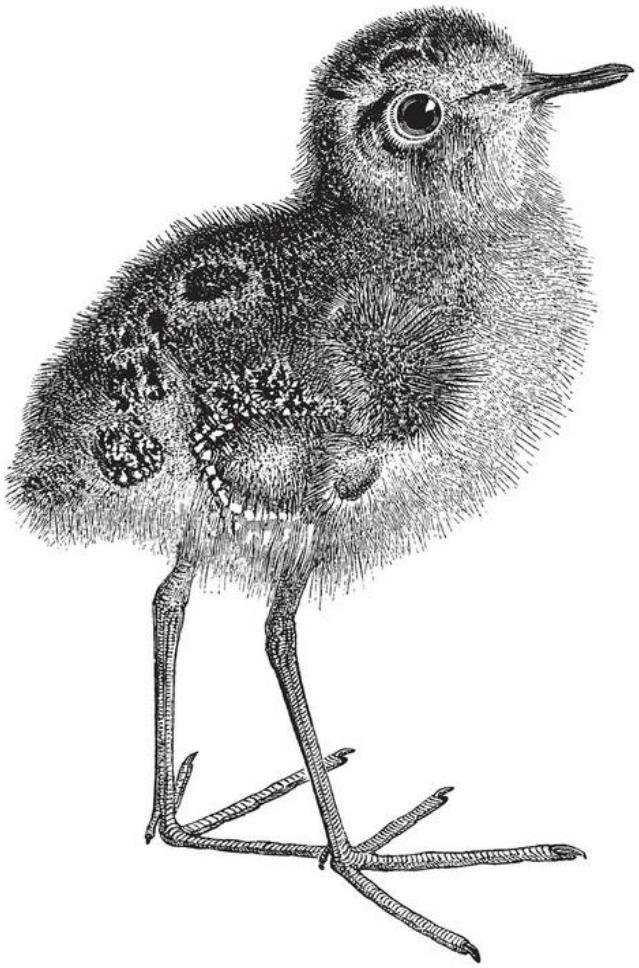
O'REILLY®

Causal Inference in Python

Applying Causal Inference in the Tech Industry



Early Release

RAW & UNEDITED

Matheus Facure

Causal Inference in Python

Applying Causal Inference in the Tech Industry

Matheus Facurer

Beijing • Boston • Farnham • Sebastopol • Tokyo

Causal Inference in Python

by Matheus Facure

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (http://oreilly.com). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editors: Virginia Wilson and Nicole Butterfield

Production Editor: Katherine Tozer

Interior Designer: David Futato

Cover Designer: Karen Montgomery

Illustrator: Kate Dullea

November 2023: First Edition

Revision History for the Early Release

- 2022-11-30: First Release
- 2023-02-08: Second Release

See http://oreilly.com/catalog/errata.csp?isbn=9781098140250 for release details.

978-1-098-14025-0

Preface

A NOTE FOR EARLY RELEASE READERS

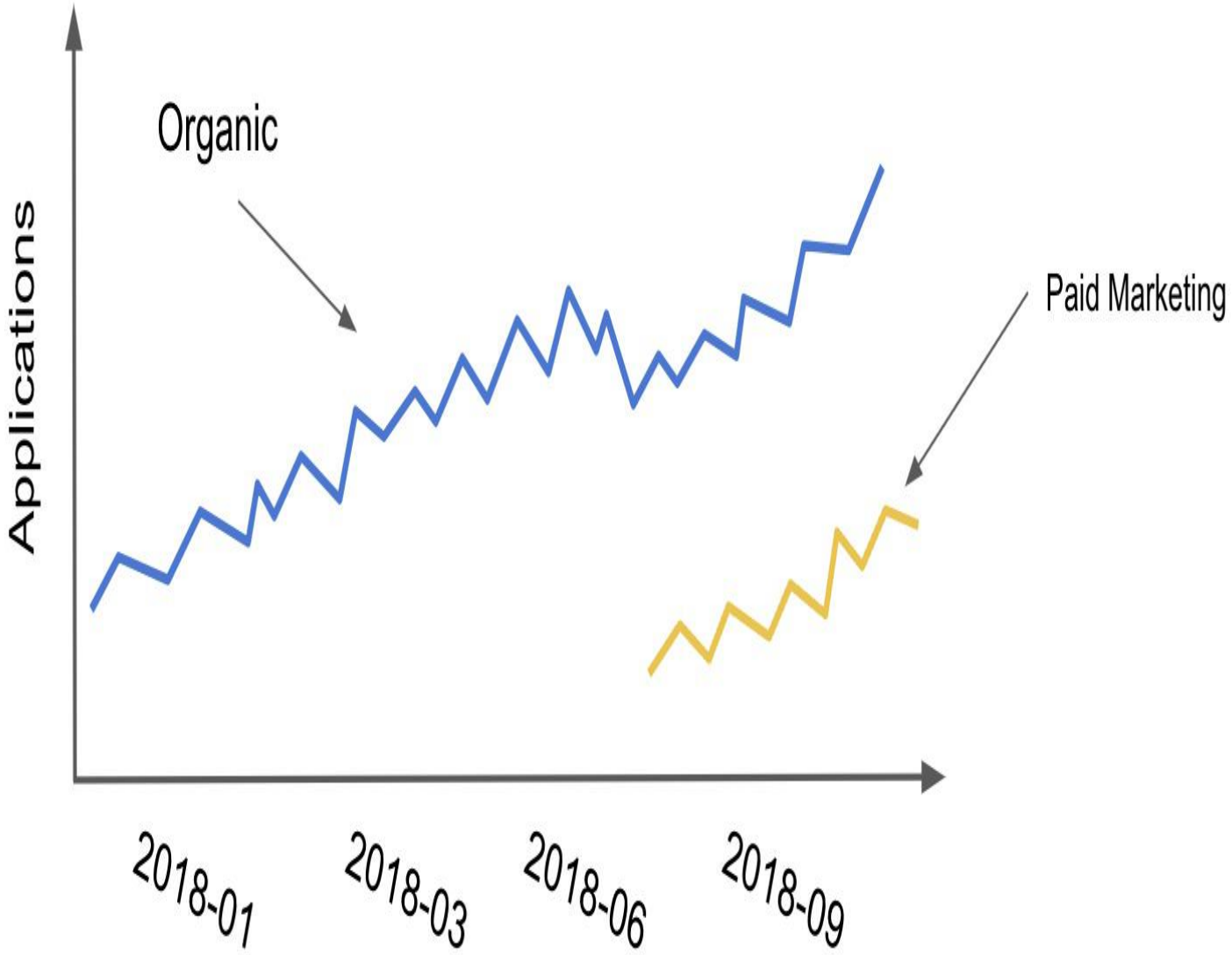
With Early Release ebooks, you get books in their earliest form—the author's raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the Preface of the final book. Please note that the GitHub repo will be made active later on.

