X} = \mathbf{x})$ on a causal model M produces a new model $M_{\mathbf{x}} = \langle \mathbf{U}, \mathbf{V}, \mathbf{F}_{\mathbf{x}}, P(\mathbf{U}) \rangle$, where $\mathbf{F}_{\mathbf{x}}$ is obtained by replacing $f_X \in \mathbf{F}$ for each $X \in \mathbf{X}$ with a constant function, where the constants are defined as the $\mathbf{x}$ values of $do(\mathbf{X} = \mathbf{x})$. It is now feasible to formalize the notion of causal effects as follows.

Definition 12 (Causal Effect, (Shpitser and Pearl 2006b))**.** Let $M = \langle \mathbf{U}, \mathbf{V}, \mathbf{F}, P(\mathbf{U}) \rangle$ be a causal model and $\mathbf{Y}, \mathbf{X} \subset \mathbf{V}$. The *causal effect* of $do(\mathbf{X} = \mathbf{x})$ on the set $\mathbf{Y}$ in M is the marginal distribution of $\mathbf{Y}$ in $M_{\mathbf{x}}$, which is noted by $P(\mathbf{Y}|do(\mathbf{X} = \mathbf{x})) = P_{\mathbf{x}}(\mathbf{Y})$.

For every action $do(\mathbf{X} = \mathbf{x})$ it is required that $P(\mathbf{x}|Pa(\mathbf{X})_G \setminus \mathbf{X}) > 0$. This limitation ensures that $P_{\mathbf{x}}(\mathbf{V})$ and its marginals are well defined. The restriction stems from the fact that it is unfeasible to force $\mathbf{X}$ to attain values which cannot be observed. No inference can be made from the distribution of such an intervention using observational data.

Definition 13 (Causal Effect Identifiability, (Shpitser and Pearl 2006b) 2)**.** Let $G = \langle \mathbf{V}, \mathbf{E} \rangle$ be a graph and $\mathbf{Y}, \mathbf{X} \subset \mathbf{V}$. The causal effect of $do(\mathbf{X} = \mathbf{x})$ on the set $\mathbf{Y}$, where $\mathbf{Y} \cap \mathbf{X} = \emptyset$, is *identifiable* in G if $P^1_{\mathbf{x}}(\mathbf{Y}) = P^2_{\mathbf{x}}(\mathbf{Y})$ for every pair of causal models M^1 and M^2 such that $P^1(\mathbf{V}) = P^2(\mathbf{V})$ and $P^1(\mathbf{x}|Pa(\mathbf{X})_G \setminus \mathbf{X}) > 0$.

It is often impossible to show that a causal effect is identifiable by using solely the definition, because one would have to compare every causal model that agree on the distribution of the

observed variables. However, the definition serves as a tool to prove unidentifiability in certain cases by constructing two causal models with the same induced graph and observational distribution, and by showing further that the interventional distributions differ. The reader is referred to (Shpitser and Pearl 2006b) for examples.

Affiliation:

Santtu Tikka
Department of Mathematics and Statistics
Faculty of Mathematics and Science
University of Jyvaskyla
P.O.Box 35, FI-40014, Finland
E-mail: santtu.tikka@jyu.fi