Identification and the causal hierarchy

This tutorial covers

- **Motivating examples for identification**
- **Using y0 and DoWhy for identification and deriving estimands**
- **How to derive counterfactual graphs in y0**
- **Deriving SWIGs for graph-based counterfactual identification**

Causal identification is the task of determining when we can make a causal inference from purely observational data, or a counterfactual inference from observational or experimental data.

Causal identification is especially important in the era of AI. There is a common belief in AI that any AI task, including causal reasoning tasks, can be solved simply by scaling up to more data. Identification tells us it is not that simple, that some causal problems can't be solved even with infinite data unless we explicitly make certain causal assumptions. Further, suppose scaling up a deep neural network architecture with more parameters and data does seem to solve a causal reasoning task. In this case, the ideas behind causal identification tell us what kind of causal inductive bias in the model architecture *must* be encoding. That can help us understand when the reasoning task will fail, as well as build better architectures that can accomplish more with less data.

Identification historically a theory-heavy part of causal inference. Fortunately, we can rely on libraries to do the theoretical heavy-lifting for us, and focus on writing good code. In this tutorial, we'll focus on two libraries, Y0 (pronounced why-not), which implements algorithms for identification using graphs and the do-calculus. We'll also look at DoWhy, a popular tool for identifying and estimating causal effects.

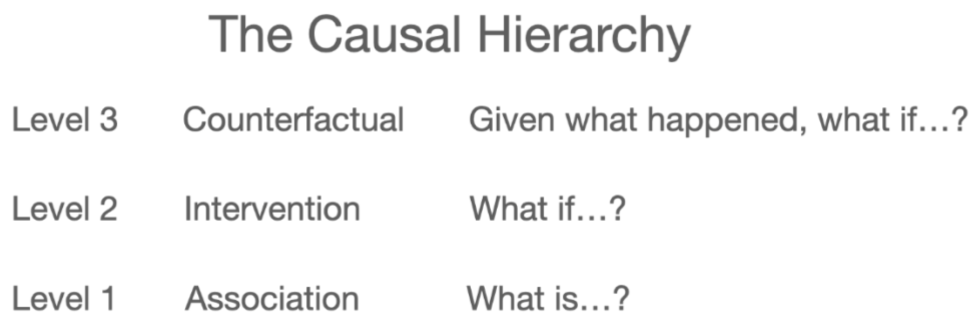
1 *The causal hierarchy*

The causal hierarchy, also known as Pearl’s hierarchy or the ladder of causation, is a three-level hierarchy over the types of causal questions we ask, the types of models we build, and the types of causal inferences we make.

The causal hierarchy consists of three levels:

1. Association
2. Intervention
3. Counterfactual

When we do a statistical or causal analysis, we are reasoning at one of these three levels.



The Causal Hierarchy

Level 3	Counterfactual	Given what happened, what if…?
Level 2	Intervention	What if…?
Level 1	Association	What is…?

Figure 1 The causal hierarchy, from level 1 to level 3.

This association level is concerned with "what is" questions. Consider for example the following case study:

ONLINE GAMING CASE STUDY

Suppose you are a data scientist at an online role-playing game company. Your leadership wants to know if side-quest engagement (mini-objectives that are tangential to the game's primary objectives) is a driver of in-game purchases of virtual artifacts. If the answer is yes, the company will intervene in the game dynamics such that players engage in more side-quests.

In our online game, many players are members of guilds. Guilds are groups of players who pool resources and coordinate their gameplay, such as working together on side-quests. Our model assumes that the amount of in-game purchases a player makes also depends on whether they are in a guild; members of the same guild pool resources and many resources are virtual items they must purchase.

In this case study, an example association-level question is "What are typical in-game purchase amounts for players highly engaged in side-quests?" Reasoning at this level aim to describe, model, or detect dependence between variables. This level does not reason about any causal relationship between the variables.

Causal reasoning starts at the intervention level. Reasoning at this level is concerned with interventional questions such as "what if we set side-quest engagement to high?", including conditional hypotheticals such as "what if side-quest engagement were high?"

Counterfactual reasoning lives at the counterfactual level. This level is concerned with counterfactual questions such as "given this player had low side-quest engagement and low purchases, what would their level of purchases have been if they were more engaged?" This level includes questions that build on counterfactual reasoning, such as attribution and explanation (e.g., "why did this player have low purchases?), fall at this level as well.

1.1 Where models and assumptions fall on the hierarchy

More generally, a "model" is a set of assumptions about the data-generating process. Those assumptions live at various levels of the hierarchy. Models at the associational level have statistical but non-causal assumptions. For example, suppose I'm interested in P(I|E=e), for either value ("low", "high") that e might take.

I might use the following model:

$i \sim \text{Normal}(u_e, \sigma_e)$

Here, I assume that in-game purchases is a mixture of normal distributions, one with mean and scale $u_{E=\text{"low"}}$, $\sigma_{E=\text{"low"}}$ when side-quest engagement is low and $u_{E=\text{"high"}}$, $\sigma_{E=\text{"high"}}$ when engagement is high.

Alternatively, I could use simple linear modeling assumptions.

$n_i \sim N(0, \sigma)$

$i = \beta_0 + \beta * \text{Ind}(E = \text{"high"}) + n_i$

Here, Ind() is an indicator function that returns 1 when engagement is high and 0 otherwise. These two model differ in non-causal assumptions; they both use a normal distribution to model in-game purchases, but with different parameterizations. Alternatively, I could use a nonlinear function f in $i = \beta_0 + \beta * f(E) + n_i$, including adding another layer to make a basic neural network like $i = \gamma_0 + \gamma * f(\beta_0 + \beta * f(E))$.

These assumptions (normal distribution vs another canonical distribution, linearity vs nonlinearity, constant/non-constant variance) are statistical assumptions placed on P(I|E). They are not causal assumptions.

Once we add causal assumptions, we move to a higher level of the hierarchy.

Level 2 assumptions

An example of a level 2 model would be a causal graphical model (A.K.A. a causal Bayesian network) – a probabilistic model trained on a causal DAG. The causal DAG by itself is a level 2 set of assumptions; assumptions about what causes what. Generally, assumptions that let you deduce the consequences of an intervention are level 2 assumptions.

Level 3 Assumptions

The canonical example of a level 3 model is a structural causal model. But more generally, assumptions about mechanism, i.e., how variables affect one another, is a level 3 assumption. For example, suppose your DAG has the edge X→Y. The DAG already states that X is a causal of Y, a level 2 assumption. Any additional information you can provide about that causal relationship beyond what's represented in the causal DAG, beyond "X is a cause of Y" is a mechanistic assumption.

In more formal terms, there is a one-to-many relationship between a DAG and the SCM. In general, there is a set of SCMs that are consistent with a given causal DAG and a given joint probability distribution. Any additional assumptions beyond what's in the DAG that you make that reduces that set of SCMs is a level 3 assumption. For example, assuming that X→Y is a linear relationship is a level 3 assumption because it eliminates all the possible SCMs where X→Y is nonlinear. This is a level 3 assumption even if you say nothing about the other edges in the DAG.

1.2 Where data falls on the hierarchy

Recall the differences between observational data and interventional data. Observational data is passively observed; as a result, it captures statistical association resulting from dependence between variables in the data generating process.

In our online gaming example, these were logged examples of side-quest engagement and in-game purchases pulled by a database query. Observational data lives at the association-level of the hierarchy.

Interventional data is generated as the result of applying an intervention, such as data collected from a randomized experiment. In the gaming example, this was the data created because of an A/B test that randomly assigned players to different groups of fixed side-quest engagement levels. Intervention data lives at the intervention-level of the hierarchy.

Counterfactual data, which lives at the counterfactual level of the hierarchy, is the odd case. Recall the "fundamental problem of causal inference", which says we can only observe one potential outcome for a unit in our data. For example, if we observe an ill subject receiving treatment and getting better, we can't observe whether they would have gotten better if given placebo. Except in rare cases, counterfactual data isn't available.

Cases where counterfactual exists

Many take the absence of counterfactual data as a reality, and in most settings, it is. However, there are cases where counterfactual data, data with multiple potential outcomes for a single unit or subject, exist. These are typically cases where the modeler can control a deterministic data generating process. For example, suppose a cloud computing company has a complex but deterministic policy for allocating resources based on customer demand and various constraints. Using logs, they can generate counterfactual allocation outcomes by simulating changes to the policy, demand, or other constraints.

Other examples include data produced by simulators and process models, such as those used in climate science, materials science, engineering, and computational biology. One can create synthetic counterfactual data by rerunning simulation with a targeted change to initial conditions. Counterfactual simulation is a common use-case for these models, e.g., in the case of climate science, simulating the effects of different greenhouse gas emissions scenarios.

Simulated environments are also commonly used to train machine learning algorithms, such as training robots and self-driving cars in virtual environments that emulate real-world physics. Those agents can be trained on counterfactual data simulated in these environments.

1.3 Where distributions fall on the hierarchy

Let's use the term "query" to refer to a probability distribution such as $P(I_{E=\text{"low"}}<50)$, or a derivative of a probability distribution such a probability value like $P(I_{E=\text{"low"}}<50)$, or an expectation like $E(I_{E=\text{"high"}})$. Expectations of differences like causal effects, such as $E(I_{E=\text{"high"}} - I_{E=\text{"low"}})$ are also queries. Ordinary probability queries (those without causal terms such as P(I|E="low")) fall at the association level of the hierarchy. Intervention queries such $P(I_{E=\text{"low"}})$ and causal effects like $E(I_{E=\text{"high"}} - I_{E=\text{"low"}})$ fall at the intervention level of the hierarchy.

Counterfactual queries such as $P(I_{E=\text{"low"}}| I>50)$ and $E(I_{E=\text{"high"}}-I_{E=\text{"low"}}|E=\text{"high"})$ fall at the counterfactual level of the hierarchy. When coding the ideas in this chapter, we want to think of queries as first-class citizens. We can do this with the python library Y0 (pronounced "why-not").

Listing 1 Creating a query in Y0

```
!pip install git+https://github.com/y0-causal-inference/y0.git@v0.2.0

from y0.dsl import P, Variable      #A
G = Variable("G")      #B
E = Variable("E")      #B
I = Variable("I")      #B
query = P[E](I)      #C
query      #D
```

#A "P" is for probability distributions, and "Variable" is for defining variables.
#B Define variables G (guild membership), E (side-quest engagement), and I (in-game purchases).
#C Define the distributional query P(I_E).
#D If running in a notebook environment, this will show a rendered image of P(IE)

The `query` object is an object of the class `Probability`. When we evaluate the last line of the code, the class's `__repr__` method (which tells Python what to return in the terminal when you call it directly) is implemented such that when we evaluate the object in the last line of the above code in a Jupyter notebook, it will display rendered LaTeX (a typesetting/markup language with a focus on math notation).

$$P(I_E)$$

Figure 9. 2 The LaTeX image returned when you evaluate the query object in Listing 9.1.

Questions fall into levels of the hierarchy via the mapping of questions to queries. For example, "What would be the level of in-game purchases if side-quest engagement were low?" ($P(I_{E=\text{"low"}})$) is an interventional question on the interventional level. For people who have high side-quest engagement, how much more (or less) in in-game purchases would they have if their engagement had been low ($E(I_{E=\text{"high"}}-I_{E=\text{"low"}}| E=\text{"high"})$, also known as effect of treatment on the treated, which we'll discuss in section 7).

We use the term "query" in the context of identification. In probabilistic inference, a "query" is the general term for the target of inference. In causal identification, we are attempting to use data from lower levels of the hierarchy to infer higher level queries.

1.4 The causal hierarchy theorem

The causal hierarchy is not merely a helpful illustration. It is backed up by *the causal hierarchy theorem*. The causal hierarchy theorem says that the three layers of the hierarchy are "almost always separate." Roughly speaking, "separate" means that data from a lower level of the hierarchy is insufficient to infer a distribution at a higher level of the hierarchy. “Almost always” roughly means "this statement is true except in cases so rare that we can dismiss them as practically unimportant."

This is a key concept to understand in identification. If you want to infer, for example, a level 2 interventional query from level 1 data, you need level 2 assumptions.

2 Introducing identification

Recall with the online gaming example in chapter 7.

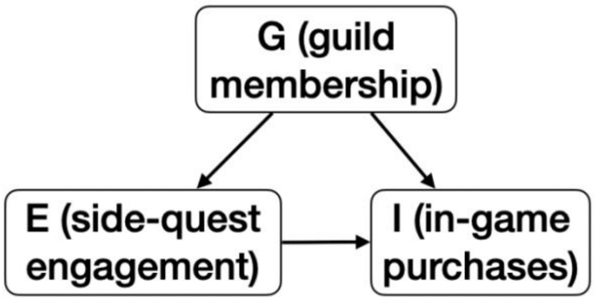


Figure 3 The DAG for the online gaming example.

Let's build the DAG with y0.

Listing 2 Building the online gaming DAG in y0

```
!wget
https://raw.githubusercontent.com/altdeep/causalML/master/book/id_utilities
.py -O id_utilities.py    #A
from id_utilities import *    #A
from y0.graph import NxMixedGraph as Y0Graph    #B
from y0.dsl import P, Variable
G = Variable("G")
E = Variable("E")
I = Variable("I")
e = E

dag = Y0Graph.from_edges(    #C
    directed=[    #C
        (G, E),    #C
        (G, I),    #C
        (E, I)    #C
    ]    #C
)    #C

gv_draw(dag)    #D
```

#A First we'll import some helper functions for identification and visualization that converts some y0 abstractions into abstractions we're familiar with.
#B Y0 works with custom graph class called NxMixedGraph. To avoid confusion, we'll call it a Y0Graph, and use it to implement DAGs.
#C Build the graph.
#D Draw the graph with a graphviz helper function.