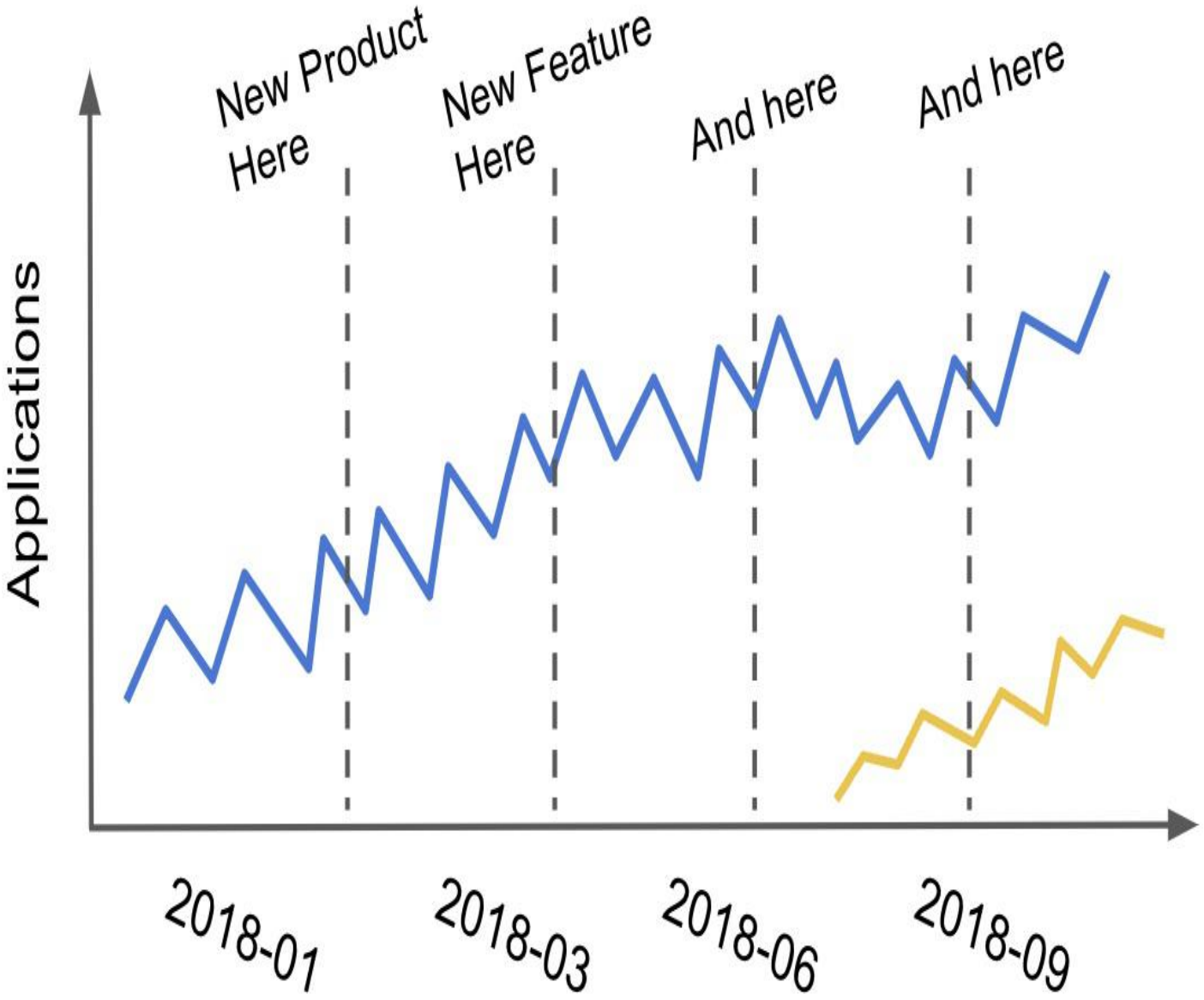
If you have comments about how we might improve the content and/or examples in this book, or if you notice missing material within this chapter, please reach out to the editor at *vwilson@oreilly*.

Picture yourself as a young data scientist who's just starting out in a fast growing and promising startup. Although you haven't mastered it, you feel pretty confident about your machine learning skills. You've completed dozens of online courses on the subject and even got a few good ranks in prediction competitions. You are now ready to apply all that knowledge to the real world and you can't wait for it. Life is good.

Then, your team lead comes with a graph which looks something like this

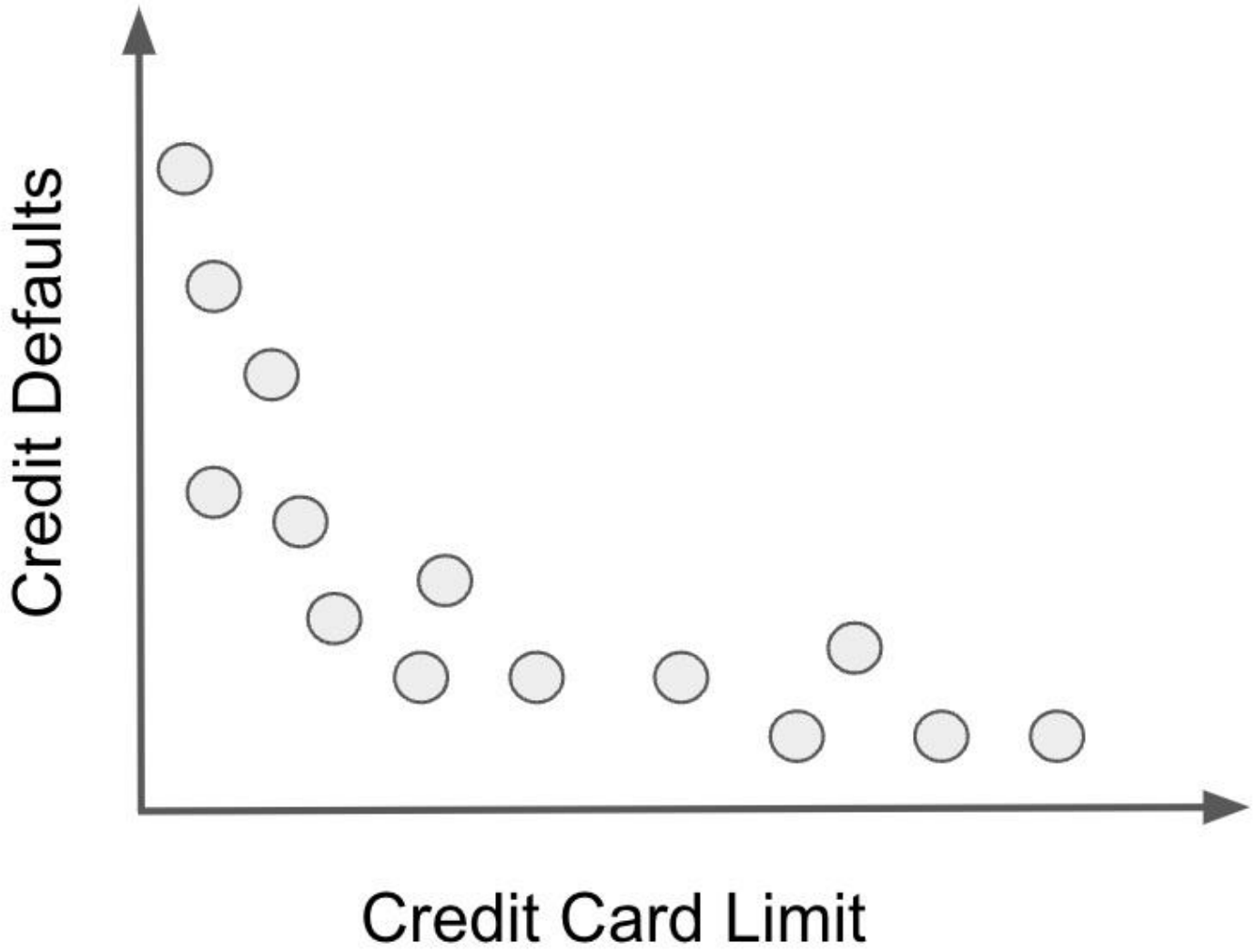


and an accompanying question: “Hey, we want you to figure out how many additional customers paid marketing is really bringing us. When we turned it on, we definitely saw some customers coming from the paid marketing channel, but it looks like we also had a drop in organic applications. We think **some of the customers from paid marketing would have come to us even without paid marketing.**”. Well…, you were expecting a challenge, but this?! How could you know what would have happened without paid marketing? I guess you could compare the total number of applications, organic and paid, before and after turning on the marketing campaigns, but in a fast growing and dynamic company, how would you know that nothing else changes when they turn on the campaign?



Changing gears a bit (or not at all), place yourself in the shoes of a brilian risk analyst. You were just hired by a lending company and your first task is to perfect their credit risk model. The goal is to have a good automated decision making system which reads customer data and decides if they are credit worthy (underwrites them) and how much credit the company can lend them. Needless to say that errors in this system are incredibly expensive, especially if the given credit line is high.

A key component of this automated decision making is understanding the impact more credit lines have on the likelihood of customers defaulting. Can they all manage a huge chunk of credit and pay it back or will they go down a spiral of overspending and crippling debt? In order to model this behavior, you start by plotting credit average default rates by given credit lines. To your surprise, the data displays this unexpected pattern



The relationship between credit and defaults seems to be negative. How come giving more credit results in lower chances of defaults? Rightfully suspicious, you go talk to other analysts to understand this. Turns out the answer is very simple: to no one's surprise, the lending company gives more credit to customers that have lower chances of defaulting. So, it is not the case that high lines reduce default risk, but rather, the other way around. Lower risk increases the credit lines. That explains it, but you still haven't solved the initial problem: how to model the relationship between credit risk and credit lines with this data? Surely you don't want your system to think more lines implies lower chances of default. Also, naively randomizing lines in an A/B test just to see what happens is pretty much off the table, due to the high cost of wrong credit decisions.

