Exploring causal inference using the DoWhy library

Simon Oergaard, Sascha Valentin Brown, Emilie Jessen, and Luc Voorend
Niels Bohr Institute

(Dated: March 22, 2024)

Causal inference plays a pivotal role in understanding the effects of interventions and treatments in various domains such as healthcare, economics, and social sciences. The `DoWhy` library provides a comprehensive framework for conducting causal inference tasks in Python, offering users a systematic approach to model causality, estimate causal effects, and assess the validity of causal claims. In this paper, we delve into the framework, mathematics, and underlying assumptions of the `DoWhy` library to provide a thorough understanding of its functionalities and applications. As a proof of concept, we apply the `DoWhy` library to a mock experiment investigating the causal effect of discounts on beer sales at the physics Friday bar 'Caféen?'. In this context, we describe various benefits of `DoWhy` compared to standard statistical analysis, but also its limitations.

INTRODUCTION

Causal inference is the study of understanding cause-and-effect from observational data. Unlike correlation, it unveils underlying mechanisms, guiding informed decision-making. The Python library `DoWhy` attempts to streamline causal analysis by automating model specification, identification, estimation, and refutation [7][1]. It empowers users to uncover causal relationships, offering deeper insights across domains. This project is an exploration of the `DoWhy` library. However, before delving into the methods implemented by `DoWhy`, it's essential to grasp the theoretical underpinnings that form its backbone. At the heart of `DoWhy` lies the work of Judea Pearl and his development of *Do Calculus* [4]. This framework provides a rigorous mathematical foundation for causal reasoning, enabling inference of causal effects from observational and interventional data. By understanding the principles of *Do Calculus*, users of `DoWhy` gain insight into the methodologies employed by the library and the principles guiding their application in real-world scenarios.

THEORY

In J. Pearl's framework for causal inference, a distinct division exists between standard statistical analysis and causal analysis [4]. While standard statistical analysis primarily investigates associations between variables under static conditions, causal analysis extends beyond raw data to infer probabilities under dynamic circumstances, such as treatments or external interventions. Within associational statistics, the population U is comprised of units denoted as u_i. Variables are real-valued functions defined on the units of the population as $Y(U = u_i)$ and $A(U = u_i)$, where Y is a measured outcome and A is an attribute inherent to the unit. From these functions, joint probability distributions $Pr(Y = y, A = a)$ and conditional distributions $Pr(Y = y \mid A = a)$ can be defined. Associational inference involves leveraging these distributions to make estimates, tests, posterior probabilities, etc., enabling the inference of relationships between outcome variables Y and attributes A across the population U [2].

The crucial disparity emerges in the realm of causal inference, where joint distributions alone cannot adequately describe the statistics, necessitating additional untestable causal assumptions visualised in causal graphs [4]. A causal graph, also known as a causal diagram or causal network, is a graphical representation of causal relationships among variables in a system. In causal inference, the observable variable $Y_v(u)$ is introduced, representing the measured outcome value Y for the individual u having undergone treatment $V = v$. This notation also allows for the description of the counterfactual observable, i.e. the potential outcome had unit u received a different treatment v_1. Assuming random selection of individuals, the probability of observing outcome $Y = y$ given treatment v_0 is expressed as:

$$P(Y = y \mid \mathrm{do}(V = v_0)) = P(Y_v = y) \qquad (1.1)$$

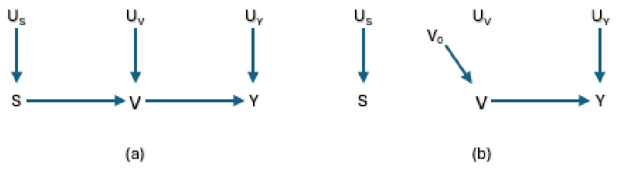

Figure 1: A causal diagram, showing the effect of enforcing the do operator, thereby creating a new causal diagram. U represents error terms, S is some cause, V is the treatment and Y is the outcome.