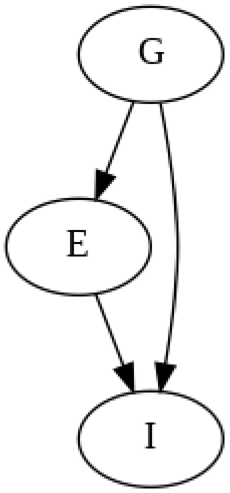


Figure 4 Drawing the online gaming graph with y0.

Our goal in causal effect estimation was to use our model of P(G, E, I) to simulate from P($I_{E="high"}$). Identification confirms that this was possible.

Listing 3 Checking identification of P($I_{E="high"}$) from P(G, E, I)

```
check_identifiable(    #A
    dag,    #A
    query=P(I @ e),    #B
    distribution=P(G, E, I)
)
```

#A Check identifiability given the DAG, a distribution, and a target query.
#B y0 represents interventions with @, we write P($I_{E=e}$) as P(I @ e).

This will return True. But suppose we didn't any observations of guild member ship G. As experts in probabilistic ML, we might train a latent variable model on P(E, I).

Listing 4 Checking identification of P($I_{E="high"}$) from P(E, I)

```
check_identifiable(
    dag,
    query=P(I @ e),
    distribution=P(E, I)
)
```

This will return False, P($I_{E=e}$) because is not identified from the DAG and P(E, I) alone. Even with *infinite* observations of E and I, we could not sample from P($I_{E=e}$) with only the assumptions in the DAG.

Given this introduction, let's define identification.

2.1 Defining identification

Suppose you had a trivial prediction problem. Your features were pairs of two variables {X, Y} and your label was Z, where Z = X + Y. It would be trivial to train a predictive model Z = f(X, Y), and as your data got larger, the better this model would perform.

Now suppose you reversed the model; now your features are Z, and your labels are {X, Y}! You need to learn the model {X, Y} = g(Z), which given a sum Z, predicts the two numbers added together to get Z. It can't be done! Even with a cutting edge deep neural network architecture and billions of examples of {x, y, z}. At least not without further assumptions to constrain the problem.

That's what identification in in statistics. You want to infer something. You have some estimator that will estimate that thing from data. Ideally, as the size of the data increases, your estimator should converge to the true value (its variance goes to zero). But the thing you hope to infer may not be identified given your assumptions, even as your data grows, your estimator will not converge to the thing you want to infer.

CAUSAL IDENTIFICATOIN

Causal identification is just statistical identifaction across levels of the hierarchy. You want to know if you can data from a distribution on level 1 or 2, and infer a query on a higher level.

2.2 Identification, estimands, and estimation

Identification tells you if you can infer a causal query from your data. Estimation is about how, in statistical terms, you go about doing it. For example, in this tutorial we'll look at a type of level 1 to level 2 identification called backdoor adjustment.

The causal *estimand* is an operation you apply to the distribution to infer the query. For example, in the next section, we'll look at the backdoor estimand, when tells us how to infer a level 2 query like a causal effect from a level 1 distribution. In our online gaming example, for the query $P(I_{E="high"}=i)$, the backdoor estimand is $\Sigma_g P(I=i|E="high", G=g)P(G=g)$. In Listing 9.3, I showed that $P(I_{E="high"})$ is identifiable from P(I, E, G). This estimand describes how, we use P(I, E, G) to derive $P(I=i|E="high", G=g)$ and $P(G=g)$, then multiply them and sum over G.

In this chapter I'll demonstrate how algorithms can use graphs to derive this and other estimands, as well as deriving a few by hand. But note in modern machine learning, you often don't need to know the estimand. If you have identification of your level 2 or 3 query given your data distribution and your causal assumptions, then it is enough to have a good inference of optimization procedure that targets your query, such as one that does gradient descent over a commonly used loss function. Your procedure is just a statistical estimator for your estimand.

2.3 Partial identification

Sometimes a level i query is not identifiable from your assumptions and level <i distribution. But it may be *partially identifiable*. Partial identifiability means you can put bounds on the thing you're trying to identify. Partial identifiability is an advanced topic that we will only mention here. But it is a useful approach if you can make those bounds tight, if you need to optimize a query (you can maximize the lower bound or minimize the upper bound), or you need your query to be greater than or less than a threshold (if you're lower bound is greater than, or your upper bound is less than, you win).

2.4 Graphical identification vs identification with level 3 assumptions

Causal identification can relies assumptions from level 2 or level 3 of the causal hierarchy. Identification with level 3 assumptions is often called *parametric identification*. Level 2 assumptions are assumptions we can capture with a DAG. In practice, this is called *nonparametric identification*, but we'll call it graphical identification. In this chapter we'll focus on graphical identification, because all you need is a DAG and a graph-based identification algorithm to get it to work. Level 3 identification relies more heavily the specific problem domain.

In the next section, we start with a common and practical example of identification, backdoor adjustment.

3 Level 2 Identification with Backdoor Adjustment

We passed our DAG and the intervention-level query $P(I_{E=\text{"high"}})$ to y0 and it told us it was identifiable from P(E, I, G). Now let's use y0 to derive the estimand – the operation we apply to P(E, I, G) to get $P(I_{E=\text{"high"}})$.

Listing 5 Deriving the estimand to get $P(I_{E=\text{"high"}})$ from P(E, I, G)

```
from y0.graph import NxMixedGraph as Y0Graph
from y0.dsl import P, Variable
from y0.algorithm.identify import Identification, identify

query=P(I @ e)
base_distribution = P(I, E, G)

identification_task = Identification.from_expression(
    graph=dag,
    query=query,
    estimand=base_distribution)

identify(identification_task)
```

This returns the expression:

$$\sum_G P(I|E,G) \sum \sum_{E,I} P(E,G,I)$$

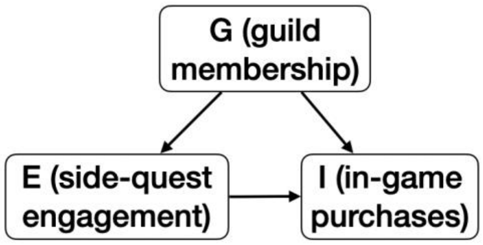
Figure 9. 5 Output of y0's identify function.

In our notation, this is $\Sigma_g P(I=i|E=\text{"high"}, G=g)\ \Sigma_{\varepsilon,\iota} P(E=\varepsilon, G=g, I=\iota)$, which simplifies to $\Sigma_g P(I=i|E=\text{"high"}, G=g)\ P(G=g)$. This is the backdoor estimand. We'll see at a high-level how y0 derives this estimand. But first let's look a bit more closely at this estimand.

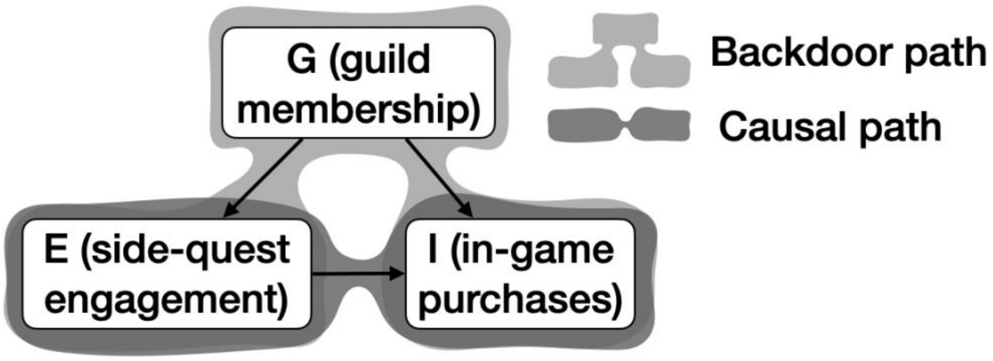
In general terms, suppose X is a cause of Y, and we are interested in the intervention-level query $P(Y_{X=x})$. Then the backdoor estimand, also known as the backdoor adjustment formula, is $\Sigma_z P(Y|X=x, Z=z)P(Z=z)$ (or $\int_z P(Y|X=x, Z=z)P(Z=z)dz$ when Z is continuous). Here, Z is a set of variables that are common causes of both X and Y. The intuition for what that set of variables should be comes from the rules of d-separation.

3.1 D-separation intuition

Consider the DAG for our online gaming example in the following Figure.



What is the difference between P(I|E="high") and P($I_{E="high"}$)? In the first case, observing E="high" gives us information about I by way of its direct causal impact on I, i.e., through path E->I. But observing E="high" also gives us information about G, and subsequently about I through the path E<-G->I. But in the second case, we only want the impact on I through the direct path E->I.



We call E<-G->I a backdoor path. Common causes like G on backdoor paths are called confounders. We are interested in the statistical "signal" flowing along the causal path from E to I, but that signal is "confounded" by the noncausal "noise" from additional statistical information through G on the backdoor path E<-G->I. We can d-separate that backdoor path between E and G by blocking on G.

Finally, to target the query P(I_E="high") by substituting I for Y, E for X and G for Z in the backdoor adjustment formula $P(I_{E=e}=i) = \Sigma_g P(I=i|E=e, G=g)P(G=g)$. In the backdoor adjustment formula, Z is a set of variables that will d-separate all the backdoor paths between X and Y; variables that "close the backdoors." The adjustment formula sums out/integrates over the backdoor statistical signal leaving only the signal derived from the direct causal relationship. But the adjustment formula only works if you include the right variables in the *adjustment set* Z.

3.2 What variables you can and can't adjust for

Again, Z is a set of variables that you select. To be a *valid adjustment set*, your set must satisfy the following criterion:

- D-separates all backdoor paths from X to Y.
- Does not introduce any new d-connecting paths between X and Y.
- Does not d-separate any causal paths from X to Y.

Let's consider a few examples. First, let's consider the example in Figure ???.

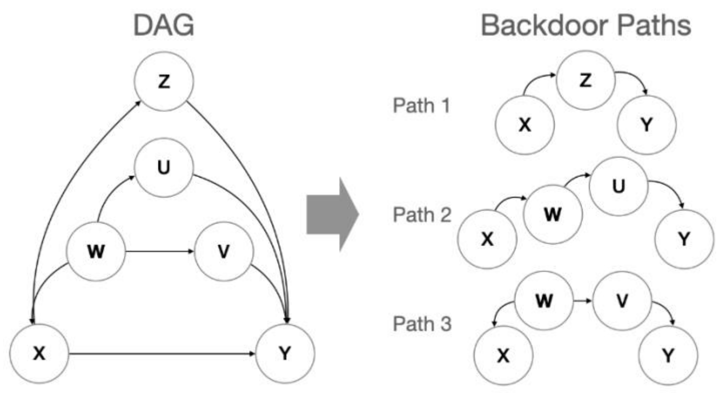


Figure 9. 6 This causal DAG has three backdoor paths between X and Y. Identification of P($Y_{X=x}$) requires an adjustment set that d-separates these three paths.

The DAG in Figure ??? has three backdoor paths between X and Y. Valid adjustors are any set of nodes that will d-separate all these paths. Let's first look at path 1, X ← Z → Y.

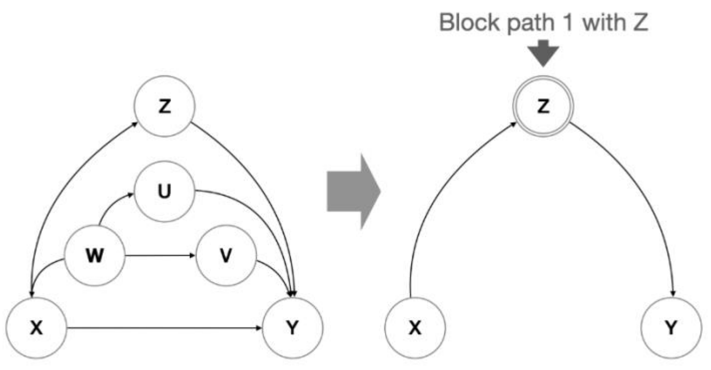


Figure 9. 7 W can d-separate path 1 with Z.

We can d-separate this path by blocking on Z.

Next we consider path 2, X ← W → U → Y. We can block this path with either W or U or both.

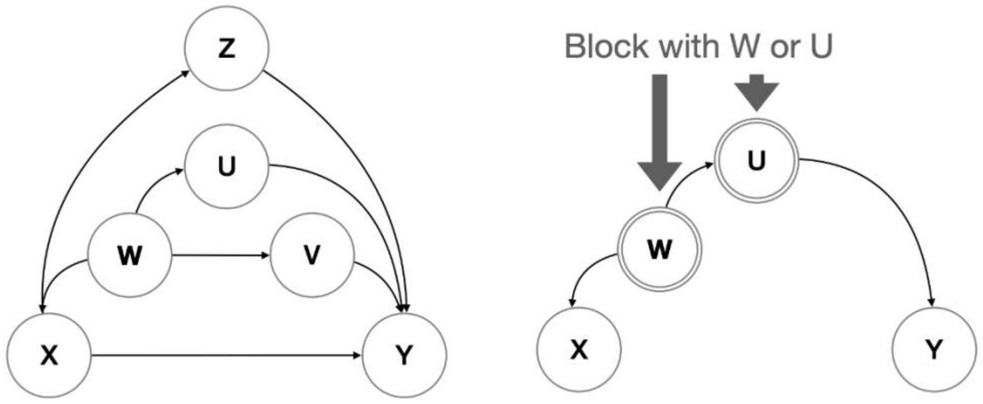


Figure 9. 8 Either W or U will block path 2.

Finally, consider the path X ← W → U → Y.