What both of these problems have in common is that you need to know the impact of changing something which you can control (marketing budget and credit limit) on some business outcome you wish to influence (customer applications and default risk). Impact or effect estimation has been the pillar of modern science for centuries, but only recently have we made huge progress in systematizing the tools of this trade into the field which is coming to be known as Causal Inference. Additionally, advancements in machine learning and a general desire to automate and inform decision making processes with data has brought causal inference into the industry and public institutions. Still, the causal inference toolkit is not yet widely known by decision makers nor data scientists.

Hoping to change that, I wrote Causal Inference for the Brave and True, an online book which covers the traditional tools and recent developments from causal inference, all with open source Python

software, in a rigorous, yet light-hearted way. Now, I'm taking that one step further, reviewing all that content from an industry perspective, with updated examples and, hopefully, more intuitive explanations. My goal is for this book to be a startpoint for whatever questions you have about decisions making with data.

Prerequisites

This book is an introduction to Causal Inference in Python, but it is not an introductory book in general. It's introductory because I'll focus on application, rather than rigorous proofs and theorems of causal inference; additionally, when forced to choose, I'll opt for a simpler and intuitive explanation, rather than a complete and complex one.

It is not introductory in general because I'll assume some prior knowledge about machine learning, statistics and programming in Python. It is not too advanced either, but I will be throwing in some terms that you should know beforehand.

For example, here is a piece of text that might appear.

> *The first thing you have to deal with is the fact that continuous variables have* $P(T=t)=0$ *everywhere. That's because the probability is the area under the density and the area of a single point is always zero. A way around this is to use the conditional density* $f(T|X)$*, instead of the conditional probability* $P(T=t|X)$

I won't provide much explanation on what a density is and why it is different from a probability. Here is another example, this time about machine learning.

> *Alternatively, you can use machine learning models to estimate the propensity score. But you have to be more careful. First, you must ensure that your ML model outputs a calibrated probability prediction. Second, you need to use out-of-fold predictions to avoid bias due to overfitting.*

Here, I won't explain what a machine learning model is, not what it means for it to have calibrated predictions, what is overfitting and out-of-fold prediction.

Here is a non exhaustive list of the things I recommend you know before reading this book:

- Basic knowledge of Python, including the most commonly used data scientists libraries: Pandas, Numpy, Matplotlib, Scikit-Learn. I come from an Economics background, so you don't have to worry about me using very fancy code. Just make sure you know the basics pretty well.
- Knowledge of basic statistical concepts like distributions, probability, hypothesis testing, regression, noise, expected values, standard deviation, independence. I will include a statistical review in the book in case you need a refresher.
- Knowledge of basic data science concepts, like machine learning model, cross validation, overfitting and some of the most used machine learning models (gradient boosting, decision trees, linear regression, logistic regression).

The main audience of this book is Data Scientists who are working in the industry. If you fit this

description, there is a pretty good chance that you cover the prerequisites that I've mentioned. Also, keep in mind that this is a broad audience, with very diverse skill sets. For this reason, I might include some note or paragraph which is meant for the most advanced reader. So don't worry if you don't understand every single line in this book. You'll still be able to extract a lot from it. And maybe it will come back for a second read once you mastered some of its basics.

Outline

Part I will cover the basics concepts about causal inference. Chapter 1 will introduce the key concepts from causal inference as you use them to reason about the effect of cutting prices on sales. Chapter 2 will talk about the importance of A/B testing (or randomized control trial) not only as an instrument for decision making, but as the gold standard you will use to benchmark the other causal inferences tools. This will also be a great opportunity to review some statistical concepts. Chapter 3 will be mostly theoretical, covering causal identification and graphical models, a powerful method for (literally) drawing our assumptions about the causal process and reasoning about what you need to do in order to untangle association from causation. After finishing part I, you should have the basic foundation to reason in terms of causal inference.

In Part II you'll be introduced to two of the workhorses for untangling causation from correlation: linear regression and propensity weighting. Chapter 4 will cover linear regression, but not from a perspective that most data scientists are familiar with. Rather, you'll learn about an important bias removal technique: orthogonalization. Chapter 5 will cover Propensity Score and Doubly Robust estimation.

Part III takes what you saw in part II and adds machine learning and big data to the mix. You'll look into causal inference as a tool for personalized decision making. Through the eyes of a food delivery service, you'll try to understand which customers should be given discount coupons to capture their loyalty and which customers don't need that extra incentive. You will enter the world of heterogeneous treatment effects and learn some of the recent developments in the intersection between machine learning and causal inference. This part covers methods like the T, X and S-learners and Double/Debiased Machine Learning.

Part IV adds a time dimension to causal inference. You will learn how to leverage the fact that you can observe the same customers or markets across multiple time periods. You'll use this sort of data to understand how you can uncover the true impact of paid marketing even without being able to randomize who gets to see your advertisements.

Chapter 1. Introduction To Causal Inference

A NOTE FOR EARLY RELEASE READERS

With Early Release ebooks, you get books in their earliest form—the author's raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the 1st chapter of the final book. Please note that the GitHub repo will be made active later on.

If you have comments about how we might improve the content and/or examples in this book, or if you notice missing material within this chapter, please reach out to the editor at *vwilson@oreilly*.

In this first chapter I'll introduce a lot of the fundamental concepts of causal inference as well as the main reason behind it. You will learn a lot of jargon that will be used in the rest of the book. Also, I want you to always keep in mind why you need causal inference and what you can do with it. This chapter will not be about coding, but about very important first principals of causal inference.

What is Causal Inference

Causality is something you might know as a dangerous epistemological terrain you must avoid going into. Your stats teacher might have said over and over again that "association is not causation" and that confusing the two would cast you to academic ostracism or, at the very least, be severely frowned upon. But you see, that is the thing: **sometimes, association is causation**.

We humans know this all too well, since, apparently, we've been primed to take association for causation. When you decide not to drink that fourth glass of wine, you correctly inferred that it will mess you up in the next day. You are drawing from past experience. From nights where you drank to much and woke up with a headache; from nights you took just a glass of wine, or not at all, and nothing happened. You've learned that there is something more to the association between drinking and hangover. You've inferred causality out of it.

On the flip side, there is some truth to your stats teacher warnings. Causation is a slippery thing. When I was a kid, I ate calamari doré twice and on both times it ended terribly, which lead me to conclude I was allergic to squid (and clam, octopus or any type of sea invertebrate). It took me more than 20 years to try it again. When I did, it was not only delicious, but it also caused me no harm. In this case, I've confused correlation with causation. This was a harmless confusion, as it only deprived me of delicious seafood for some years, but mistaking correlation for causation can have much darker consequences. If you invest in the stock market, you've probably been through a situation where you decided to put money in just before a steep increase in prices, or to withdrawn just before everything collapsed. This likely tempted you to think you could time the market. If you managed to ignore that temptation, good for you. But many fall for it, thinking that their intuition is causally linked to the erratic movements of

stocks. In some situations, this belief leads to riskier and riskier bets until, eventually, close everything is lost.

In a nutshell, association is when two quantities or random variables move together, while causality is one change in one variable causes change in another. It's easy to see why one does not imply the other. The number of Nobel prize a country has is associated with per capita consumption of chocolate. These variable move together, but it would be foolish to think one causes the other. But equating the two is a hole different matter. **Causal inference is the science of inferring causation from correlation and understanding why they differ**.

Why we Do Causal Inference

Causal inference can be done for the sole purpose of understanding reality. But there is often a normative component to it. The reason you've inferred that too much drinking causes headaches is because you want to change your drinking habits to avoid the pain. The company you work for wants to know if marketing costs cause growth in revenues because if it does, they can use it to increase profits. Generally speaking, you want to know cause and effect relationships so that you can intervene on the cause to bring upon a desired effect. If you take causal inference to the industry, it becomes mostly a branch of the decision making sciences.

Since this book is mostly industry focused, it will cover the part of causal inference that is preoccupied with understanding the impact of interventions. What would happen if you used another price instead of this price you're currently asking for your merchandise? What would happen if you switch from this low-sugar diet your on to that low-fat diet? What will happen to the bank's margins if it increases the customers' credit line? Should the government give tablets to every kid in school to boost their reading test score or should it build an old-fashioned library? Is marrying good for your personal finances or are married couples wealthier just because wealthy people are more likely to attract a partner in the first place? These questions are all practical. They stem from a desire to change something, in your business or in your life, so that you can be better off.