Here, the so-called 'do-operator' is introduced [4]. The do-operator symbolizes an intervention within the framework of causal modelling. When applied to a causal model, such as the one depicted in Fig. 1a, the expression $\mathrm{do}(V = v_0)$ corresponds to the creation of a new causal model shown in Fig. 1b. In this case, imposing V to be v_0 cuts of other effects. If S serves as a covariate of V, V represents the treatment, and Y denotes the response variable, then the distribution $P(s, y \mid \mathrm{do}(v_0))$ signifies the probability of attaining a response $Y = y$ and a covariate level $S = s$ under the null hypothesis, that the treatment $V = v_0$ is uniformly distributed over the population. From this, the post-treatment distribution is defined as:

$$P_M(y \mid \mathrm{do}(v)) \triangleq P_{M_v}(y) \tag{1.2}$$

Eq. 1.2 allows us to assess the effects of interventions on the outcome. The effect of interventions is quantified as the difference between the expected outcomes under different intervention scenarios represented by:

$$\mathbb{E}(Y \mid \mathrm{do}(v_0)) - \mathbb{E}(Y \mid \mathrm{do}(v_1)) \tag{1.3}$$

However, a significant challenge in causal inference is the concept of identification. Identification concerns whether it is feasible to determine the post-treatment distribution solely based on the pre-treatment distribution. That is, whether causal effects can be reliably estimated from observational data and what assumptions are necessary to make the causal claim hold. This challenge emphasizes the importance of robust methodologies and careful consideration of underlying assumptions in causal inference research. Identification is always possible for a Directed Acyclic Graph (DAG) [4]. A DAG is a graphical structure consisting of nodes connected by directed edges (arrows) such that there are no directed cycles. For a DAG the post-treatment distribution is given by :

$$P(s_1, s_2, \ldots, s_k \mid \mathrm{do}(v_0)) = \prod_{i \mid s_i \notin V} P(s_i \mid \mathrm{pa}_i)\Big|_{v=v_0} \tag{1.4}$$

The importance of eq. 1.4 can hardly be overstated. This equation represents the transition from a counterfactual scenario to a conditional probability. In the example illustrated in Fig. 1, the post-treatment variable is $P(w, y \mid \mathrm{do}(v_0)) = P(w \mid v_0)P(y \mid v_0) = P(w)P(y \mid v_0)$. The second equality can be inferred from the graph in Fig. 1, since information concerning v_0 does not alter the probability distribution of $P(w)$.

Measuring effects

A confounding variable is defined as a variable, that influences both the treatment and response variables, hereby complicating the estimation of the true causal effect of the treatment on the response variable. In certain scenarios, such as in a laboratory setting, it may be impractical or impossible to measure all confounding variables. Although confounding variables complicate measuring causal effects, various criteria exist to identify which variables are necessary for estimating the effect of V on Y.

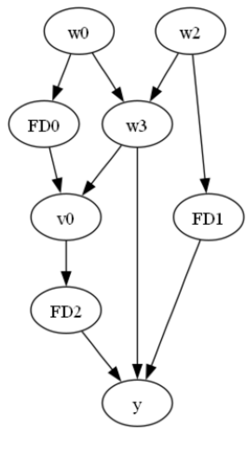

Figure 2: A causal diagram showing the causal chain of v_0 acting on y dependent on various factors.

The *Backdoor Criterion* states that the set S is a sufficient set of variables to measure the effect of V on Y, if no element of S is a descendent of V, and the elements of S block all backdoor paths from $V \to Y$ [4]. A backdoor path is any path from the treatment to the outcome that contains an arrow pointing into the treatment variable. In Fig. 2, $v0$ is a treatment, y is the response variable and the sets $S = \{FD_1, w_3\}$, $S = \{FD_0, w3\}$, $S = \{w_0, w_1, w_2\}$ are sufficient to estimate the causal effect of $v0 \to y$. Given a set of variables S which meets the backdoor criteria, it is possible to calculate the post-treatment distribution as [3]

$$P(y|do(v)) = \sum_s P(y|v, s)P(s) \tag{1.5}$$

The *Frontdoor Criterion* is satisfied for a set of variables S relative to an ordered pair of variables V and Y, if S intercepts all directed paths from $V \to Y$, there is no backdoor path from $V \to Y$, and all backdoor paths $S \to Y$ are blocked by V. If a set of variables satisfies the *Frontdoor Criterion* then it is possible to calculate the post-treatment distribution as [4]:

$$P(y \mid \mathrm{do}(v)) = \sum_v P(s|v) \sum_{v'} P(y|(v', s)P(v') \tag{1.6}$$

Applying this to figure 2, it is possible to calculate the causal effect solely by knowing $FD2$, v_0 and y. In the `DoWhy` library, the elements of set S are called instrumental mediating variables. When a causal model has been proposed, it is crucial to evaluate the stability and reliability of the asserted causal relationship across diverse conditions and assumptions. Robustness testing entails subjecting the causal model to rigorous scrutiny against a spectrum of potential confounding factors, including both known and unknown variables. It is imperative to ensure that the inferred causal relationship remains consistent and robust despite the introduction of confounders. A successful robustness test should yield results that align closely with the initial claimed causal relationship, reaffirming its validity under varied circumstances and mitigating concerns regarding spurious associations or biases [5].