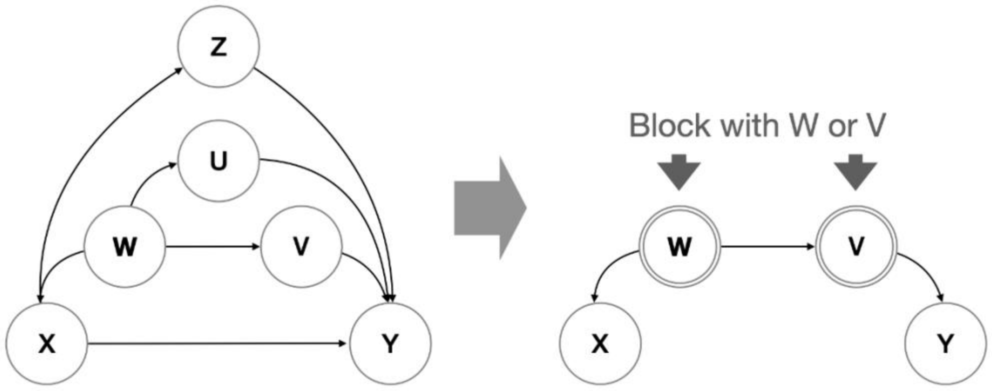


Figure 9. 9 Either W or V will d-separate path 3.

We d-separate this path with either W or V or both.

Having determined what nodes can d-separate all our backdoor paths, we can bring all those d-separating nodes into one set "**Z**". For instance, we can have **Z** = {Z, W, V, U}. But we could d-separate the backdoor paths with just {Z, W}. Enumerating all our valid adjustment sets, we have {Z, W, V, U}. {Z, W, V, U}, {Z, V, U}, {Z, W, U} and {Z, W}. Our next task is to select one of these valid adjustment sets to use with the backdoor adjustment formula.

Before we do that, let's consider one more example in Figure ???.

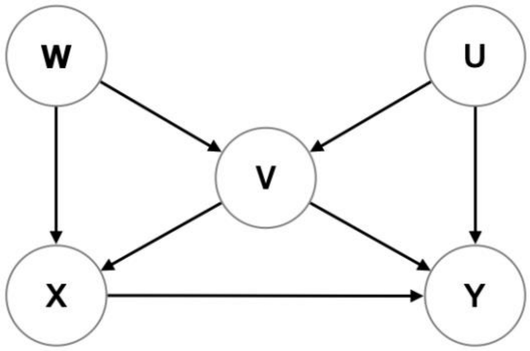


Figure 9. 10 This causal DAG has three backdoor paths from X to Y; X ← W → V → Y, X ← V ← U → Y, and X ← V → Y. But closing these paths by blocking on V d-connects X ← W → V ← U→ Y.

In Figure ???, we have three backdoor paths from X to Y; X ← W → V → Y, X ← V ← U → Y, and X ← V → Y. All of these paths contain V, and V is the only node that will d-separate X ← V → Y , so we need to include V in our adjustment set. But V is a collider between W and U, so blocking other paths with V will d-connect X ← W → V ← U → Y, so we need either W or U (or both) to d-separate that path as well, as illustrated in Figure ???.

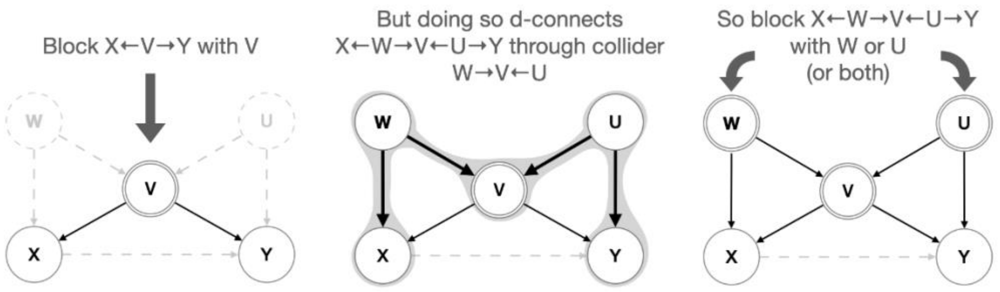

Block X←V→Y with V
But doing so d-connects X←W→V←U→Y through collider W→V←U
So block X←W→V←U→Y with W or U (or both)
W
U
V
X
Y

Figure 9. 11 Blocking backdoor path X ← V → Y with V opens a collider that d-connects X ← W→ V ←U → Y, which we need to block with W or U or both. So there are only three valid adjustment sets; {V, W}, {V, U}, or {V, W, U}.

So listing all our valid adjustment sets gives us {V,W}, {V, U} or {V, W, U}.

3.3 You can only adjust for what's in your data

Your query is identified given your assumptions (e.g., your DAG) and your data. For backdoor identification, your adjustment set is only valid if the members of that set are present in your data. If they are not present, you cannot apply the adjustment formula. See, for example, Figure ???.

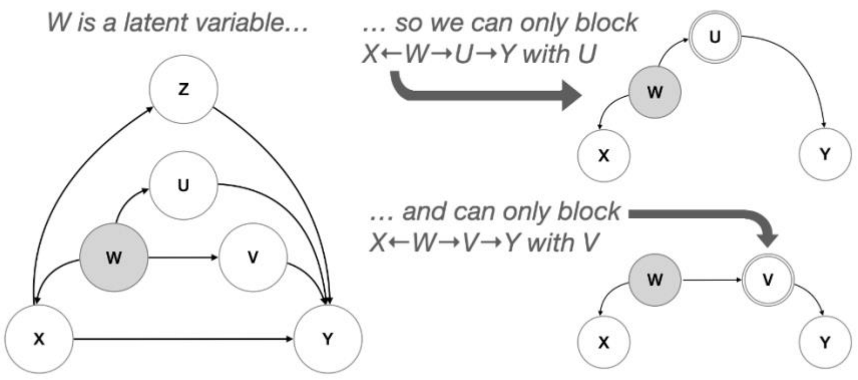

W is a latent variable…
… so we can only block X←W→U→Y with U
… and can only block X←W→V→Y with V
Z
U
W
V
X
Y

Figure 9. 12 When a variable is latent, we cannot use it to d-separate the backdoor paths. In this case, the only valid adjustment set is {Z, V}.

Recall that when W was not latent, our valid adjustment sets were {Z, W, V, U}, {Z, V, U}, {Z, W, U} and {Z, W}. Now the only valid set is {Z, V, U}, as it closes all the backdoor paths without relying on W.

Now consider Figure ???, where both W and U are latent.

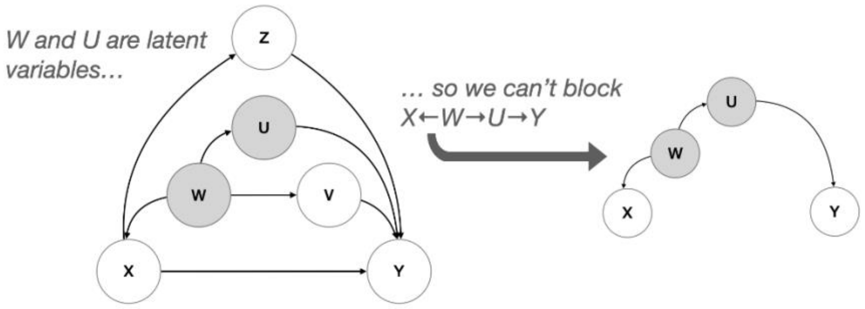


Figure 9. 13 When W and U are both latent, you can't close the backdoor X ← W → U → Y. When latent variables prevent you from closing all backdoor paths, you can't identify the query using a backdoor estimand.

In this case, we have no way of closing the backdoor through $X \leftarrow W \rightarrow U \rightarrow Y$. When there are backdoor paths we can't close due to the absence of variables in our data, the backdoor estimand does not exist. However, there may be other non-backdoor ways of identifying $P(Y_X)$. Now, we have to pick a valid adjustment set. But first let's look at common examples of *invalid* adjustment.

3.4 Common mistakes in backdoor adjustment

Figure ??? has adjustment sets {V,W}, {V, U} or {V, W, U}. You might have already guessed it would be silly to adjust for three things when two are enough; the less variables you have to sum or integrate over in the adjustment formula, the better. So why this focus on "valid" adjustment sets? Why not go directly for the "best" adjustment set? To answer that, let's look at an example of invalid adjustment, i.e., plugging the wrong variables into the adjustment formula.

Suppose you are an econometrician interested in the causal effect of education on income level. You assume the DAG in Figure ???.

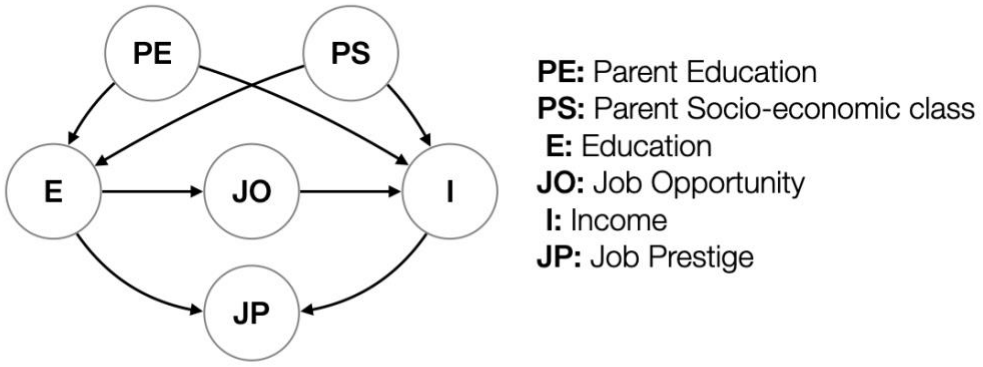


Figure 9. 14 To identify P(IE), we should not adjust for JO and JP.

The DAG has variables for education level (E), income level (I), parental education (PE), parental socioeconomic status (PS), job opportunities (JO), and job prestige (JP). To estimate the causal effect, we need to identify the intervention level query $P(I_E)$, i.e. what would be someone's income (I) if they had a certain education level (E). Your DAG assumes two common cause confounders: parental education (PE) and parental socioeconomic status (PS). Income level (I) is mediated by job opportunities (JO). Finally, you include a measure of job prestige (JP) as a child of education level and income level (I); you reason that in our culture, occupations that pay more and employ highly educated people (e.g., doctors, lawyers, investment bankers, data scientists) causes people to view them as prestigious.

Common practice in econometrics is to adjust for/control for all variables relevant to E and I. In this case, that would be PE, PS, JO, and JP. But we can see right away that would be a mistake, this would be an invalid adjustment set. It is appropriate to adjust for PE and PS as doing so would close the backdoor paths between E and I. But including JO in the adjustment set would block the direct causal path you are trying to quantify. And including collider JP in the adjustment set would d-connect a non-causal path E → JP ← I, adding non-causal "collider bias" to the estimate of the causal effect.

In other words, JP and JO are invalid adjusters; they are not members of any valid adjustment set. More generally, when you are interested in P(YX), mediators (nodes on the path from X to Y) and colliders between X and Y are invalid. Despite what the econometrics handbook says, you should *not* blindly try to control for as many variables as you can.

Now that we know what not to adjust for, we can look at how to choose between valid sets of adjusters.

3.5 Which adjustment set should I pick?

Once we have at least one valid set of adjusters, the task of identification is done and you are now doing statistical estimation. So you pick your adjustment set based on statistical and computational criteria.

Several practical considerations may have you favor one valid set over another. For example, suppose you are choosing between two valid sets {V,W} and {V, U}, you may prefer {V, W} over {V, U} for reasons such as:

- W and U are both discrete, but W has two possible values while U has eight, which gives your estimator more statistical degrees of freedom.
- Suppose U is continuous while W is discrete. If your adjustment set was {V, U}, you would have to either analytically or numerically integrate, which you'd like to avoid. So you favor {V, W}.
- Perhaps U and W are vectors, and W has a low dimension while U has a high dimension. So you go with {V, W} to avoid the curse of dimensionality.
- While you can observe both U and W, perhaps U is more expensive to observe, or its measurements are noisier.
- Perhaps W is compatible with an estimator your prefer because it has lower bias or variance, or because it scales well, while U is incompatible with that estimator.
- Maybe W is a common control variable in your domain, while adjusting for U would raise eyebrows. If either {V, W} or {V, U} will work, then you choose the path of least resistance.

This is to say that once you know what sets of variables are valid for backdoor adjustment, the selection of one of those sets generally boils down to non-causal considerations.

We now know how backdoor adjustment works, but we still don't know where it comes for, nor where other estimands come from. In the following sections, I'll introduce an intuitive derivation of the adjustment formula.

4 Demystifying the backdoor

Identification involves a good deal of math, which we use to prove that a query on a higher level of the hierarchy is equivalent under higher level assumptions to an estimand that operates on data from a lower level of the hierarchy. In this tutorial, rather than deep-dive into the math, we'll build the intuition for how it works. Let's start with building intuition for how we get to the backdoor estimand.

Let's consider our online gaming example again. The query is $P(I_{E=e})$ where "e" is "high" or "low". The backdoor formula says our $P(I_{E=e})$ is equal to the estimand $\Sigma_g P(I|E=e, G=g)P(G=g)$. So how do we arrive at this equality?

4.1 Ignorability

In counterfactual terms, let's consider two possible worlds, one with our original DAG, and one where we apply the intervention to side-quest engagement (E). Let's view the parallel world graph in Figure ???.

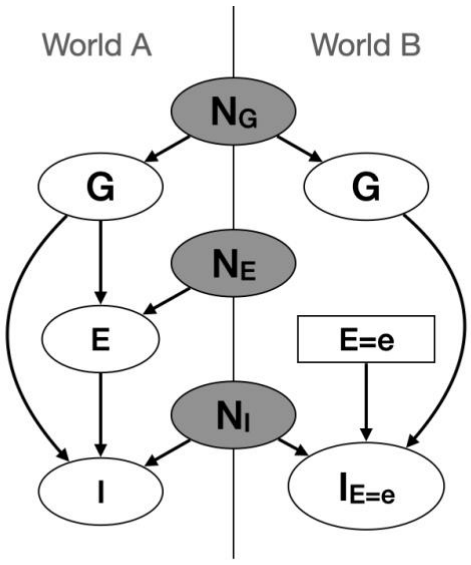


Figure 9. 15 We have two parallel worlds, world A where E is not intervened upon, and world B where E is intervened upon.