Machine Learning and Causal Inference

If you take a deeper look at the types of questions you want to answer with causal inference, you will see they are mostly of the "what if" type. I'm sorry to be the one that says it, but Machine Learning (ML) is just awful at those types of questions.

ML is very good at answering prediction questions. As Ajay Agrawal, Joshua Gans, and Avi Goldfarb put it in the book Prediction Machines, "the new wave of artificial intelligence does not actually bring us intelligence but instead a critical component of intelligence - prediction". You can do all sorts of beautiful things with machine learning. The only requirement is to frame your problems as prediction ones. Want to translate from English to Portuguese? Then build an ML model that predicts Portuguese sentences when given English sentences. Want to recognize faces? Then create an ML model that predicts the presence of a face in a subsection of a picture.

However, ML is not a panacea. It can perform wonders under rigid boundaries and still fail miserably if

its data deviates a little from what the model is accustomed to. To give another example from Prediction Machines, “in many industries, low prices are associated with low sales. For example, in the hotel industry, prices are low outside the tourist season, and prices are high when demand is highest and hotels are full. Given that data, a naive prediction might suggest that increasing the price would lead to more rooms sold.”

Machine learning uses correlations between variables to predict one from the other. This means it will work incredibly well as long as you don’t change the variables it is using to make predictions. This completely defeats the purpose of using predictive ML for most decision making which involves interventions.

The fact that most Data Scientists know a lot of ML but not much about causal inference leads to an abundance of ML models being deployed where they are not useful for the task at hand. One of the main goals of companies is to **increase** sales or usage. Yet, a ML model that just predicts sales is oftentimes useless - if not harmful - for this purpose. This model might even conclude something nonsense, as in the example where high volumes of sales are associated with high prices. Yet, you will be surprised at how many companies implement ML models to simply predict what the company is actually trying to influence.

This does not mean that ML is completely useless for causal inference. It just means that, when naively applied, it often does more harm than good. But if you approach ML from a different angle, as a toolbox of powerful models, rather than purely predictive machines, you’ll start to see how they can connect to the goals of causal inference. In part III of book, I’ll show what you need to watch out for when mixing ML and Causal Inference and how to repurpose common ML algorithms, like decision trees and gradient boosting, to do causal inference.

Association and Causation

Intuitively, you kind of know why association is not causation. If someone tells you that top notch consulting causes your business to improve, you are bound to raise an eyebrow. How can you know if the consulting firm is actually causing business to improve or is it just that only flourishing businesses have the luxury to hire those services?

To make things a more tangible, put yourself in the shoes of an online marketplace company. Small and medium sized businesses use your online platform to advertise and sell their products. These businesses have complete autonomy in stuff like setting prices and when to do sales. But it is in the best interest of your company that these businesses flourish and prosper. So, you decided to help them by giving guidance on how, if and when to set up a sales campaign where they announce a temporary price drop to consumers. To do that, the first thing you need to know is the **impact of lowering prices on units sold**. If the gains from selling more compensate for the loss of selling cheaper, sales will be a good idea. If you hadn’t already noticed, this is a causal question. You need to answer how many additional units a business would have sold had they lowered prices, compared to not doing anything.

Needless to say, this is a complicated question. Maybe too complicated for the beginning of this book. Different businesses operate within your platform. Some sell food; some, clothes. Some sell fertilizers and agricultural products. This means that price cuts might have different impacts, depending on the

type of business. For instance, it might be a good idea for a clothing business to announce lower prices one week prior to father's day. Yet, a similar price drop for an agribusiness will probably do very little. So, let's simplify the problem a bit. Let's focus your attention only on one type of business: those that sell kid's toys. Also, let's focus your attention on one period of the year: December, before Christmas. For now, you'll just try to answer how cutting prices during these periods increases sales so you can pass this information along to the businesses operating in the kids toy industry, allowing them to make better decisions.

To answer these question, you can leverage data from multiple kid's toys businesses. This data comes stored in a Pandas DataFrame, with 5 columns.

	store	weeks_to_xmas	avg_week_sales	is_on_sale	weekl
0	1	3	12.98	1	219.60
1	1	2	12.98	1	184.70
2	1	1	12.98	1	145.75
3	1	0	12.98	0	102.45
4	2	3	19.92	0	103.22
5	2	2	19.92	0	53.73

The first column is the store's unique identifier (ID). You have weekly data for the each store in the month of December. You also got information about the size of each business in terms of average products sold per week during that year. A boolean column (0 or 1) flags the business as being on sale at the time. The last column shows the average weekly sales of that store during that week.

The Treatment and the Outcome

Now that you have some data to look at, it's time to learn our first bit of technicality. Let's call T_i the treatment intake for unit i.

$$T_i = \begin{cases} 1 \text{ if unit i received the treatment} \\ 0 \text{ otherwise} \end{cases}$$

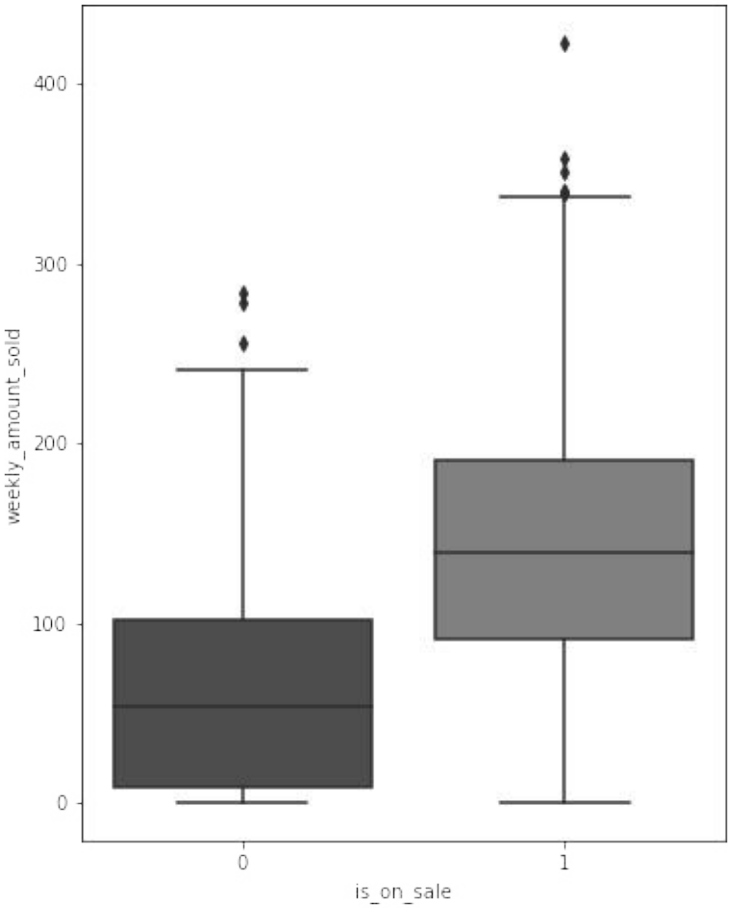
The treatment here doesn't need to be medicine or anything from the medical field. Instead, it is just a term I'll use to denote some intervention for which I want to know the effect of. In this case, the treatment is simply a price drop for one of the businesses inside your online platform, represented by the column `is_on_sale`.

> **NOTE**
>
> In some texts and latter on in this book, you might sometimes see D instead of T to denote the treatment. D will avoid many confusions when you have a time dimension to your causal problems.

Additionally, I'll be referring to the variable that I want to influence (`weekly_amount_sold` here) as the **outcome**. I'll represent the outcome for unit i with Y_i. With these two new concepts, I can restate the goal of causal inference as the process of learning the impact T has on Y. In our example, this amounts to figuring out the effect of `is_on_sale` on `weekly_amount_sold`.

The Fundamental Problem of Causal Inference

Here is where things get interesting. The **fundamental problem of causal inference** is that you can never observe the same unit with and without treatment. It is as if you have two diverging roads and can only know what lies ahead of the one taken. To fully appreciate this issue, let's go back to our example and plot the outcome by the treatment, that is, `weekly_amount_sold` by `is_on_sale`. You can immediately see that those days and stores that dropped their price sell a lot more.