METHODS IN DOWHY

The DoWhy library in Python is grounded in the mathematical framework of *Do Calculus*. The main results of *Do Calculus* are described in the theory, but the DoWhy library extends further than described there. It operates within the realm of DAGs. Understanding how the library works involves breaking down its functionality into four distinct stages:

1. Setting up the causal network, i.e. specifying the DAG. The library leverages DAG's to visualize and represent causal relationships between variables. DAG's provide a graphical depiction of causal structures, illustrating the direct and indirect pathways between variables [4].
2. Identify the relevant estimand method, i.e. choose between the backdoor criteria, frontdoor criteria, instrumental variables or mediation of variables.
3. Estimate the causal effect. The functional relationship between variables should be specified. Often, a linear relationship is chosen for simplicity, however, DoWhy offers a wide range of estimators such as propensity score matching, non-parametric natural direct- or indirect effects.
4. Testing the robustness of the model and the effect estimates. DoWhy incorporates various methods to help validating the results, like negative control tests and sensitivity analysis.

Built into the library is the additive noise model. Which accounts for the fact that the relationship between cause and effect isn't deterministic but is influenced by random fluctuations. In J. Pearl's *Do Calculus* this is incorporated in eq. 1.1, while in the DoWhy library it is explicitly expressed as:

$$Y_i = f_i(X_i) + N_i \tag{1.7}$$

Here, Y_i is the outcome, X_i is the causal variable, $f(X_i)$ is the deterministic part of their relationship, and N_i is random noise. As a result of this, even when intervening on X, the outcome Y may still vary due to random noise. Accounting for this variability is vital for accurately estimating the causal effect of X on Y, which is implicitly done in DoWhy through residual error terms in regression models or other estimation methods.

RESULTS AND DISCUSSION

Generating a causal model

To show the practical applications of the DoWhy library, a mock experiment is set up. The experiment considers total beer sales at the physics Friday bar 'Caféen?' and the potential effect of offering a discount on these sales. Artificial data was generated following the causal graph shown in figure 3. Here, we presume that an offer being present positively influences the total sales. The value of the offer is binary and is in itself influenced by two other parameters, the day of the week and the occasion of a 'special event' (Oktoberfest for example). Here, the day of the week takes discrete values, while the event parameter is continuous, sampled from a uniform distribution. Both of these parameters also directly influence the total sales, making them confounders.

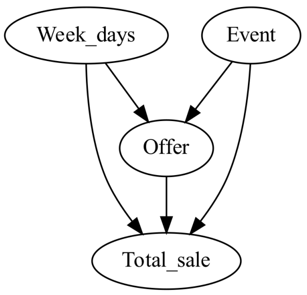

Figure 3: A simple causal graph for estimating the effects of offering a discount on beers on the total beer sales at 'Caféen?'.

For this experiment, ten thousand data points are generated with a linear dependency between the total sales and the other three parameters, each with a different slope (or strength). It is most likely to have an offer on Thursday or Friday, except for when there are events. For the full details on the data generation, the reader is referred to the supplementary code. The only value of real importance is the strength of the effect of the offer on the total sales since the goal is to use DoWhy to estimate the strength of this effect (assuming this is unknown). This strength is set to be 150. The normalized distributions of the total sales for days with and without an offer are shown in Fig. 4.

Before carrying out any causal analysis on the data, it is interesting to estimate the effect of the offer on the total sales by using a naive linear regression. Doing this results in an estimated effect strength of 240. Apparently, ignoring the day of the week and event parameters results in an overestimation of 60% of the offer effect compared to the true value. This bias can be addressed easily using a multivariate analysis in DoWhy. Using the dataset and the DAG of Fig. 3 as inputs for the causal model (but without knowledge on the

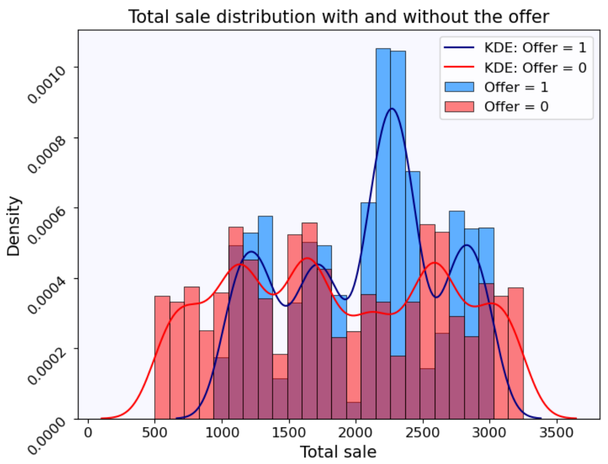

Figure 4: Distribution of the total sales for days with (1) and without (0) an offer, with on top a probability density estimated with a Gaussian kernel density estimator.

underlying parameters used for the data generation), a *backdoor linear regression* method in `DoWhy` (applying eq. 1.5) gives an effect estimation of almost exactly 150 (to 11 decimal points precision). So, by controlling for all of the confounding variables, spurious correlations can be removed and the real causal effect can be determined.