If you squint hard enough at this graph, you'll notice that it implies that E is conditionally independent from $I_{E=e}$ given G. We'll use some d-separation-based reasoning to see this. Remember that in general, we can't use d-separation to reason across worlds on a parallel world graph because the d-separation rules don't account for equivalent nodes (like G) in both worlds. But we'll use a trick where we reason about d-connect to G in world A, then extend that d-connection from G in world B.

First, consider that paths from E in world A to world B have to cross one of two bridges between worlds, N_G and N_I. But the two paths to N_I ($E \rightarrow I \leftarrow N_I$, $E \leftarrow G \rightarrow I \leftarrow N_I$) are both d-separated due to the collider on I.

So we have one d-connected path to world B ($E \leftarrow G \leftarrow N_G$). Now suppose we look at G in world B; from world B's G, it is one step to $I_{E=e}$. But we know that the value of G in both worlds must be the same; G is deterministically set by N_G, and neither G is affected by an intervention. So for convenience, we'll collapse the G's into one node in the parallel world graph.

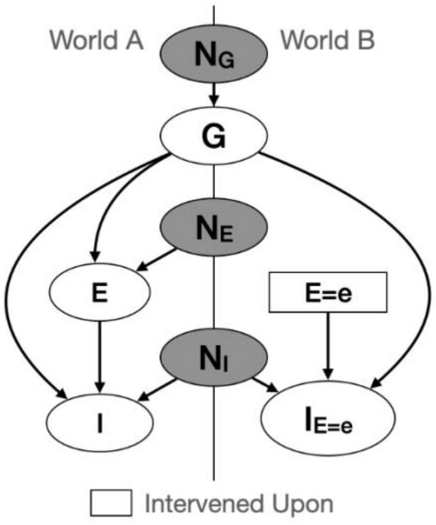


Figure 9. 16 Collapsing G across worlds reveals G d-separates E and $I_{E=e}$.

Looking now at the path $E \leftarrow G \rightarrow I_{E=e}$,, we can see this path is d-separated by G. Hence we can conclude $E \perp I_{E=e} | G$.

In causal inference jargon, this assumption is called *ignorability*. *Ignorability* means the causal variable E and the potential outcomes like $I_{E=e}$ are conditionally independent given confounders. We use this ignorability assumption in deriving the backdoor estimand.

4.2 Arriving at the backdoor

Now we'll use ignorability to show $P(I_{E=e})=P(I|E=e, G=g)P(G=g)$. Before we start, let's recall a key definitional fact of condition independence; if two variables U and V are conditionally independent given Z, then $P(U|Z=z, V=z) = P(U|Z=z)$. Flipping that around, $P(U|Z=z) = P(U|Z=z, V=v)$. $P(U|Z=z) = P(U|Z=z, V=\text{"apples"}) = P(U|Z=z, V=\text{"oranges"})$; it doesn't matter what value V takes because since Z rendered it independent from U, its value has no bearing on U. Introducing V and giving it whatever value we want is the trick that makes the derivation work.

Now let's start with $P(I_{E=e})$ and see how to turn it into $P(I|E=e, G=g)P(G=g)$.

1. First, for some value of in-game purchases I, $P(I_{E=e}=i) = \sum_g P(I_{E=e}=I, G_{E="high"}=g)$ by the law of total probability.
2. $\sum_g P(I_{E=e} = i, G_{E="high"}=g) = \sum_g P(I_{E=e}=i, G =g)$ as we know from our original DAG that G is not affected by the intervention on E.
3. Next we use the chain rule to factorize $P(I_{E=e}=i, G=g)$: $\sum_g P(I_{E=e}=i, G=g)= \sum_g P(I_{E=e}=i|G=g)P(G=g)$.
4. Now we come to the trick. $\sum_g P(I_{E=e}=i|G=g)P(G=g) = \sum_g P(I_{E=e}=i|E=e, G=g)P(G=g)$.
5. Finally, $\sum_g P(I_{E=e}=i|E=e, G=g)P(G=g) = \sum_g P(I=i|E=e, G=g)P(G=g)$ by the law of consistency.

Let's explain steps 4 and 5. Our ignorability result shows that $I_{E=e}$ and E are conditionally independent given G. So in step 4 we apply the independence trick that lets us introduce E. Further, I set the value of E to be e so it matches the subscript $_{E=e}$. This allows me to apply the law of consistency and drop the subscript $_{E=e}$.

Voila. Identification is just coming up with bits of math like this. Much if not most of traditional causal inference research boils down to doing this kind of math, or writing algorithms that do it for you.

Next, we'll look at the do-calculus, which provides simple graph-based rules for identification that we can use in identification algorithms.

5 The do-calculus and graphical identification

The do-calculus is a set of three rules used for identification. The rules use mutilated graphs and d-separation to determine cases when you can replace causal terms like $I_{E=e}$ with non-causal terms like $Y|X=x$. Starting with a query on a higher level of the causal hierarchy, one applies these rules in sequence to derive a lower level estimand.

In rough terms, the rules all say some version of the following:

> When certain nodes in the causal DAG become d-separated after removing certain edges, then certain probability distributions with certain terms are equivalent other probability distributions without those terms.

Since the rules are graph-based and general, we can apply use them in graphical algorithms for identification called the ID algorithms.

5.1 Demystifying the do-calculus

The do-calculus is intimidating; it is unlikely that upon seeing the rules for the first time you will say "now I understand!" A useful analogy is the trigonometric identities one learns in high school. The first time you saw the Pythagorean identity "$\cos^2 x + \sin^2 x = 1$", you also probably didn't proclaim "now I understand!" But nonetheless you learned to apply the identity in solving more practical math problems. The same applies for using do-calculus to solve identification problems.

Moreover, while the fact the Pythagorean identity may not have been immediately obvious, you at least had an intuition of the cosine and sine functions. Similarly, you already understand the core elements of the do-calculus; the causal DAG, d-separation and its connection to conditional independence, and graph mutilation and its connection to interventions. If the do-calculus seem opaque, relax knowing that it is an extension of concepts where you already stand with solid footing.

Indeed, identification algorithms implemented in packages such as y0 apply the rules of the do-calculus for us. Further, these ID algorithms are complete[1], meaning if a query can be identified with the do-calculus, the ID algorithms will successfully do so.

But for purposes of building intuition, I'll introduce the rules and illustrate them on some simple causal DAGs. In these examples, we'll be focus on the target distribution Y under an intervention on X. We want to generalize to all DAGs. So we'll name two other nodes, Z and W. Z and W will allow us to cover cases where we have another potential intervention target Z and a node and any node W we'd like to condition upon. Further, while I'll focus on cases where Y, X, Z and W are the names of individual variables, but keep in mind that the three rules work when Y, X, Z, and W represent sets of variables.

5.2 Rule 1: Insertion or removal of observations

Suppose that, once again, you have a causal DAG and are interested in reasoning about $Y_{X=x}$ conditional on Z and W. Your task is to derive an estimand. Furthermore, suppose that in your DAG, the following condition is true:

> Y and Z are d-separated by X and W after the incoming edges to X are removed.

Rule 1 is as follows: in any graph that satisfies this condition, $P(Y_{X=x}=y|Z=z, W=w) = P(Y_{X=x}=y| W=w)$. What does this mean? It means that if I'm deriving an estimand for query on a graph where this condition holds, and I encounter the term $P(Y_{X=x}=y| W=w)$, I can "insert" the condition Z=z to get $P(Y_{X=x}=y|Z=z, W=w)$. If I encounter the term $P(Y_{X=x}=y|Z=z, W=w)$, I can remove Z=z to get $P(Y_{X=x}=y| W=w)$.

5.3 Rule 2: Exchange of an intervention for an observation

Again, suppose I had a causal DAG that met the following condition.

> Y and Z are d-separated by X and W after incoming edges in X and outgoing edges from Z have been removed.

[1] Huang, Yimin, and Marco Valtorta. "Pearl's calculus of intervention is complete." arXiv preprint arXiv:1206.6831 (2006).

iginal DAG (left), Y and Z are not d-separated by X and W. But after you remove incoming edges to X and outgoing edges from Z (right), the d-separation statement becomes true.

Rule 2 says that in that graph, and any graph that satisfies this condition, the following condition holds:

$$P(Y_{X=x,\ Z=z}=y\ |W=w) = P(Y_{X=x}=y\ |Z=z,\ W=w)$$

Here we can either exchange the intervention do(Z=z) in $P(Y_{X=x,\ Z=z}=y\ |W=w)$ for conditioning on the observation Z=z to get $P(Y_{X=x}=y\ |Z=z, W=w)$, or vice versa.

5.4 Rule 3: Insertion or removal of interventions

For rule 3, where are going to define Z as a set of two nodes Z_1 and Z_2[2]. Rule 3 is going to apply to an causal DAG that satisfies the following conditions:

1. After you remove all incoming edges to X, Z1 is not an ancestor of W.
2. After you remove all incoming edges to X and Z1, Y and Z are d-separated by X and W.

For any graph that satisfies this condition, rule 3 says the following equality holds:

$$P(Y_{X=x,\ Z=z}=y|W=w) = P(Y_{X=x}=y|W=w)$$

This rule allows you to insert do(Z=z) into $P(Y_{X=x}=y|W=w)$ to get $P(Y_{X=x,\ Z=z}=y|W=w)$ or remove do(Z=z) from $P(Y_{X=x,\ Z=z}=y|W=w)$ to get $P(Y_{X=x}=y|W=w)$.

5.5 Using the do-calculus for backdoor identification

Let's revisit backdoor identification of $P(I_{E=e})$ in the online gaming example. Our goal is to derive the backdoor formula $P(I_{E=e}) = \Sigma_g P(|\ S=s, G=g)P(G=g)$.

When I presented how to use parallel world-based "ignorability" reasoning, my goal as an author was for you to understand how that reasoning leads us to the backdoor formula. But with the do-calculus, my goal is not for you to understand the logic. Rather, my goal is for you to see how the do-calculus is applied. As I walk you through the steps, think about how an algorithm would work through these steps.

Here are the steps to showing $P(I_{E=e}=i)= \Sigma_g P(I=i|E=e, G=g) P(G=g)$ with the do-calculus.

1. $P(I_{E=e}=i)= \Sigma_g P(I_{E=e}=i, G_{E=e}=g)$ by the law of total probability.
2. $= \Sigma_g P(I_{E=e,\ G=g}=i)P(G_{E=e,\ I=i}=g)$ by way of *c-component factorization*[3].
3. $P(I_{E=e,\ G=g}=i) = P(I=i|E=e, G=g)$ by rule 2 of the do-calculus.
4. $P(G_{E=e,\ I=i}=g) = P(G=g)$ by rule 3 of the do-calculus.
5. Therefore, $P(I_{E=e}=i)= \Sigma_g P(I=i|E=e, G=g) P(G=g)$ by plugging 3 and 4 into 2.

[2] in the general case, these are two *sets* of nodes.
[3] Step 2 uses a factorization rule called c-component factorization. We touched on c-components in chapter 4; a c-component is just set of nodes that all share common causes. Factorizing over c-components is common in identification algorithms. See the references in the book notes at www.altdeep.com/p/causalAIbook.

Step 3 applies rule 2 of the do-calculus. We want to replace with $P(I_{E=e,\ G=g}=i)$ with $P(I=i|E=e, G=g)$, i.e., we want to use rule 2 to "exchange interventions for observations." Rule 2 tells us we can replace intervention subscripts $_{E=e,\ G=g}$ with conditions E=e, G=g, if we meet the following condition; both E and G have to become independent of I when we remove outgoing edges from E and G. In Figure ??? we see that with our simple graph, removing outgoing edges from E and G removes *all* the edges! Obviously in this path-free graph, both E and I are d-separated from G. So we can apply rule 2.

Step 4 applies rule 3. Here, we want to replace $P(G_{E=e,\ G=g})$ with P(G), so we want rule 3's "removal of interventions." To remove the intervention subscripts $_{E=e}$ and $_{G=g}$, rule 3 requires the DAG meet the condition that both E and G are independent of I when outgoing edges from E and G are removed. Again, in our case, this produces an edgeless graph! So the d-separation condition holds and we can apply rule 3.

5.6 The ID Algorithms

The rules of the do-calculus themselves don't tell us how to apply them to derive an estimand, as we did with the backdoor estimand derivation. Fortunately, we can rely on a series of interrelated algorithms for graphical identification called the ID algorithms. These algorithms provide *graphical criteria* for identification, meaning they can check if a level i query is identifiable from a given DAG by checking if the DAG has certain properties, and if it does, they can return the estimand by composing various mathematical operations over a level <i probability distribution. We can think of these algorithms is a recursively applying the rules of the do-calculus. The ID algorithms will give us the correct estimand if one can be graphically identified (formally, they are *sound* and *complete*).

The mechanics of the ID algorithms are too advanced for an introductory text (see, www.altdeep.ai/p/causalai for a list of ID algorithm references). But the core concepts are those of the do-calculus, graphical concepts you could implement using graph utilities found in any commonly use graph library, such as Python's networkx:

1. A DAG representation (e.g., networkx DiGraph class combined with acyclicity checks)
2. The ability to define sets of nodes based connecting paths (like c-components).
3. The ability to check if sets of nodes are d-separated (e.g., networkx's `d_separated` function)
4. The ability to remove edges from the DAG (e.g., networkx's remove_edges_from function)

In this tutorial, we've been using the library Y0 to derive estimands. Y0 has been using its implementation of the ID algorithms to derive those estimands.

We've seen the backdoor estimand. In the next section, I present two other estimands of intervention that we can derive using the do-calculus.

6 Other notable level 2 estimands; the front door and napkin

estimand

The backdoor estimand is the most commonly used level 2 estimand. But the do-calculus is complete; we can use it to derive other level 2 estimands. In this section, we look at two more do-calculus derivable leve 2 intervention query estimands; the front door estimand and the napkin estimand.