This also matches our intuition about how the world works: people buy more when prices are low and sales (usually) means lower prices. This is very nice, as causal inference goes hand in hand with expert knowledge. But you shouldn't be too careless. It is probably the case that giving and advertising discounts will make customers buy more. But this much more? From the plot above, it looks like the amount sold is about 150 units higher, on average, when a sale is going on than otherwise. This sounds suspiciously high, since the range of units solds when there is no sale is about 0 to 50. If you scratch

your brains a bit, you can start to see what you might be mistaking correlation for causation. Maybe it is the case that only larger businesses, which are the ones that sell the most anyway, can afford to do aggressive price drops. Maybe businesses do sales closer to Christmas and that's when customers buy the most anyway.

The point is, you would only be certain about the true effect of price cuts on units sold if you could observe the same business, at the same time, with and without sales going on. Only if you compare these two counterfactual situations will you be sure about the effect of price drops. However, as discussed earlier, the fundamental problem of causal inference is that you simply can't do that. Instead, you'll need to come up with something else.

Causal Models

You can reason about all these problems intuitively, but if you want to go beyond simple intuition, I need to give you some formal notation. This will be our everyday language to speak about causality. Think of it as the common tongue we will use with fellow practitioners of the arts of causal inference.

A **causal model** is a series of assignment mechanisms denoted by $\leftarrow$. In these mechanisms, I'll use u to denote variables outside the model. Everything else are variables I care very much about and are hence included in the model. Finally, there are functions f that map one variable to another. Take the following causal model as an example

$$T \leftarrow f_t\left(u_t\right)$$
$$Y \leftarrow f_y\left(T, u_y\right)$$

With the first equation, I'm saying that u_t, a set of variables I'm not explicitly modeling (also called exogenous), causes the treatment T via the function f_t. Next, T alongside another set of variables u_y (which I'm also choosing not to model) jointly causes the outcome Y via the function f_y. u_y is in this last equation to say that the outcome is not determined by the treatment alone. Some other variables also play a role in it, even if I'm choosing not to model them. In this example, it would mean that `weekly_ammount_sold` is caused by the treatment `is_on_sale` and other factors I won't specify. The point of u is to account for all the variation in the variables caused by it that are not already accounted for with the endogenous variables. In our example, I could be saying that price drops are caused by factors - could be business size, could be something else - that are not inside the model.

$$IsOnSales \leftarrow f_t\left(u_t\right)$$
$$AmountSold \leftarrow f_y\left(IsOnSales, u_y\right)$$

I'm using $\leftarrow$ instead of $=$ to explicitly state the non reversibility of causality. With the equality sign, $Y = T + X$ is equivalent to $T = Y + X$, but I don't want to say that T causing Y is equivalent to T causing Y. Having said that, I'll often refrain from using $\leftarrow$ just because it's a bit cumbersome. Just keep in mind that, due to non reversibility of cause and effects, unlike with traditional algebra, you can't simply throw things around the equal sign when dealing with causal models.

If you want to explicitly model more variables, you can take them out of u and account for them in the

model. For example, remember how I said that the large difference you are seeing between price cuts and no price cuts could be because larger businesses can engage in more aggressive sales? In the model above, $BusinessSize$ is not explicitly included in the model. Instead, its impact gets relegated to the side, with everything else in u. But I could model it explicitly.

$$\begin{aligned} BusinessSize &\leftarrow f_t\left(u_s\right) \\ IsOnSales &\leftarrow f_t\left(BusinessSize, u_t\right) \\ AmountSold &\leftarrow f_y\left(IsOnSales, BusinessSize, u_y\right) \end{aligned}$$

To include this extra endogenous variable, first, I'm adding another equation to represent how that variable came to be. Next, I'm taking $BusinessSize$ out of u_t. That is, I'm no longer treating it as a variable outside the model. I'm explicitly saying that $BusinessSize$ causes $IsOnSales$ (along with some other external factors which I'm still choosing not to model). This is just a formal way of encoding our beliefs that bigger businesses are more likely to cut prices. Finally, I can also add $BusinessSize$ to the last equation. This encodes our belief that bigger businesses also sell more. In other words, $BusinessSize$ is a common cause to both the treatment $IsOnSales$ and the outcome $AmountSold$.

Since this way of modeling is probably new to you, it's useful to link it to something perhaps more familiar. If you come from economics or statistics, you might be used to another way of modeling the same problem:

$$AmountSold_i = \alpha + \beta_1 IsOnSales_i + \beta_2 BusinessSize_i + e_i$$

It looks very different at first, but closer inspection will reveal how the model above is very similar to the one you saw earlier. First, notice how it is just replacing the final equation in that previous model and opening up the f_y function, stating explicitly that endogenous variables $IsOnSales$ and $BusinessSize$ are linearly and additively combined to form the outcome $AmountSold$. In this sense, this **linear model** assumes more than the one you saw earlier. You can say that it imposes a functional form to how the variables relate to each other. Second, you are not saying anything about how the independent (endogenous) variables - $IsOnSales$ and $BusinessSize$ - come to be. Finally, this model uses the equals sign, instead of the assignment operator, but we already agreed not to stress too much about that.

Interventions

The reason I'm taking my time to talk about causal models is because, once you have one, you can start to tinker with it in the hopes of answering a causal question. The formal term for this is **intervention**. For example, you could take that very simple causal model and force everyone to take the treatment t_0. This will eliminate the natural causes of T, replacing them by a single constant.

$$\begin{aligned} T &\leftarrow t_0 \\ Y &\leftarrow f_y\left(T, u_y\right) \end{aligned}$$

Notice that this is done as a thought experiment to answer the question “what would happen to the outcome Y if I where to set the treatment to t_0”. you don’t actually have to intervene on the treatment (although you could and will, but later). In the causal inference literature, you can refer to these interventions with a $do(.)$ operator. If you want to reason about what would happen if you intervene on T, you could write $do(T = t_0)$.

EXPECTATIONS

I’ll use a lot of Expectations and Conditional Expectations notation from now on. You can think about expectations as the population value that the average is trying to estimate. $E[X]$ denotes the (marginal) expected values of the random variable X. It can be approximate by the sample average of X. $E[Y|X = x]$ denotes the expected value of Y when $X = x$. This can be approximated by the average of Y when $X = x$.

The $do(.)$ operator also allows you to have a first glance at why association is different from causation. I already argued how high sales volume for business on sale, $E[AmountSold|IsOnSales = 1]$, could be overestimating the average sales volume a business would have, had it engaged in a price cut, $E[AmountSold|do(IsOnSales = 1)]$. In the first case, you are looking at businesses that chose to cut prices, which are probably bigger businesses. In contrast, the latter quantity, $E[AmountSold|do(IsOnSales = 1)]$, refers to what would’ve happened if you forced every business to engage in sales, not just the big ones. Importantly, in general,

$$E[AmountSold|IsOnSales = 1] \neq E[AmountSold|do(IsOnSales = 1)]$$

One way to think about the difference between the two is in terms of selection and intervention. When you condition on sales, you are measuring the amount sold on a selected subsample where all the business did sales. When you condition on the intervention $do(isOnSales)$, you are forcing every business to do sales and measuring the amount sold on the entire sample.

$do(.)$ is used to define causal quantities which are not always recoverable from observed data. In the example above, we can't observe $do(IsOnSales = 1)$ for every business, since we didn't forced them to do sales. $do(.)$ is most useful as a theoretical concept which you can use to explicitly state the causal quantity you are after.

Individual Treatment Effect

For starters, the $do(.)$ operator also allows you to express the **individual treatment effect**, or the impact of the treatment on the outcome for a individual unit i. We can write it as the difference between two interventions.

$$\tau_i = Y_i \left| do\left(T = t_1\right) - Y_i \right| do\left(T = t_0\right)$$

In words, you would read this as "the effect, τ_i, of going from treatment t_0 to t_1 for unit i is the difference in the outcome of that unit under t_1 compared to t_0". We could use this to reason about our problem of figuring out the effect of flipping $IsOnSales$ from 0 to 1 in $AmountSold$

$$\tau_i = AmountSold_i \left| do\left(IsOnSales = 1\right) - AmountSold_i \right| do\left(IsOnSales = 0\right)$$

Notice that, due to the fundamental problem of causal inference, you can only observe one term of the equation above. So even though you can theoretically express that quantity, it doesn't necessarily means you can recover it from data.