Still, the causal effect estimation is only as good as the assumptions about the causal graph used as input for the model. The truth of these assumptions is untestable, however, `DoWhy` does offer various refutation tests to test robustness. One of these refutation tests was applied to the causal model, where another confounding variable is added to the causal diagram, but no data is known on this so called '*unobserved common cause*'.

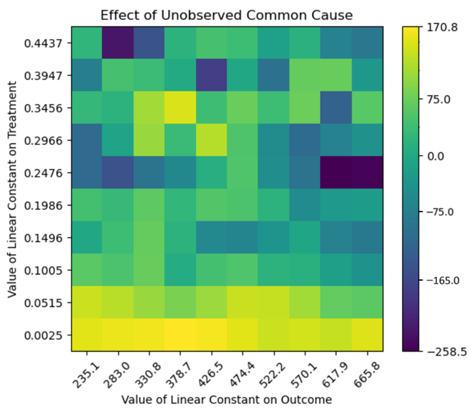

Figure 5: Heat map of the estimated effect of the offer on total sales as a function of influence of some unobserved confounder.

The graph in Fig. 5 shows the influence that such a confounder could have on the estimated effect of the offer on the total sales. From this, it can be concluded that if the hidden confounder has little direct effect on the offer, the model will still be able to predict the true effect rather accurately. However, with increasing direct effect on both the offer and total sales, the presence of this hidden confounder will result in incorrect estimated effects.

Validating causal graphs

Causal graphs are dynamic and can change over time. Suppose that the staff of 'Caféen?' just made the discovery that most of their beer stock will expire soon and they need to make sure they sell them fast. They might consider having offers more often because of this, which would result in a new causal diagram shown in figure 6. Mock data for the new instrumental variable 'stock' was generated and added to the causal model.

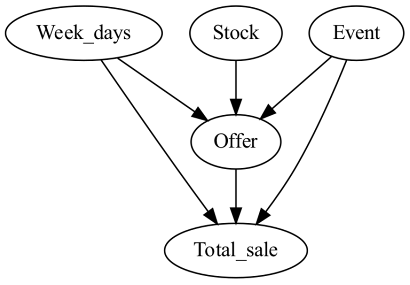

Figure 6: A similar causal diagram as in Fig. 3, this time with the added instrumental variable 'stock'.

Next to the refutation tests for the estimated causal effect, `DoWhy` also offers refutation tests to test the validity of a full DAG. To check if the new causal model fits the data well, the different nodes are tested for meeting the Causal Markov condition (CMC). This condition states that each variable in a causal graph is independent of its non-effects (variables not directly or indirectly affected by it) given its parents (variables that directly influence it). This condition can be checked with `DoWhy`'s `falsify.validate_lmc` feature from the `gcm` library [6]. The results can be seen in table I. From this table, it can be seen that the CMC is satisfied in nearly all instances, but there is some unexpected correlation between total sales and stock.

Edge	**P-value**	**Significant**
Week_days → Event	0.2787	False
Week_days → Stock	0.2773	False
Stock → Event	0.6566	False
Stock → Week_days	0.2773	False
Event → Week_days	0.2787	False
Event → Stock	0.6566	False
Total_sale → Stock	0.0033	True

Table I: Statistical significance of causal relationships.

Another method of verifying the DAG is by using `gcm`'s `falsify_graph` function. This verifies a DAG by comparing it to permutated DAGs and conducting tests related to the Local Markov Condition (LMC). Applying these methods to the causal model of figure 6 results in the following insights:

1. The DAG is informative, which means that the observed data is suitable for making causal claims.
2. Out of 20 permutations, 0 permutations lie in the Markov equivalence class of the given DAG. This means that none of the permutated DAGs are equivalent to the given DAG in terms of the conditional independence relationships assumed by the DAG.
3. There are 2 out of 7 possible violations of the LMC. This means that there are conditional independence relationships implied by the given DAG that do not hold true in the observed data.
4. The given DAG is better than 95.0% of the permuted DAGs. This suggests that the observed data fits the structure of the given DAG relatively well compared to randomly permuted DAGs.
5. The DAG is not rejected. This means that despite the violations of some LMCs, the observed data provides valuable information for causal inference, and the given DAG is considered acceptable for further analysis.