6.1 The front-door estimand

In our online gaming example, suppose you were not able observe guild membership. Then you would be out of luck, at least in terms of backdoor identification of $P(I_{E=e})$. However, supposer we had a *mediator* between side-quest engagement (E) and In-game purchases (I) – a node on the between E and I. Specifically, our mediator represents *won virtual items* (W), as seen in Figure ???.

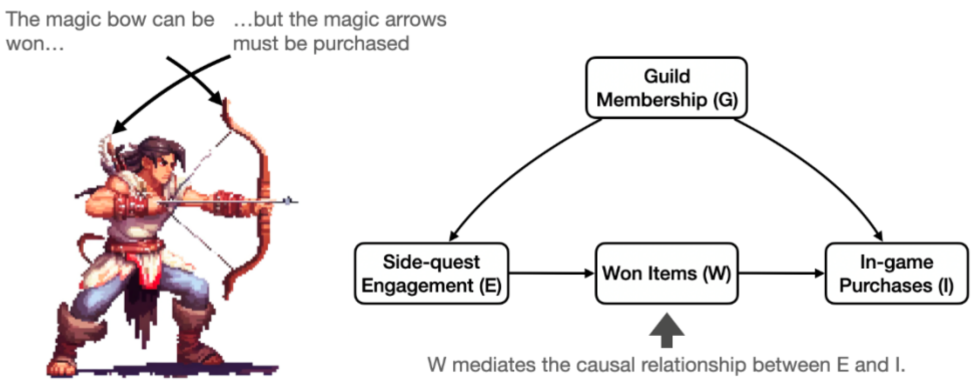

The magic bow can be won…
…but the magic arrows must be purchased
Guild Membership (G)
Side-quest Engagement (E)
Won Items (W)
In-game Purchases (I)
W mediates the causal relationship between E and I.

Figure 9. 17 Side-quest engagement leads to winning virtual items like this magic bow. Won items drive more in-game purchases, such as magic arrows for the magic bow. So we introduce a mediator "won items" on the causal path between side-quest engagement and in-game purchases.

The idea of won items is as follows. When a player successfully completes a side-quest, they win a virtual item. The more they side-quests they finish, the more items they earn. Those *won* virtual items and *purchased* virtual items can complement one another. For example, winning a magic bow motivates purchases of magical arrows. Thus, the amount of won items a player has influences the amount of virtual items they purchase.

Given this graph, we can derive the front door estimand:

Listing 6 Deriving the front door estimand in y0

```
from y0.graph import NxMixedGraph as Y0Graph
from y0.dsl import P, Variable
G = Variable("G")
```

```
E = Variable("E")
I = Variable("I")
W = Variable("W")
e = E

dag = Y0Graph.from_edges(
    directed=[
        (G, E),
        (G, I),
        (E, W),
        (W, I)
    ]
)

query=P(I @ e)
base_distribution = P(I, E, W)

identification_task = Identification.from_expression(
    graph=dag,
    query=query,
    estimand=base_distribution)

identify(identification_task)
```

This code will return the output in Figure ???.

$$\sum_{G,W} P(I|E,G,W) \sum \sum_{E,I,W} P(E,I,W) P(W|E,G)$$

In our notation, this is $\sum_g \sum_w P(W=w|E=e,G=g) \sum_{\epsilon,\imath,\omega} P(I=i|E=\epsilon,G=g,W=w) P(E=\epsilon,W=\omega,I=\imath)$. Simplifying as before, we get the front door estimand:

$$P(I_{E=e}=i) = \sum_w P(W=w|E=e) \sum_\varepsilon P(I=i|E=\varepsilon, W=w) P(E=\varepsilon)$$

Note there is an outer summation over W and an inner summation over all values of E (with each value of E denote ε, distinct from the intervention value e).

The rough intuition behind the front-door estimand is that the statistical association between side-quest engagement and in-game purchases comes from both the direct causal path and the path through the backdoor confounder guild membership (G). The front-door estimand uses the mediator to determine how much of that association is due to the direct causal path; the mediator acts as a gauge of the flow of statistical information through that direct causal path.

The do-calculus derivation of the front-door estimand takes several steps and involves repeated substitutions using rule 2 and rule 3. I'll omit the derivation and focus on a key benefit of the estimand – that it does not require any backdoor adjustment; it provides a means of non-parametric identification when there is not a valid set of backdoor adjustors observed in the data.

6.2 Comparing assumptions between the backdoor and the front door

The library DoWhy is useful for detecting whether your level 2 causal effect query is backdoor or front door identified, and showing you the assumptions you are relying on in either case.

```
import pandas as pd
import dowhy
from dowhy import CausalModel

url =
'https://raw.githubusercontent.com/altdeep/causalML/master/datasets/sideque
sts_and_purchases_full_mediator.csv'
df = pd.read_csv(url)

df.rename(columns={"Side-quest Engagement": "E",
                   "Guild Membership": "G",
                   "In-game Purchases": "I",
                   "Won Items": "W"},
          inplace=True)

dag = """digraph {
    G -> E;
    G -> I;
    E -> W;
    W -> I;
    }"""

model = CausalModel(
    data=df.drop(columns=["User ID"]),
    treatment='E',
    outcome='I',
    graph=dag
)

print(model.identify_effect())
```

This returns output that summarizes the assumptions of backdoor inference. For Estimand 1, it states an unconfoundedness assumption "if U→{E} and U→I then P(I|E,G,U) = P(I|E,G)." This means that there aren't any d-connecting paths between I and E through latent confounders U that aren't d-separated by G. For the front door estimand, it highlights three assumptions. Ther first is "W intercepts (blocks) all directed paths from E to I". The second, "If U→{E} and U→{W} then P(W|E,U) = P(W|E)", and third " If U→{W} and U→I then P(I|W, E, U) = P(I|W, E)", assume there are not any d-connected paths through latent confounders from E to W and from I to W. We can use domain knowledge and statistical tests to see which estimand has more reliable assumptions in a given analysis.

The backdoor and front door estimands are not the only two level 2 estimands of potential interest. We can derive these estimands using the do-calculus and graphical identification algorithms.

Next, we move on from identifying interventional (level 2) queries to counterfactual (level3)

7 Counterfactual (level 3) identification with counterfactual graphs and SWIGs

To motivate identification of counterfactual (level 3) estimands, we'll introduce a new case study.

When you open Netflix, you see the Netflix dashboard. On the dashboard, there are several forms of recommended content. Two types of recommended content are "Top Picks For You", which is a personalized selection of shows and movies that Netflix's algorithms predict you will enjoy based on your past view behavior and ratings, and "Because You Watched", which recommends content based on things you watched recently. Your model of this system includes the following variables:

- **T**: A variable for the recommendation policy that selects a subscriber's "Top Picks for You" content. For simplicity, we'll consider policy "+t" and all other policies "-t" meaning "not t".
- B: A variable for the recommendation policy that selects a subscriber's "Because You Watched" content. Again, we'll simplify this to a binary variable with policy "+b" and all other policies "-b" as in "not b."
- V: The amount of engagement that a subscriber has with the content recommended by the "Top Picks for You" content.
- W: The amount of engagement that a subscriber has with the content recommended by the "Because You Watched" content.
- A: Attrition, i.e., whether a subscriber eventually leaves Netflix.
- C: Subscriber context, i.e., the type of subscriber we are dealing with.

Recommendation algorithms always take the profile of the subscriber into account along with the viewership history, so subscriber profile C is a cause of both recommendation policy variables T and B.

In this section, we'll use y0 to analyze this problem at various levels of the hierarchy. We start by visualizing the graph.

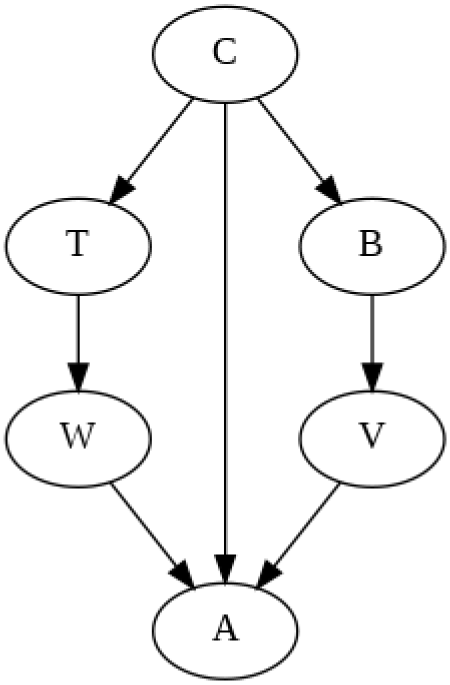
Listing 7 Plot the recommendation DAG

```
!pip install git+https://github.com/y0-causal-inference/y0.git@id_star
!apt-get install graphviz libgraphviz-dev pkg-config
!pip install pygraphviz
!wget
https://raw.githubusercontent.com/altdeep/causalML/master/book/id_utilities
.py -O id_utilities.py    #A
from id_utilities import *

T = Variable("T")     #B
W = Variable("W")     #B
```

```
B = Variable("B")     #B
V = Variable("V")     #B
C = Variable("C")     #B
A = Variable("A")     #B
t, a, w, v, b = T, A, W, V, B     #B
dag = Y0Graph.from_edges(directed=[     #C
    (T, W),     #C
    (W, A),     #C
    (B, V),     #C
    (V, A),     #C
    (C, T),     #C
    (C, A),     #C
    (C, B)     #C
])     #C
gv_draw(dag)     #D
```

#A Download some helper functions.
#B Define variables for the model.
#C Create the graph.
#D Plot the graph.



$$\begin{tikzcd}
 & C \arrow[dl] \arrow[dr] \arrow[ddd] & \\
T \arrow[d] & & B \arrow[d] \\
W \arrow[dr] & & V \arrow[dl] \\
 & A &
\end{tikzcd}$$

Figure 9. 18 Causal DAG for the recommendation algorithm problem.

We'll use y0 to analyze this model at various levels of the hierarchy. As a preliminary investigation, you may look at the causal effect (ATE) of the "Top Picks for You" content on attrition E($A_{T=+t}$ – $A_{T=-t}$). Given attrition A has a binary outcome, we can write this as P($A_{T=+t}$=+a) – P($A_{T=-t}$=+a). Focusing on P($A_{T=-t}$=+a), we know right away that we can identify this via both the backdoor and the front door. So let's move on to an interesting counterfactual query called *effect of treatment on the treated* (ETT).

7.1 Effect of treatment on the treated

Recall that you get the ATE directly (without the need for estimating a level 1 estimand) from a randomized experiment. Suppose you ran such an experiment a cohort of users and it showed a favorable ATE, that +t has a favorable impact on W and A relative to -t. So your team deploys the policy.

But the policy is selected for users based on the subscriber's background C. C is also a cause of subscriber attrition. While randomly selected subscribers got +t in the experiment, a certain type of subscriber gets the policy in production. Is the policy effective for these users? Perhaps, if the policy were not introduced, they would have responded more to some other algorithms (e.g., the "Because you watched…" algorithm) and would still have had favorable outcomes in terms of attribution. In other words, it's possible that while +t had a favorable result for people who were assigned the policy in the experiment, it might have a different result for people are assigned the policy in the wild.

The counterfactual query that addresses this is a counterfactual version of the ATT called effect of treatment on the treated (ETT, or sometimes ATT, as in *average treatment effect on the treated*). We write this as counterfactual query E($A_{T=+t}$ – $A_{T=-t}$|T=+t), as in *"for people who saw policy +t, how much more attrition do they have relative to what they would have if they had seen -t?"*. Decomposing for binary A as we did with the ATE, we can write this as P($A_{T=+t}$=+a|T=+t) – P($A_{T=-t}$=+a|T=+t). The first term simplifies to P(A=+a|T=+t) by the law of consistency. So we focus on the second term P($A_{T=-t}$=+a|T=+t).

In the case of binary A, we can identify the ETT using graphical identification (for non-binary A, we'll see more level 3 assumptions are needed). To do graphical identification for counterfactuals, we can use ID algorithms with counterfactual graphs.

7.2 ID algorithms and do-calculus for counterfactual identification

Y0 can derive an estimand for ETT using a graphical identification algorithm IDC*.

Listing 8 Identifying ETT with a graphical identification algorithm

```
from y0.algorithm.identify.idc_star import idc_star

idc_star(
    dag,
    outcomes={A @ -t: +a},     #A
    conditions={T: +t}     #B
)
```

#A Counterfactual outcome $A_{T=-t}$ = +a

#B Corresponds to the actual outcome T = +t

The algorithm will print out its steps, including applying rules of the do-calculus, and recursively calling IDC* and a related algorithm called ID*. It will produce a rather verbose level 2 estimand. We can then apply level 2 ID algorithms to get a level 1 estimand, that will simplify to the following: P(A=+a| C= c, T = -t)P(C=c|T=+t). I'll show a much simpler derivation in the next section.

For now, it's enough to get a high-level understanding of the how the ID algorithms work for counterfactual queries. Recall that the do-calculus relies on d-separation-based reasoning over mutilated graphs. I've shown you that when reasoning about counterfactuals, our graph of choice is the parallel world graph. Indeed, have y0 make a parallel world graph for us.

Listing 9 Plotting the parallel world graph with y0

```
from y0.algorithm.identify.cg import make_parallel_worlds_graph
parallel_world_graph = make_parallel_worlds_graph(
    dag,
     {frozenset([+t])}
)
gv_draw(parallel_world_graph)
```

#A The make_parallel_worlds_graph method takes an input DAG and sets of interventions. It constructs a new world for each set.
#B The helper function visualizes the graph in a familiar way.