Potential Outcomes

The other thing you can define with the $do(.)$ operator is perhaps the coolest and most widely used concept in causal inference: counterfactual or **potential outcomes**:

$$Y_{ti} = Y_i | do\left(T_i = t\right)$$

You should read this as "unit i's outcome would be Y_t if its treatment is set to t". Sometimes, I'll use function notation to define potential outcomes, since subscripts can quickly become too crowded

$$Y_{ti} = Y(t)_i$$

When talking about a binary treatment (treated or not treated), I'll denote Y_{0i} as the potential outcome for unit i without the treatment and Y_{1i} as the potential outcome for **the same** unit i with the treatment. I'll also refer to one potential outcome as factual, meaning I can observe it, and the other one as counterfactual, meaning it cannot be observed. For example, if unit i is treated, I get to see what happens to it under the treatment, that is, I get to to see Y_{1i}, which I'll the factual potential outcome. In contrast, I can't see what would happen if, instead, unit i wasn't treated. That is, I can't see Y_{0i}, since it is counterfactual.

$$Y_i = \begin{cases} Y_{i1} \text{ if unit i received the treatment} \\ Y_{i0} \text{ otherwise} \end{cases}$$

You might also find the same thing written as follows

$$Y_i = T_i Y_{i1} + \left(1 - T_i\right) Y_{i0} = Y_{i0} + \left(Y_{i0} - Y_{i0}\right) T_i$$

Back to our example, we can write $AmountSold_{i0}$ to denote the amount business i would have sold had it not done any price cut and $AmountSold_{i1}$, the amount it would have sold had it done sales. We can also define the effect in terms of these potential outcomes.

$$\tau_i = Y_{1i} - Y_{0i}$$

ASSUMPTIONS

Throughout this book, you'll see that causal inference is always accompanied by assumptions. Assumptions are statements you make when expressing a belief about how the data was generated. The catch is that they usually can't be verified with the data, that's why you need to assume them. Assumptions are not always easy to spot, so I'll do my best to make them transparent.

Consistency and No Interference Assumptions

In The equations above, there are two hidden assumptions. First, they say that the potential outcome is consistent with the treatment: $Y_i(t) = Y$ when $T_i = t$. This means there are no hidden multiple versions of the treatment beyond the ones specified with T. This assumption can be violated if the treatment comes in multiple dosages, but you are only accounting for two of them. For example, if you care about the effect of discount coupons on sales and you treat it as being binary - customers received a coupon or not - but in reality you tried multiple discount values.

A second assumption that is implied is that of no interference, or stable unit of treatment value (SUTVA). This means that the effect of one unit is not influenced by the treatment in other units: $Y_i(T_i) = Y_i(T_1, T_2, \ldots, T_i, \ldots, T_n)$. This assumption can be violated if there are spillovers or network effects. For example, if you want to know the effect of vaccines on preventing a contagious illness, vaccinating one person will make other people close to her less likely to catch this illness, even if they themselves did not get the treatment. Violations of this assumption usually cause us to think that the effect is lower than it is. With spillover, control units get some treatment effect, which in turn causes treatment and control to differ less sharply than if there was no interference.

Causal Quantities of Interest

Of course, the fundamental problem of causal inference still holds. **You can never know the individual treatment effect because you only observe one of the potential outcomes**. But not all is lost. Now that you have all these new concepts, you are ready to make some progress in working around our fundamental problem. Even though you can never know the individual effects, τ_i, there are other interesting causal quantities that you can learn about. For instance, define *average treatment effect* (ATE) as follows:

$$ATE = E[\tau_i] = \frac{1}{N}\sum_{i=0}^{N} \tau_i$$

or

$$ATE = E[Y_{1i} - Y_{0i}] = \frac{1}{N}\sum_{i=0}^{N} Y_{1i} - Y_{0i}$$

Or even

$$ATE = E[Y|do(T = 1)] - E[Y|do(T = 0)]$$

The average treatment effect represents the impact the treatment T would have on average. Some units will be more impacted by it, some, less, and you can never know the individual impact of a unit. But, as you'll soon see, you can use data to estimate the ATE or any other group effect.

Another group effect of interest is the **average treatment effect on the treated:**

$$ATT = E[Y_{1i} - Y_{0i}|T = 1]$$

This is the impact of the treatment on the units that got the treatment. For example, if you did an offline marketing campaign in a city and you want to know how many extra customers this campaign brought you in that city, this would be the ATT: the effect of marking on the city that got the campaign.

Finally, you have conditional averages treatment effects ($CATE$):

$$CATE = E[Y_{1i} - Y_{0i}|X = x]$$

Which is the effect in a group defined by the variables X. For example, you might want to know the effect of sending an email on customers that are older than 45 years and on those that are younger than that. Conditional Average Treatment effect is invaluable for personalization, since it allows us to know which type of unit responds better to an intervention.

You can also define the quantities above when the treatment is continuous. In this case, you replace the difference with a partial derivative.

$$\frac{\partial}{\partial t} E\left[Y_i\right]$$

This might seem fancy, but it's just a way to say how much you expect $E[Y_i]$ to change given a small increase in the treatment.

Causal Quantities: An Example

Let's see how you can define these quantities in our business problem. First, notice that you can never know the effect price cuts (doing sales) have on an individual business, as that would require you to see both potential outcomes, $AmountSold_{i0}$ and $AmountSold_{i1}$, at the same time. But you could, instead, focus your attention on something that is possible to estimate, like the average impact of price cuts on amount sold, $ATE = E[AmountSold_{i1} - AmountSold_{i0}]$, how the business that engaged in price cuts increased their sales,
$ATT = E[AmountSold_{i1} - AmountSold_{i0}|IsOnSales = 1]$ or even the the impact of doing sales on the week of Christmas,
$CATE = E[AmountSold_{i1} - AmountSold_{i0}|weeksToXmas = 0]$.

Now, I know you can't see both potential outcomes, but just for the sake of the argument and to make things a lot more tangible, let's suppose you could. Pretend for a moment that the causal inference deity is pleased with the many statistical battles you fought and has rewarded you with godlike powers to see the potential alternative universes, one where each outcome is realized. With that power, say you collect data on 6 businesses, 3 of which were on sale and 3 of which weren't. Remember that being on sale is the treatment and amount sold, the outcome. Let's also say that, for two of these businesses, you gathered data one week prior to Christmas, which is denoted by $x = 1$, while the other observations are the same week as Christmas.

	i	y0	y1	t	x
0	1	200	220	0	0
1	2	120	140	0	0
2	3	300	400	0	1
3	4	450	500	1	0
4	5	600	600	1	0
5	6	600	800	1	1

With your godly powers, you can see both $AmountSold_0$ and $AmountSold_1$. This makes calculating all the causal quantities we've discussed earlier incredibly easy. For instance, the ATE here would be the mean of the last column, that is, of the treatment effect:

$$ATE = (20 + 20 + 100 + 50 + 0 + 200)/6 = 65$$

This would mean that sales increase the amount sold, on average, by 65 units. As for the ATT, it would just be the mean of the last column when $T = 1$:

$$ATT = (50 + 0 + 200)/3 = 83.33$$

This is saying that, for the business that chose to cut prices (where treated), lowered prices increased the amount they sold, on average, by 83.33 units. Finally, the average effect conditioned on being one week prior to Christmas ($x = 1$) is simply the average of the effect for units 3 and 6.

$$CATE(x = 1) = (100 + 200)/2 = 150$$

And the average effect on Christmas week is the the average treatment effect when $x = 0$:

$$CATE(x = 0) = (20 + 20 + 50 + 0)/4 = 22.5$$

meaning that business benefited from price cuts much more one week prior the Christmas season (150 units), compared to price cuts in the same week as Christmas (increase of 22.5 units).

Now that you have a better understanding about the causal quantities you are usually interested in (ATE, ATT and $CATE$), it's time to leave fantasy island and head back to the real world. Here things are brutal and the data you actually have, much harder to work with. Here, you can only see one potential outcome, which makes the individual treatment effect hidden from us.

	i	y0	y1	t	x
0	1	200.0	NaN	0	0
1	2	120.0	NaN	0	0
2	3	300.0	NaN	0	1
3	4	NaN	500.0	1	0
4	5	NaN	600.0	1	0
5	6	NaN	800.0	1	1

MISSING DATA PROBLEM

One way to see causal inference is as a missing data problem. To infer the causal quantities of interest, you must impute the missing potential outcomes.