These results were expected for this specific causal model, since the data was artificially created following the DAG. However, for real data, `DoWhy`'s DAG validation is extremely powerful, especially for complex data structures for which causal relations are hard to predict.

Complex models

In practice, causal models can quickly become very complex. Therefore, as the last part of the mock experiment, a causal model following the notably more complex causal graph of Fig. 7 is implemented. To increase complexity, data is generated using a combination of both linear and non-linear relations. As a result, it becomes very hard to predict the expected causal effect of the offer on the total sales. Still, `DoWhy`'s methods can be used to get a reasonable approximation for the causal effect.

To show this, both a propensity score matching and a backdoor linear regression method are applied to the model to determine the estimated causal effect (ECE). The results can be seen in table II. Even though the data does not follow a linear relation, the linear regression methods still give a similar result as the propensity score matching. The robustness of the methods is tested by changing the treatment value in the dataset to be random while leaving the rest of the data untouched. As a result, one would expect the causal effect to be random for each data point, resulting in an ECE close to zero. Table II shows that this behaviour is indeed observed. From this, it can be concluded that, somewhat obviously, the causal graph matches the generated data.

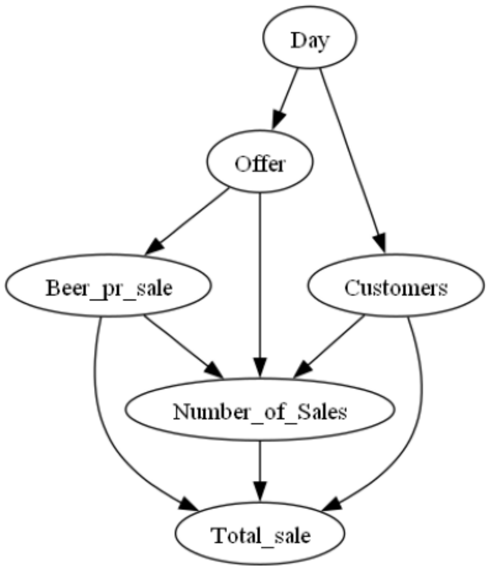

Figure 7: A more complex causal diagram describing beer sales at the physics Friday bar 'Caféen?'.

Method	**ECE**	**ECE with random treatment**
Propensity score matching	156.37	1.31
Backdoor linear regression	141.64	-3.12

Table II: Estimated causal effect (ECE) for the complex causal model.

CONCLUSION

The `DoWhy` library offers a comprehensive toolkit for conducting causal inference analysis. It combines the potential outcome framework, mathematical formulations and key assumptions from *Do Calculus* with graphical causal models. Applying the library to mock data showed its potential for accurate estimation of causal effects, outperforming naive results from standard statistical analysis, which one might be tempted to use otherwise. Of course, the causal effect estimations can only be as good as the causal model that was put in, which forms a major challenge for using the `DoWhy` library on real data. Even though the correctness of a causal graph can never be proven, `DoWhy` offers various refutation tests to investigate the validity and robustness of both the effect estimations and the causal graphs. Everything combined, this makes it a powerful library for exploring causal inference statistics.

REFERENCES

[1] Patrick Blöbaum et al. “DoWhy-GCM: An extension of DoWhy for causal inference in graphical causal models”. In: *arXiv preprint arXiv:2206.06821* (2022).

[2] Paul W. Holland. “Statistics and Causal Inference”. In: (Dec. 1986). DOI: `10.1080/01621459.1986.10478354`. URL: `http://www.jstor.org/stable/2289064`.

[3] Judea Pearl. “Causal diagrams for empirical research”. In: *Biometrika* 82.4 (1995), pp. 669–688.

[4] Judea Pearl. “Causal inference in statistics: An overview”. In: (Oct. 2009). DOI: `10.1214/09-SS057`. URL: `https://doi.org/10.1214/09-SS057`.

[5] Judea Pearl. “Robustness of Causal Claims”. In: (2004). URL: `https://arxiv.org/ftp/arxiv/papers/1207/1207.4173.pdf`.

[6] PyWhy Developers. *Refuting Causal Structure.* `https://www.pywhy.org/dowhy/v0.11.1/user_guide/modeling_causal_relations/refuting_causal_graph/refute_causal_structure.html`. Accessed: 2024-03-22.

[7] Amit Sharma and Emre Kiciman. “DoWhy: An End-to-End Library for Causal Inference”. In: *arXiv preprint arXiv:2011.04216* (2020).