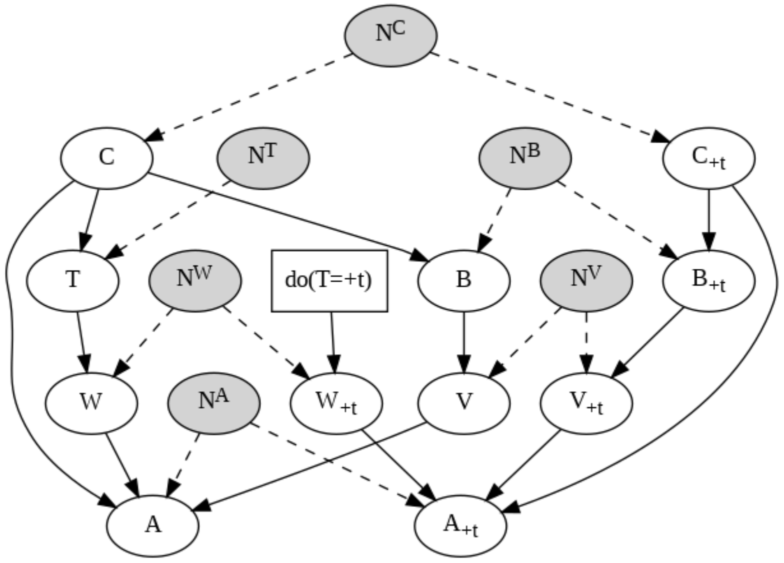


Figure 9. 19 Parallel world graph drawn by y0 (and graphviz). In this version of the parallel world graph, the subscripts indexes a world. For example $_{+t}$ indexes the world where the intervention do(T=+t) is applied.

This graph differs slightly from the ones I've drawn because the algorithm applies the subscript for an intervention to every node in the world where the intervention occurred; the subscript indexes all the variables in a world. It's up to us to reason that C from one world and C_{+t} from another must have the same outcomes since C_{+t} is not affected by its world's intervention do(T=+t).

Now recall that the problem with the parallel worlds graph is that we can't use d-separation with it. For example, d-separation suggests that C and C_{+t} are conditionally independent given their common exogenous parent N^C, but we just articulated that C has and C_{+t} must be the same; if C has a value, C_{+t} must have the same value, thus the are *perfectly* dependent.

We can remedy this with the counterfactual graph. The counterfactual graph is created by using the graph and the counterfactual query to understand which nodes across worlds in the parallel world graph are actually the same, an then combining those nodes. The resulting graph contains nodes across parallel worlds that are relevant to the events in the query. We can use y0 to create the counterfactual graph for events $A_{T=-t}$=+a and T=+t.

Listing 10 Counterfactual graph events $A_{T=-t}$=+a and T=+t

```
from y0.algorithm.identify.cg import make_counterfactual_graph

events = {A @ -t: +a, T: +t}     #A
cf_graph, _ = make_counterfactual_graph(dag, events)
gv_draw(cf_graph)
```

#A Counterfactual graphs work with event outcomes in theequery. For P($A_{T=-t}$=+a|T=+t), we want events $A_{T=-t}$ =+a and T=+t.

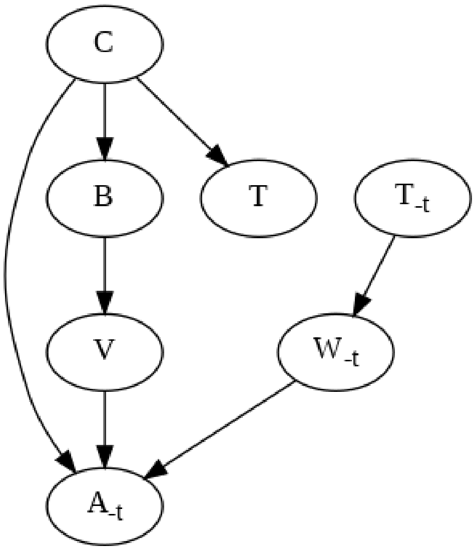


Figure 9. 20 Counterfactual graph for events produced by y0 (and graphviz). T_{-t} corresponds to the

intervention do(T=-t).

At a high-level, the ID algorithms in y0 do graph-based counterfactual inference applying the do-calculus-based reasoning to the counterfactual graph. First it finds a level 2 estimand for a level 3 query. From there, you can use experimental data to answer the level 2 terms in the estimand, or you can attempt to further derive them to level 1 estimands from the level 2 terms.

Using the counterfactual graph

The main use-case for counterfactual graphs is identification, and in that case you are usually relying on an ID algorithm that calls the graph under the hood. I expose the graph here to give you a high-level intuition for how the ID algorithms work. That said, if you were building a probabilistic machine learning model for counterfactual reasoning, and wanted to make use of conditional independence between counterfactual variables in parameter learning and inference, the counterfactual graph could be a useful tool.

We can also use graphical identification for more advanced queries. For example, suppose you want to isolate how T affects A from how B affects A. You want to focus on users where B was -b. You find the data from a past experiment where "Because you watched…" policy B was randomized. You take that data and zoom in on participants in the experiment who were assigned -b. The outcome of interest in that experiment was V, the amount of engagement with recommended, so you have the outcomes of $V_{B=-b}$ for those subscribers of interest. With this new data, you expand your query from $P(A_{T=-t}=+a|T=+t)$ to $P(A_{T=-t}=+a|T=+t, B=-b, V_{B=-b}=v)$, including $V_{B=-b}=v$ because it is helpful in predicting attrition. Now you have three parallel to reason over, the actual, the world with do(T=+t) and the world with do(B=-b).

Listing 11 Create parallel world graph for do(T=+t) and do(B=-b)

```
parallel_world_graph = make_parallel_worlds_graph(
        dag,
         {frozenset([-t]), frozenset([-b])}
)
gv_draw(parallel_world_graph)
```

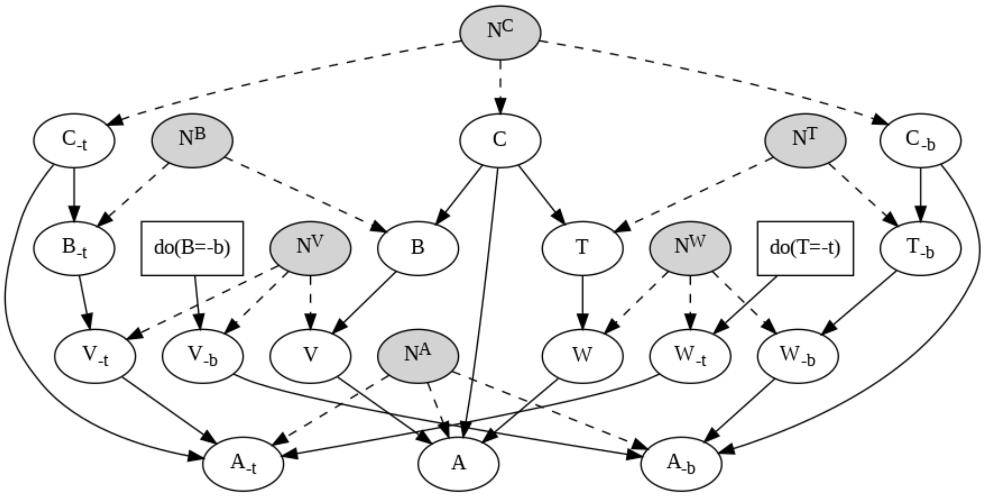


Figure 9. 21 Parallel world graph with actual world T=+t hypothetical worlds do(T=-t) and do(B=-b).

Notably, the query P($A_{T=-t}$=+a|T=+t, B=-b, $V_{B=-b}$=v) collapses the parallel world graph to the same counterfactual graph as P($A_{T=-t}$=+a|T=+t).

Listing 12 Counterfactual graph for expanded expression

```
joint_query = {A @ -t: +a, T: +t, B: -b, V @ -b: +v}
cf_graph, _ = make_counterfactual_graph(dag, joint_query)
gv_draw(cf_graph)
```

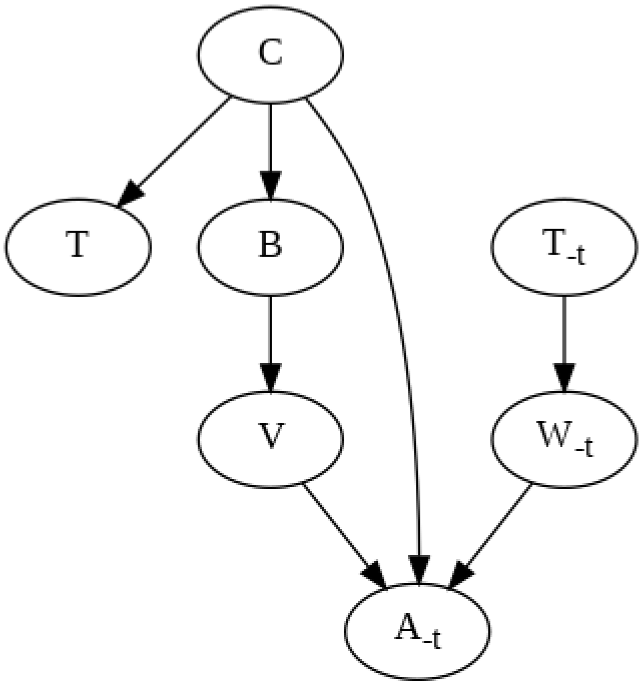


Figure 9. 22 The counterfactual graph for $P(A_{T=-t}=+a|T=+t, B=-b, V_{B=-b}=v)$ is the same as for $P(A_{T=-t}=+a|T=+t)$.

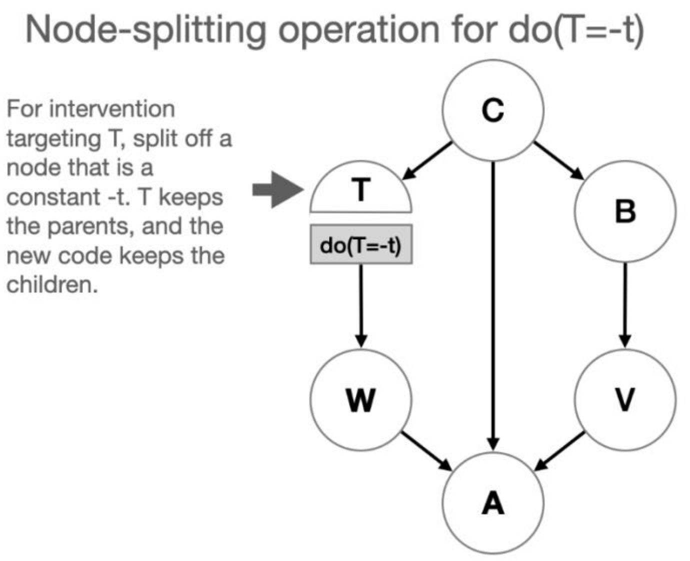
Next I introduce another graph-based approach called single-world intervention graphs.

7.3 Counterfactual identification with single-world intervention graphs (SWIGs)

Single-world intervention graphs (SWIGs) provide an alternative to counterfactual identification with counterfactual graphs. Like a counterfactual graph, we construct a SWIG using the original causal DAG and the causal query. We'll use the Netflix recommendation example to construct a SWIG for the interventions do(T=-t) and do(B=-b). To construct a SWIG from a causal DAG, follow the following recipe.

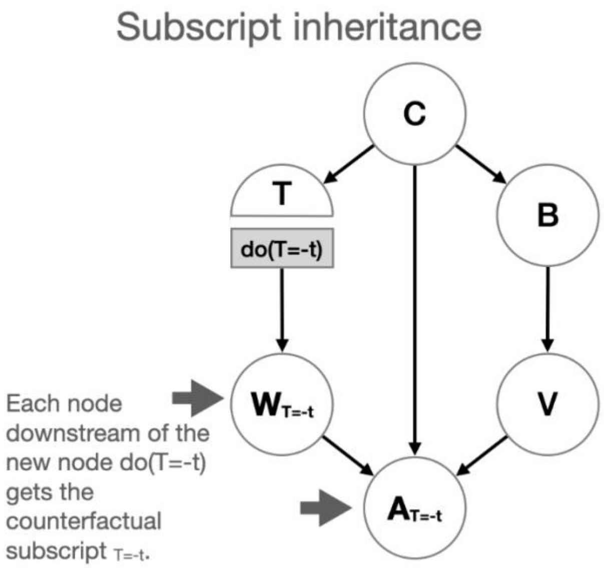
NODE-SPITTING OPERATION

We have the intervention that targets do(T=+t). We implement the intervention with a special kind of graph mutilation called a *node-splitting operation*. We split a new node off T. T still represents the same variable as in the original graph. The new node represents a constant, the intervention value +t. T keeps its parents (in this case C) but loses its children to the new node (in this case W).


Node-splitting operation for do(T=-t)
For intervention targeting T, split off a node that is a constant -t. T keeps the parents, and the new code keeps the children.
C
T
do(T=-t)
B
W
V
A

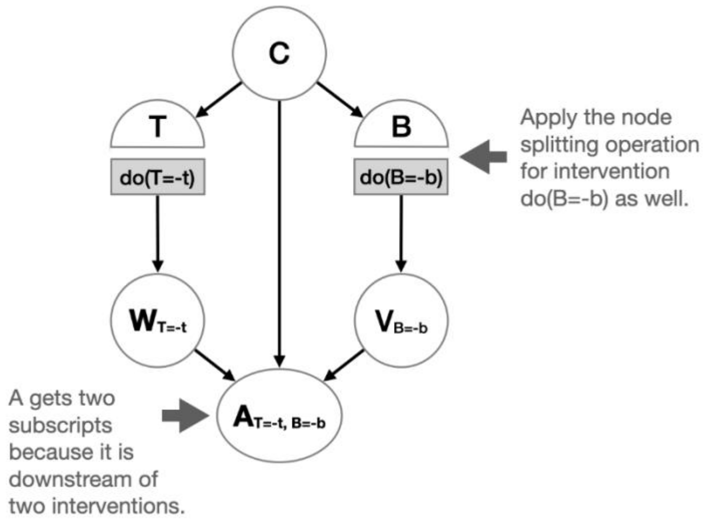
SUBSCRIPT INHERITANCE

Next, every node downstream of the new node inherit the new node's values as a subscript.