You might look at this and ponder "this is certainly not ideal, but can't I just take the mean of the treated and compare it to the mean of the untreated? In other words, can't I just do $ATE = (500 + 600 + 800)/3 - (200 + 120 + 300)/3 = 426.33$? No! you've just committed the gravest sin of mistaking association for causation!

Notice how different the results are. The ATE you calculated earlier was less than 100 and now you are saying it is something above 400. The issue here is that the business that engaged in sales would have already sold more regardless of price cut. To see this, just go back to when you could see both potential outcomes. Notice how Y_0 for the treated units are much higher than that of the untreated units. This difference in Y_0 between treated groups makes it much harder to uncover the treatment effect by simply comparing both groups.

Although comparing means is not the smartest of ideas, I think that your intuition is in the right place. It's time to apply the new concepts that you just learned to refine this intuition and finally understand why association is not causation. It's time to face the main enemy of causal inference.

Bias

BIAS

You can say that an estimator is biased if it differs from the parameter it is trying to estimate, $Bias = E\left[\hat{\beta} - \beta\right]$. For example, an estimator for the average treatment effect is biased if it's systematically under or over estimating the true ATE.

To get right to the point, **bias is what makes association different from causation**. The fact that what you estimate from data doesn't match the causal quantities you want to recover is the whole issue. Fortunately, this can easily be understood with some intuition. Let's recap our business example. When confronted with the claim that cutting prices increases the amount sold by a business, you can question it by saying that those businesses that did sales would probably have sold more anyway, even without the price decrease. That is because they are probably bigger and can afford to do more aggressive sales. In other words, it is the case that treated businesses (businesses on sale) are not comparable with untreated businesses (not on sale).

To give a more formal argument, you can translate this intuition using potential outcome notation. First, notice that, in order to estimate the ATE, you need to estimate what would have happened to the treated had they not been treated, $E[Y_0|T=1]$, and what would happen to the untreated, had they been treated, $E[Y_1|T=0]$. When you compare the average outcome between treated and untreated, you are essentially using $E[Y|T=1]$ to estimate $E[Y_0|T=1]$ and $E[Y|T=0]$ to estimate $E[Y_1|T=0]$. In other words, you are estimating the $E[Y|T=t]$ hoping to recover $E[Y_t]$. If they don't match, an estimator with recovers $E[Y|T=t]$, like the average outcome for those that got treatment t, will be a biased estimator of $E[Y_t]$.

Back to intuition, you can even leverage your understanding of how the world works to go even further. You can say that, probably, Y_0 of the treated business is bigger than Y_0 of the untreated business. That is because businesses that can afford to engage in price cuts tend to sell more regardless of those cuts. Let this sink in for a moment. It takes some time to get used to talking about potential outcomes, as it involves reasoning about things that would have happened but didn't. Reread this paragraph and make sure you understand it.

The Bias Equation

Now that you understand why a sample average can differ from the average potential outcome it is trying to estimate, you can take a look at why difference in averages does not recover the ATE, in general. In our example, association between the treatment and the outcome is measured by $E[Y|T=1] - E[Y|T=0]$. This is the average amount sold for the business on sales minus the average amount sold for those not on sales. On the other hand, causation is measured by $E[Y_1 - Y_0]$.

To understand how they are different, let's take the association measure, $E[Y|T=1] - E[Y|T=0]$, and replace the observed outcomes with the potential outcomes. For the treated, the observed outcome is Y_1. For the untreated, the observed outcome is Y_0.

$$E\left[Y|T=1\right] - E\left[Y|T=0\right] = E\left[Y_1|T=1\right] - E\left[Y_0|T=0\right]$$

Now, let's add and subtract $E[Y_0|T=1]$. This is a counterfactual outcome. It tells what would have happened to the outcome of the treated, had they not received the treatment.

$$E\left[Y|T=1\right] - E\left[Y|T=0\right] = E\left[Y_1|T=1\right] - E\left[Y_0|T=0\right] + E\left[Y_0|T=1\right] - E\left[Y_0|T=$$

Finally, you can reorder the terms, merge some expectations, and lo and behold:

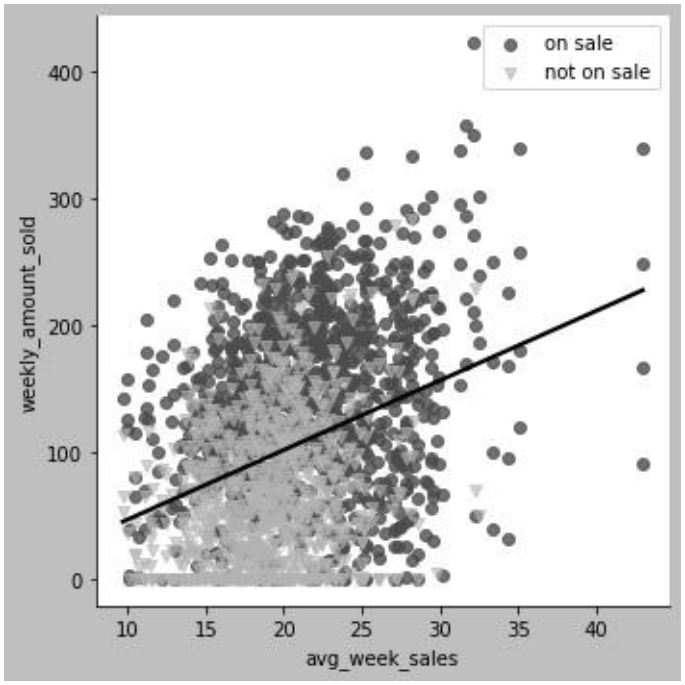
$$E\left[Y|T=1\right]-E\left[Y|T=0\right]=\underbrace{E[Y_1-Y_0|T=1]}_{ATT}+\underbrace{\{E\left[Y_0|T=1\right]-E\left[Y_0|T=0\right]\}}_{BIAS}$$

This simple piece of math encompasses all the problems you'll encounter in causal questions. I cannot stress how important it is that you understand every aspect of it. If you're ever forced to tattoo something on your arm, this equation should be a good candidate for it. It's something to hold onto dearly and understand what it's telling you, like some sacred text that can be interpreted 100 different ways. In fact, let's take a deeper look. Let's break it down into some of its implications. First, this equation tells why association is not causation. As you can see, the association is equal to the treatment effect on the treated plus a bias term. **The bias is given by how the treated and control group differ regardless of the treatment, that is, in case neither of them received the treatment** . You can now say precisely why you are suspicious when someone tells you that price cuts boost the amount sold by such a high number. You think that, in this example, $E\left[Y_0|T=0\right]<E\left[Y_0|T=1\right]$, that is, business that can afford to do price cuts sell more, **regardless of doing sales**.

Why does this happen? That's an issue for chapter 3, when you'll look into confounding, but, for now, you can think of bias arising because many things you can't observe are changing together with the treatment. As a result, the treated and untreated businesses differ in more ways than just whether or not they do sales. They also differ on size, location, the week they choose to do sales, management style, cities they are located in and many other factors. For you to say how much price cuts increase the amount sold, you would need for business with and without sales to be, on average, similar to each other. In other words, treated and control units would have to be exchangeable.