Subscript inheritance
C
T
do(T=-t)
B
W_{T=-t}
V
A_{T=-t}
Each node downstream of the new node do(T=-t) gets the counterfactual subscript T=-t.

Repeat for each intervention

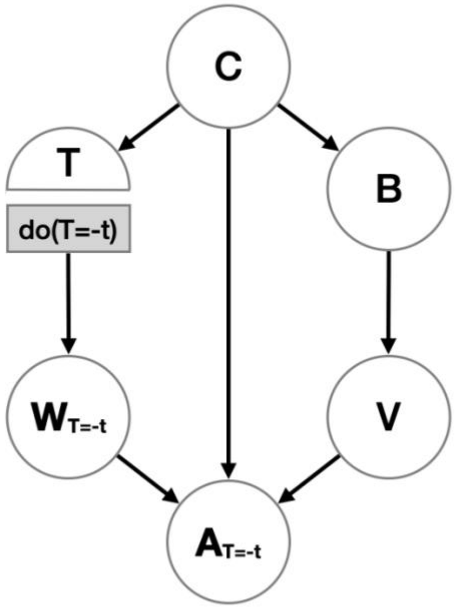
We repeat this process for each intervention. In this case, we apply do(B=-b), convert V to $V_{B=-b}$ and A to $A_{T=-t,B=-b}$.


C
T
do(T=-t)
B
do(B=-b)
Apply the node splitting operation for intervention do(B=-b) as well.
$W_{T=-t}$
$V_{B=-b}$
A gets two subscripts because it is downstream of two interventions.
$A_{T=-t, B=-b}$

Like the counterfactual graph, the SWIG contains counterfactual variables and admits d-separation. With these properties, we can do identification.

7.4 Identification with SWIGs

Suppose we were interested in ETT and want to identify $P(A_{T=-t}=+a|T=+t)$. We derive the SWIG in Figure???.



With this graph, we can identify $P(A_{T=-t}=+a|T=+t)$ using the ignorability trick I introduced in

1. $P(A_{T=-t}=+a|T=+t) = \Sigma_c P(A_{T=-t}=+a, C_{T=-t}=c|T=+t)$ by the law of total probability.
2. $\Sigma_c P(A_{T=-t}=+a, C_{T=-t}=c|T=+t) = \Sigma_c P(A_{T=-t}=+a, C=c|T=+t)$ since C is not affected by interventions on T.
3. $\Sigma_c P(A_{T=-t}=+a, C=c|T=+t)$ factorizes into $\Sigma_c P(A_{T=-t}=+a|C=c, T=+t) P(C=c | T=+t)$ by the chain rule of probability.
4. $P(A_{T=-t}=+a|C=c, T=+t) = P(A_{T=-t}=+a|C=c, T=-t)$, again by the ignorability trick.
5. And as before $P(A_{T=-t}=+a|C=c, T=-t) = P(A =+a|C=c, T=-t)$ by the law of consistency. Thus $P(A_{T=-t}=+a|T=+t) = \Sigma_c P(A =+a|C=c, T=-t) P(C=c | T=+t)$

The magic happens in the ignorability trick step 4, where C's d-separation of $A_{T=-t}$ and T let me change T=+t to T=-t. Notice the same d-separation exists in the counterfactual graph we derived for $P(A_{T=-t}=+a|T=+t)$, shown in Figure 9-17. The difference is that deriving the SWIG is easy, while deriving the counterfactual graph is nuanced and one usually uses an algorithm like `make-cg` in y0.

7.5 The single world assumption

The node-splitting operation makes a new assumption. If you are going to node-split a variable X, then you are assuming it is possible to know what value X is naturally going to take, but also possible for you to intervene on X before it realizes that value. Imagine that in our Netflix example, that given a subscriber had profile C=c, the recommendation algorithm was about to assign a subscriber a policy +t for recommending "Top picks for you", but before that policy went into effect, you intervene and artificially change it to -t. That intervention can't have any side effects that would affect the value T would take (+t) had you not applied the intervention.

That assumption allows you to avoid the need of creating additional worlds to reason over, you can condition on outcome T=+t and intervene do(T=-t) in a single world. The reduces the number of counterfactual queries you can answer, but proponents of SWIGs suggest this is a strength because it limits you to counterfactuals that can be validated by experiments.

SWIGS VS COUNTERFACTUAL GRAPHS

Counterfactual graphs and SWIGs are both useful in graphical identification of counterfactuals but differ in a couple of key ways. Firstly, counterfactual graphs reason about events like {T=+t} while SWIGs reason on variables. Secondly, SWIGs are limited to a narrow set of counterfactuals where the single-world assumption holds, while you can draw counterfactual graphs even for queries that can't be graphically identified.

7.6 SWIGs are Pyro's default model of intervention

I pay special attention to the SWIG because the are Pyro's default model of intervention. Consider for example our outline game model. To model P($I_{E=0}$), you apply pyro's `do` function to apply the intervention do(E=0). However, the original sample site for E in the model is preserved. That means you can still condition on E=1 with `pyro.condition` to the variable being intervened upon. Thus, we can model P($I_{E=0}$|E=1) with `condition(do(model))` (or equivalently `do(condition(model))`). I illustrate using numpyro (though the code will work in pyro with only small tweaks).

Listing 13 Generating from P($I_{E=0}$) vs P($I_{E=0}$|E=1) in Pyro

```
import jax.numpy as np
from jax import random
from numpyro import sample
from numpyro.handlers import condition, do
from numpyro.distributions import Bernoulli, Normal
from numpyro.infer import MCMC, NUTS
import matplotlib.pyplot as plt

rng = random.PRNGKey(1)

def model():     #A
    p_member = 0.5
    is_guild_member = sample(     #A
        "Guild Membership",     #A
        Bernoulli(p_member)     #A
    )     #A
    p_engaged = (0.8*is_guild_member + 0.2*(1-is_guild_member))     #A
    is_highly_engaged = sample(     #A
        "Side-quest Engagement",     #A
        Bernoulli(p_engaged)     #A
    )     #A
    p_won_engaged = (.9*is_highly_engaged + .1*(1-is_highly_engaged))     #A
    high_won_items = sample("Won Items", Bernoulli(p_won_engaged))    #A
    mu = (    #A
        37.95*(1-is_guild_member)*(1-high_won_items) +    #A
        54.92*(1-is_guild_member)*high_won_items +    #A
        223.71*(is_guild_member)*(1-high_won_items) +    #A
        125.50*(is_guild_member)*high_won_items    #A
    )    #A
    sigma = (    #A
        23.80*(1-is_guild_member)*(1-high_won_items) +    #A
        4.92*(1-is_guild_member)*high_won_items +    #A
        5.30*(is_guild_member)*(1-high_won_items) +    #A
        53.49*(is_guild_member)*high_won_items    #A
    )    #A
    in_game_purchases = sample("In-game Purchases", Normal(mu,
sigma))    #A

intervention_model = do(model, {"Side-quest Engagement":
np.array(0.)})    #B
intervention_kernel = NUTS(intervention_model)    #C
intervention_model_sampler = MCMC(    #C
    intervention_kernel,    #C
    num_samples=5000,    #C
    num_warmup=200    #C
)    #C
intervention_model_sampler.run(rng)    #C
intervention_samples = intervention_model_sampler.get_samples()    #C
int_purchases_samples = intervention_samples["In-game Purchases"]    #C
cond_and_int_model = condition(    #D
    intervention_model,    #D
     {"Side-quest Engagement": np.array(1.)}    #D
)    #D
int_cond_kernel = NUTS(cond_and_int_model)    #E
int_cond_model_sampler = MCMC(    #E
    int_cond_kernel,    #E
    num_samples=5000,    #E
    num_warmup=200    #E
)    #E
int_cond_model_sampler.run(rng)    #E
int_cond_samples = int_cond_model_sampler.get_samples()    #E
int_cond_purchases_samples = int_cond_samples["In-game Purchases"]    #E

plt.hist(    #F
    int_purchases_samples,    #F
    bins=30,    #F
    alpha=0.5,    #F
    label='$P(I_{E=0})$'    #F
)    #F
plt.hist(    #F
    int_cond_purchases_samples,    #F
    bins=30,    #F
    alpha=0.5,    #F
    label='$P(I_{E=0}|E=1)$'    #F
)    #F
plt.legend(loc='upper left')    #F
plt.show()    #F
```

#A A version of the online gaming model.

#B Applying the do operator to model P($I_{E=0}$)
#C Apply inference to sample from P($I_{E=0}$)
#D Now apply the condition operator to sample from P($I_{E=0}$|E=1)
#E Apply inference to sample from P($I_{E=0}$|E=1)
#F Plot samples as histograms

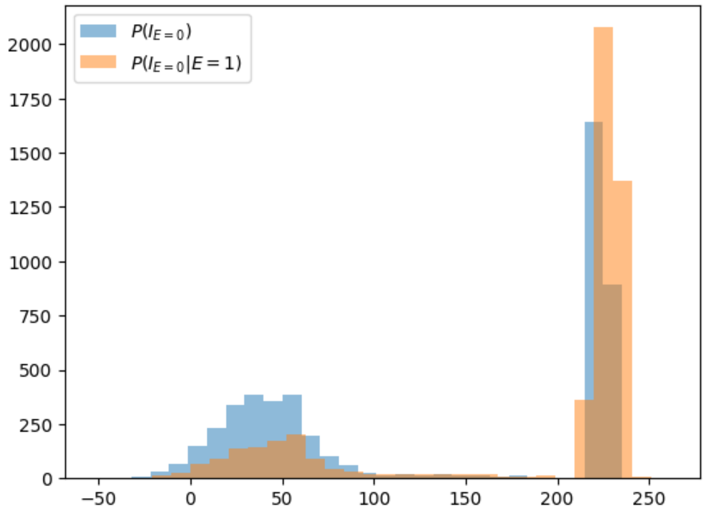


Figure 9. 23 Histograms of samples from P($I_{E=0}$) and P($I_{E=0}$|E=1) generated in Pyro.

Pyro uses SWIGs by default because if you are modeling just an intervention, the results are the same. Similarly, it doesn't interfere with an implementation of counterfactual inference using abduction-action-prediction with an SCM. But Pyro uses the SWIG to allow you to condition and intervention on the same variable without having to create parallel worlds. It extends SWIGs beyond the core use case of identification to probabilistic machine learning.

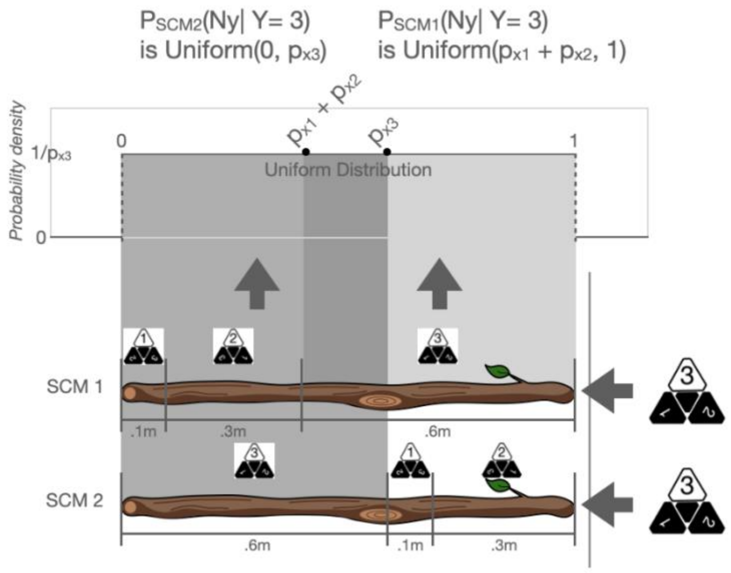
8 Counterfactuals require level 3 assumptions

The causal DAG is a level 2 modeling assumption. The causal hierarchy theorem tells us that the graph in general is not sufficient to identify level 3 counterfactual queries. For counterfactual identification from level 1 or level 2 distributions, you need level 3 assumptions.

8.1 What is a level 3 assumption?

In simple terms, a level 3 assumption as any causal assumption that you can't represent with a simple causal DAG. More formally, we can reason about the ground truth SCM for your model.

If you are trying to identify a counterfactual from a level 1 or level 2 distribution, you don't have an SCM (if you did you would use it to directly infer your counterfactual using the abduction-action-prediction algorithm). But you can assume there is an unknown ground truth SCM. Recall that multiple SCMs could entail the same DAG and level ½ distributions. I illustrated this with the following figure.



24 Two SCMs, with the same DAG, would produce different conditional distributions on the exogenous variable, thus produce different counterfactual inferences.

In the above figure we see two SCMs, from a class of 3*2*1 = 6 SCMs that all entail the same DAG and distribution on a three-sided die. Given a die outcome, they induce different conditional distributions of the exogenous variable given the observed variable. Thus, for a counterfactual inference query, they get different results in the abduction step of the abduction-action-prediction algorithm for computing counterfactual predictions. Thus, with the DAG alone, you don't know which of this class of SCMs would produce the correct prediction – this is the very definition of non-identifiability.

In general, there is a many-to-one relationship between the SCM and the DAG, as in the following figure

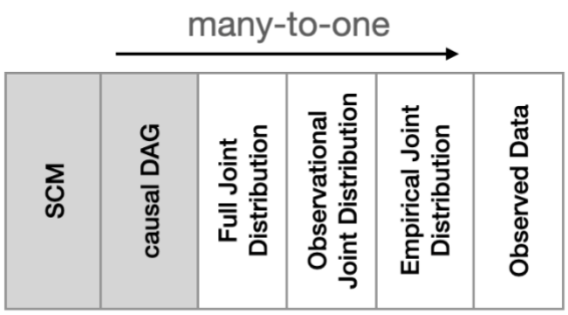

many-to-one
SCM
causal DAG
Full Joint Distribution
Observational Joint Distribution
Empirical Joint Distribution
Observed Data

Figure 9. 25 The challenge of identification stems from challenge of picking the correct one out of the many.

Different SCMs that entail the same DAG and observational distribution, could have produce different answers to a counterfactual query.

Thus, a level 3 assumption is any assumption that reduces that class of possible SCMs. The most common examples of level 3 information is assuming the SCM belongs to a canonical class of SCM, such as *linear additive SCMs*. One practically useful level 3 assumption is called monotonicity, discussed in the next subsection.

8.2 But didn't we just do graphical identification with counterfactual graphs and SWIGs?

For our counterfactual graph-based identification of the ETT and related queries worked because T had a binary outcome -t and +t. If the T were non-binary, counterfactual graph-based identification wouldn't work.

In a sense, the fact that T is binary is model assumption that you can't see on the DAG, thus we can think of it as a level 3 assumption. To get an intuition as to why identification works in the binary case, consider what would happen in the stick breaking example 24 if the stick broke into two regions instead of three (like for a coin flip). In this case, the the exogenous variables are Uniform(0, p) or Uniform(p, 1), which are equivalent. "Binary-ness" enables graphical identification by reducing the space of counterfactual possibilities. But if you have a non-binary variable, you can rely on graphical identification if you can reduce it to a binary outcome.

SWIG-based identification of the ETT relied on the single-world assumption, which I argue is a level 3 assumption[4]. Note that with this assumption, identification doesn't rely on T being binary.

[4] In a conversation I had with Thomas Richardson, co-author of the SWIG paper, he called it a "level 2.5 assumption", focusing on the ability to falsify the assumption with (level 2) experiments. I attend it is as level 3, as it is an assumption about mechanism that restricts the class of possible SCMs consistent with the DAG and joint distribution. See www.altdeep.com/p/causalAIbook for comments and references.

8.3 You can't graphically identify counterfactuals that condition on outcome!

Suppose that instead of the ETT term $P(A_{T=-t}=+a|T=+t)$, you were interested in $P(A_{T=-t}=+a|T=+t, A=+a)$, answering the question "Given a subscriber exposed to policy +t and later unsubscribed, would they still have unsubscribed had they not received that policy?" Relatedly you could be interested in $E(A_{T=+t} - A_{T=-t}|T=+t, A=+a)$ sometimes called *counterfactual regret*, which captures the amount the policy +t contributed to an unsubscribed individual's decision to unsubscribe.

Let's describe queries like $P(A_{T=-t}=+a|T=+t, A=+a)$ as queries that condition on the outcome. But that I mean that we are interested in the probability of the outcome of A in a world where do(T=-t) is applied, but we condition that probability on another outcome of A in a world where T=+t is observed. In these cases, SWIGs and counterfactual graphs are not sufficient, we absolutely must have level 3 assumptions. Indeed, these are precisely the kind of "how might things have turned out differently?" counterfactual questions that are the most interesting, and the most central to how humans reason and make decisions.

8.4 Example: Monotonicity and “Let Sleeping Dogs Lie”

Uplift modeling models how individuals respond to some sort of communication intended to induce some sort of behavior (e.g., buying a product, clicking on a link, voting in an election). The goal is to quantify how much a given communication drove an individual behavior.

Recall that in uplift modeling, you assume your target audience segments as follows:

- **Persuadables**: People whose chance of doing the target behavior increases when you engage them. Ideally, you would engage solely with this group.
- **Sure things**: People who will engage in the target behavior regardless of whether you engage them. Engaging with this group is a waste of resources.
- **Lost causes**: People who will not engage in the target behavior regardless of whether you engage them. Engaging with this group is a waste of resources.
- **Sleeping dogs**: People whose chances of doing the target behavior *go down* when you engage them. Engaging with this group is the worst, you are spending resources to lose customers.

Consider the Netflix example. For Netflix, maintaining subscribers is paramount. So we can view the ultimate goal of their algorithms as preventing attrition (i.e., we want A to be -a. Suppose there is a baseline policy T=-t, but a certain individual was assigned a newly released personalized policy T=+t. Unfortunately, they subsequently canceled their subscription (A=+a). Would they have canceled had they not been assigned that policy?

Given our segments, either the user was a lost cause or a sleeping dog. If they were a lost cause, they would have unsubscribed regardless of what policy we assigned them. If they were a sleeping dog, our new policy irritated them and they unsubscribed in frustration; had they been given the baseline policy, they would have stayed. Thus, the counterfactual is not identified, not without a level 3 assumptoin.

Monotonicity

A common level 3 assumption is called monotonicity. It means that a change in the cause only drives a change in effect in one direction. For example, you might assume that taking an aspirin only relieves headaches or does nothing, and it never causes headaches or makes headaches worse.

In our uplift modeling case, if you assume monotonicity, you are assuming that the user either responds weil to the change in policy or completely ignores it, that there are no users who throw up their hands and unsubscribe due to the policy. That level 3 assumption might be too strong; have you ever canceled a subscription because of an annoying change to the UI or because of too many marketing emails? But nonetheless this is an example of how level 3 knowledge can help you identify interesting counterfactuals that you can't identify with graphs alone.

In SCM modeling, one might use an SCM that you do not believe is the true SCM, but still has important properties in common with the true SCM. One of these might be monotonicity. For example, normalizing flows are monotonic, and that makes them useful if you think the causal mechanism you are modeling with the flow indeed is monotonic. Of course, if a level 3 assumptions like monotonicity gives you identification, you don't need an SCM, you can just derive and work with the corresponding level 2 or level 1 estimand. That said, without the aid of graph-base identification algorithms, these take more work to derive.

9 Summary

- Causal identification is the procedure of discerning when causal inferences can be drawn from experimental or observational data. It is crucial in untangling complex causal issues that cannot be fully addressed with boundless data.
- The significance of identification has increased in the AI era to comprehend the causal inductive bias in model architectures. It aids in predicting failure points in reasoning tasks and contributes to the creation of enhanced architectures with reduced data necessities.
- The complex, theory-laden domain of causal inference, identification, can be streamlined using libraries like Y0 and DoWhy, which undertake the theoretical heavy-lifting.
- The causal hierarchy, or Pearl's hierarchy, is a three-tiered structure that categorizes the causal questions we pose, the models we develop, and the causal inferences we draw. These levels are Association, Intervention, and Counterfactual.
- Association-level reasoning addresses "what is" questions and attempts to model or recognize dependencies between variables without a causal relationship.
- Intervention-level reasoning handles queries like "what if we increase side-quest engagement?", whereas counterfactual reasoning deals with questions such as "if this player had low side-quest engagement and low purchases, what would their purchases be if they were more engaged?"

- A model's assumptions are spread across various levels of the causal hierarchy. Assumptions at Level 2 concern deducing the impacts of an intervention, while Level 3 assumptions require detailed information about causal relationships beyond what a causal DAG represents.
- Data types also adhere to the hierarchy. Observational data resides at the association level, interventional data at the intervention level, and counterfactual data at the counterfactual level.
- Counterfactual data arises in situations where the modeler can control a deterministic data-generating process, as seen in cloud computing resource allocation or data produced from simulators in scientific and engineering disciplines.
- Queries, characterized as probability distributions or their derivatives, fall within the hierarchy. Ordinary probability queries exist at the association level, intervention queries at the intervention level, and counterfactual queries at the counterfactual level.
- The causal hierarchy theorem asserts that the three layers of the hierarchy are "almost always separate," meaning lower-level data is insufficient to infer a distribution at a higher level. This concept is fundamental for identification, suggesting that higher-level assumptions are needed to infer a higher-level query from lower-level data.
- The Y0 Python library is utilized to check a query's identifiability from a given DAG and distribution, verifying if simulation from $P(I_{E=\text{"high"}})$ is possible using the model of P(G, E, I).
- Statistical identification, as referred to in statistics, is the capacity to infer a specific value or relationship from data. It signifies whether the estimator will converge to the actual value as the data size expands.
- Causal identification, essentially statistical identification across hierarchy levels, involves determining if data from one level can infer a query at a higher level.
- Identification pertains to whether data can infer a causal query, while estimation relates to the statistical techniques used to achieve this. An example of Level 1 to Level 2 identification is backdoor adjustment.
- The causal estimand, an operation applied to the distribution to infer the query, can be derived using algorithms and graphs. For a query $P(I_{E=\text{"high"}}=i)$, the backdoor estimand is $\Sigma_g P(I=i|E=\text{"high"}, G=g)P(G=g)$.
- Occasionally, a query may not be fully identifiable from the assumptions and distribution but might be partially identifiable, allowing for the establishment of bounds on the value being identified.
- Causal identification may depend on assumptions from different levels of the causal hierarchy. Graphical or nonparametric identification employs a Directed Acyclic Graph (DAG) and a graph-based identification algorithm.
- Back door adjustment is a prevalent identification method. For instance, in the context of online gaming, a backdoor estimand is employed to infer a level 2 query from a level

1 distribution.

- The backdoor adjustment formula, $\Sigma_z P(Y|X=x, Z=z)P(Z=z)$, involves Z, a set of variables that are common causes of both X and Y. The concept of d-separation helps identify this set.
- D-separation aids in distinguishing the causal "signal" along the causal path from the non-causal "noise" associated with additional statistical information on backdoor paths.
- Z is a set of variables that should d-separate all backdoor paths from X to Y, should not create new d-connecting paths between X and Y, and should not d-separate any causal paths from X to Y.
- An example demonstrates a DAG with three backdoor paths between X and Y. Valid adjusters for this situation are any set of nodes that will d-separate all these paths.
- Adjustment for factors is only feasible if these factors exist in the dataset. Without them, the adjustment formula cannot be applied.
- When a variable is latent, it cannot d-separate the backdoor paths. In such cases, the only valid adjustment set is {Z, V}. If more variables, such as W and U, are latent, it becomes impossible to close the backdoor paths, rendering the query unidentifiable through a backdoor estimand.
- Common backdoor adjustment mistakes involve adjusting for unnecessary variables. For instance, adjusting for job opportunities or job prestige while studying the causal effect of education on income level would be inappropriate as it would obstruct the direct causal path or d-connect a non-causal path, respectively.
- Once valid adjustment sets are pinpointed, the selection of one set usually depends on non-causal considerations. Variables' potential values, continuity, dimensionality, observability, compatibility with the estimator, and industry-standard practice may influence this choice.
- The ignorability assumption is essential in causal inference. It suggests that the causal variable and the potential outcomes are conditionally independent given confounders.
- The backdoor estimand is derived by leveraging the ignorability assumption and important statistical principles like the law of total probability, chain rule, and law of consistency.
- Identification involves formulating mathematical algorithms or formulations that can assist in understanding the relationships between variables in a dataset. Future discussions will introduce do-calculus, which offers graph-based rules for identification to aid in algorithm development.
- A case study involving Netflix's recommendation systems is presented to demonstrate the identification of counterfactual (level 3) estimands. System variables are T, B, V, W, A, and C, which represent different facets of recommendation and viewer engagement.
- The tutorial discusses how to analyze the model with Y0, beginning with the causal

effect of the "Top Picks for You" content on attrition, leading to the "Effect of treatment on the treated" (ETT) concept.

- The ETT is the counterfactual query that calculates the attrition difference for those who were exposed to a particular policy versus those who weren't. Graphical identification using ID algorithms with counterfactual graphs for binary outcome A can identify this.
- The Y0 library can derive an ETT estimand using the IDC* graphical identification algorithm. This results in a verbose level 2 estimand that can be further simplified using level 2 ID algorithms.
- The counterfactual graph, created by identifying which nodes across worlds in the parallel world graph are the same and then combining those nodes, contains nodes across parallel worlds that are relevant to the events in the query.
- Y0's ID algorithms employ the counterfactual graph to execute graph-based counterfactual inference by applying
- do-calculus-based reasoning. It identifies a level 2 estimand for a level 3 query. The level 2 terms in the estimand can be answered with experimental data or be broken down to level 1 estimands.
- The counterfactual graph is versatile, supporting complex queries. For instance, it can isolate how one variable impacts an outcome from how another variable affects the same outcome. This is demonstrated using data from an experiment where the "Because You Watched" policy B was randomized. The query's expansion creates three parallel worlds to reason over: the actual, the world with do(T=+t), and the world with do(B=-b). The query collapses the parallel world graph to the same counterfactual graph as $P(A_{T=-t}=+a|T=+t)$.
- Single-world intervention graphs (SWIGs) provide an alternative to counterfactual identification with counterfactual graphs. They are constructed using the original causal DAG and the causal query, incorporating operations like node-splitting and subscript inheritance for each intervention.
- SWIGs contain counterfactual variables and allow for d-separation, enabling identification.
- The "ignorability trick" can facilitate identification within SWIGs.
- Node-splitting in SWIGs necessitates a "single world" assumption, which assumes it's possible to know a variable's natural value while also intervening on it before it realizes that value.
- Both SWIGs and Counterfactual graphs are useful for identifying counterfactuals, yet they differ considerably. SWIGs work with variables and a narrow set of counterfactuals under the single-world assumption, while counterfactual graphs can accommodate queries that cannot be graphically identified.
- Pyro defaults to SWIGs for interventions, as it enables conditioning and intervention on the same variable without creating parallel worlds.

- Identifying counterfactuals from level 1 or 2 distributions requires level 3 assumptions. These are causal assumptions that cannot be represented with a straightforward causal DAG.
- Level 3 assumptions reduce the class of possible SCMs. Common examples include the assumption that the SCM belongs to a canonical class, such as linear additive SCMs.
- Considering the binary nature of an outcome as a level 3 assumption enables graphical identification by reducing counterfactual possibilities.
- SWIG-based identification of the ETT relies on the single-world assumption, a level 3 assumption. This allows identification without needing binary outcomes.