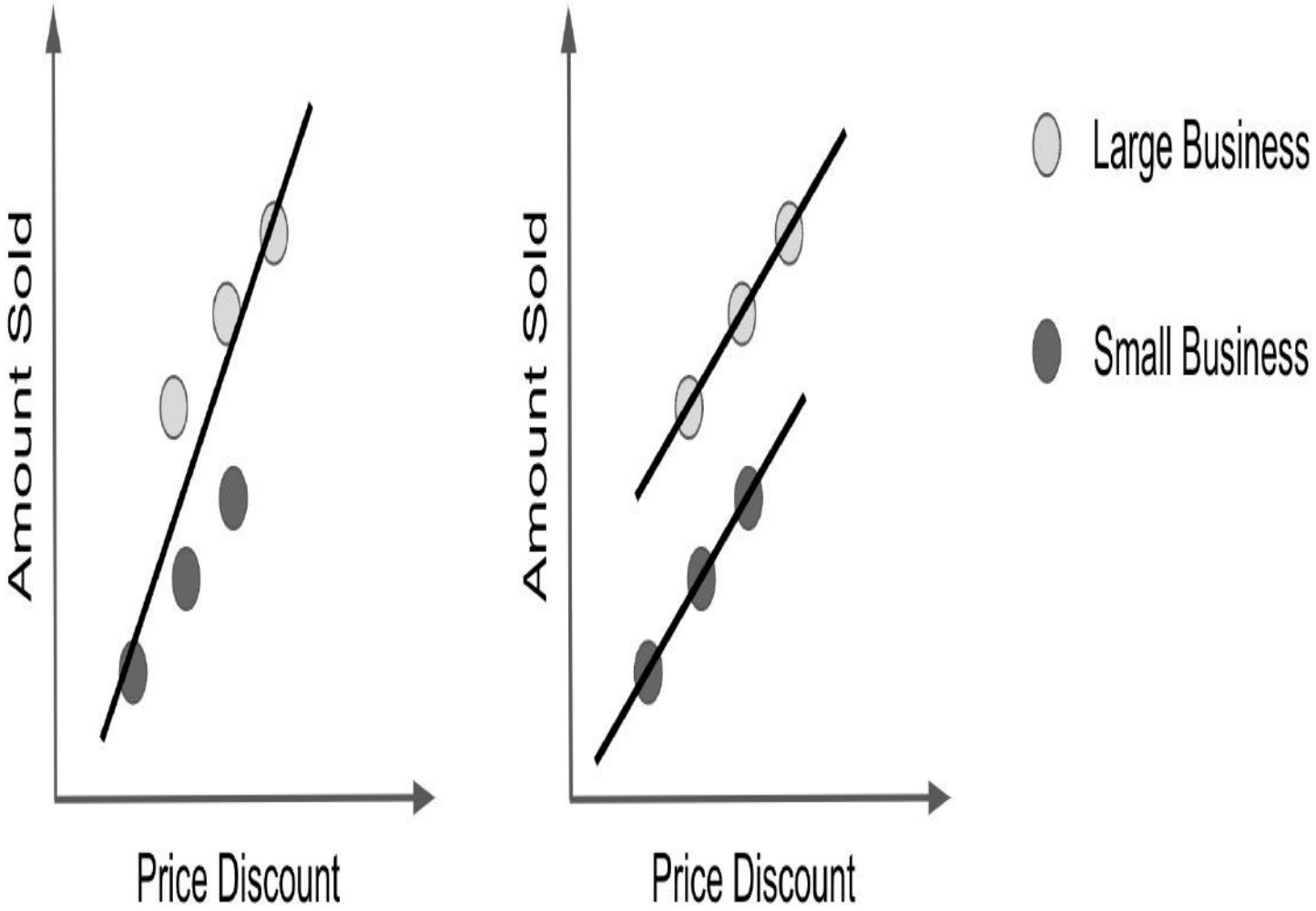
A Visual Guide to Bias

You don't have to only use math and intuition to talk about exchangeability. In our example, you can even check that they are not exchangeable by plotting the relationship in outcome by covariates for the different treatment groups. If you plot the outcome (`weekly_amount_sold`) by business size, as measured by `avg_week_sales` and color each plot by the treatment, `is_on_sale`, you can see that the treated - business on sale - are more concentrated to the right of the plot, meaning that they are usually bigger businesses. Another way of saying this is that treated and untreated are not balanced.

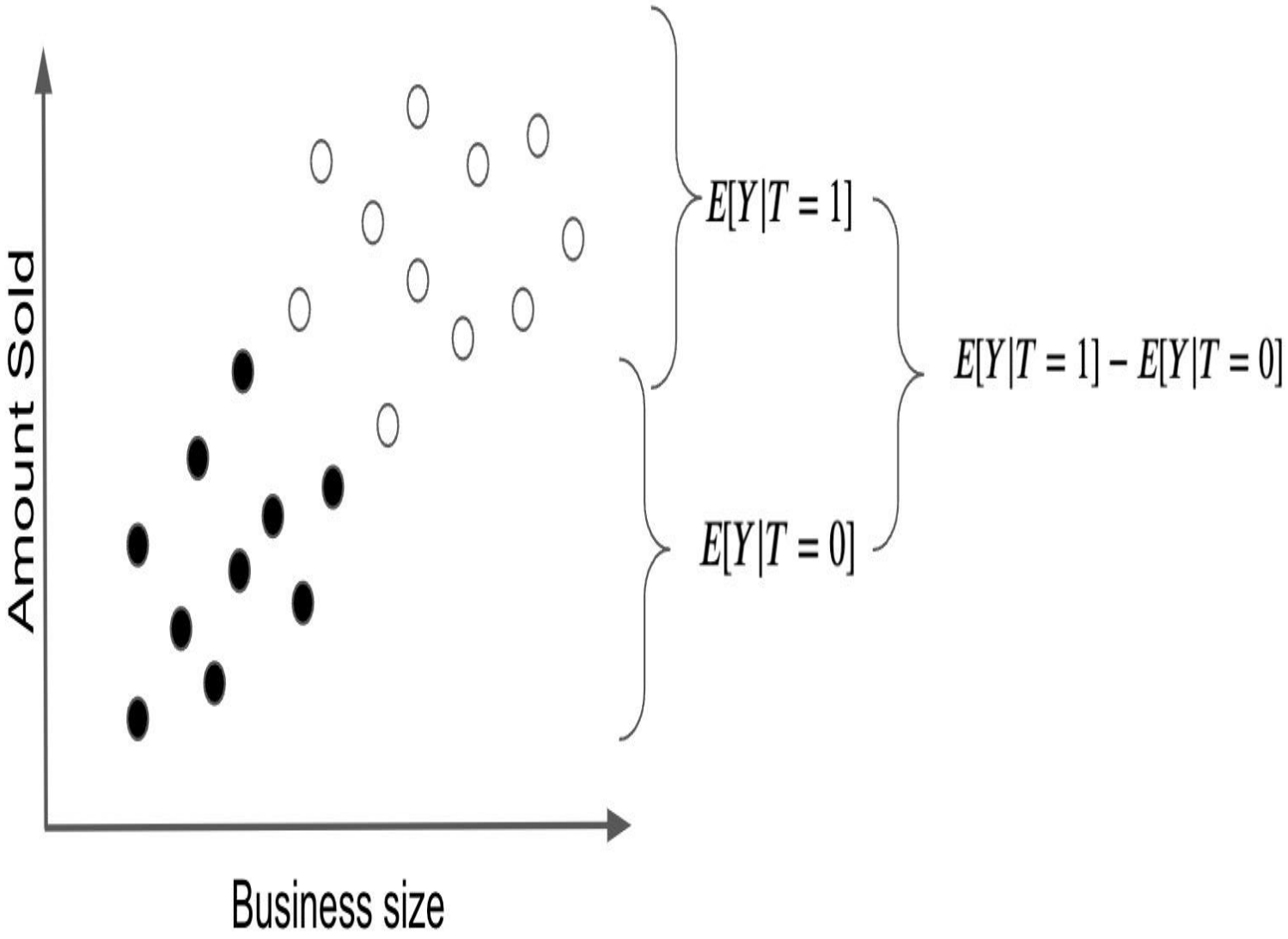


This is very strong evidence that your hypothesis $E[Y_0|T=1] > E[Y_0|T=0]$ was correct. That is, there is upward bias, as both the number of businesses with price cuts ($T=1$) and the outcome of those businesses, had they not done any sale, (Y_0 for those businesses), would go up with business size.

If you've ever heard about the Simpson's Paradox, this bias is like a less extreme version of it. In Simpson's Paradox, the relationship between two variables is initially positive, but, once you control for a third variable, it becomes negative. In our case, bias is not so extreme as to flip the sign of the association. Here, you start with a situation where the association between price cuts and amount sold is too high and controlling for a third variable reduces the size of that association. If you zoom in inside businesses of the same size, the relationship between price cuts and amount sold decreases, but remains positive.



Once again, this is so important that I think it is worth going over it again, now with pretty pictures. They are not realistic, but they do a good job of explaining the issue with bias. Let's suppose you have a variable indicating the size of the business. If you plot the amount sold against size, you'll see an increasing trend, where the bigger the size, the more the business sells. Next, you color the dots according to the treatment: white dots are businesses that did a price cut and black dots, businesses that didn't do that. If you simply compare the average amount sold between treated and untreated business, this is what you'll get:

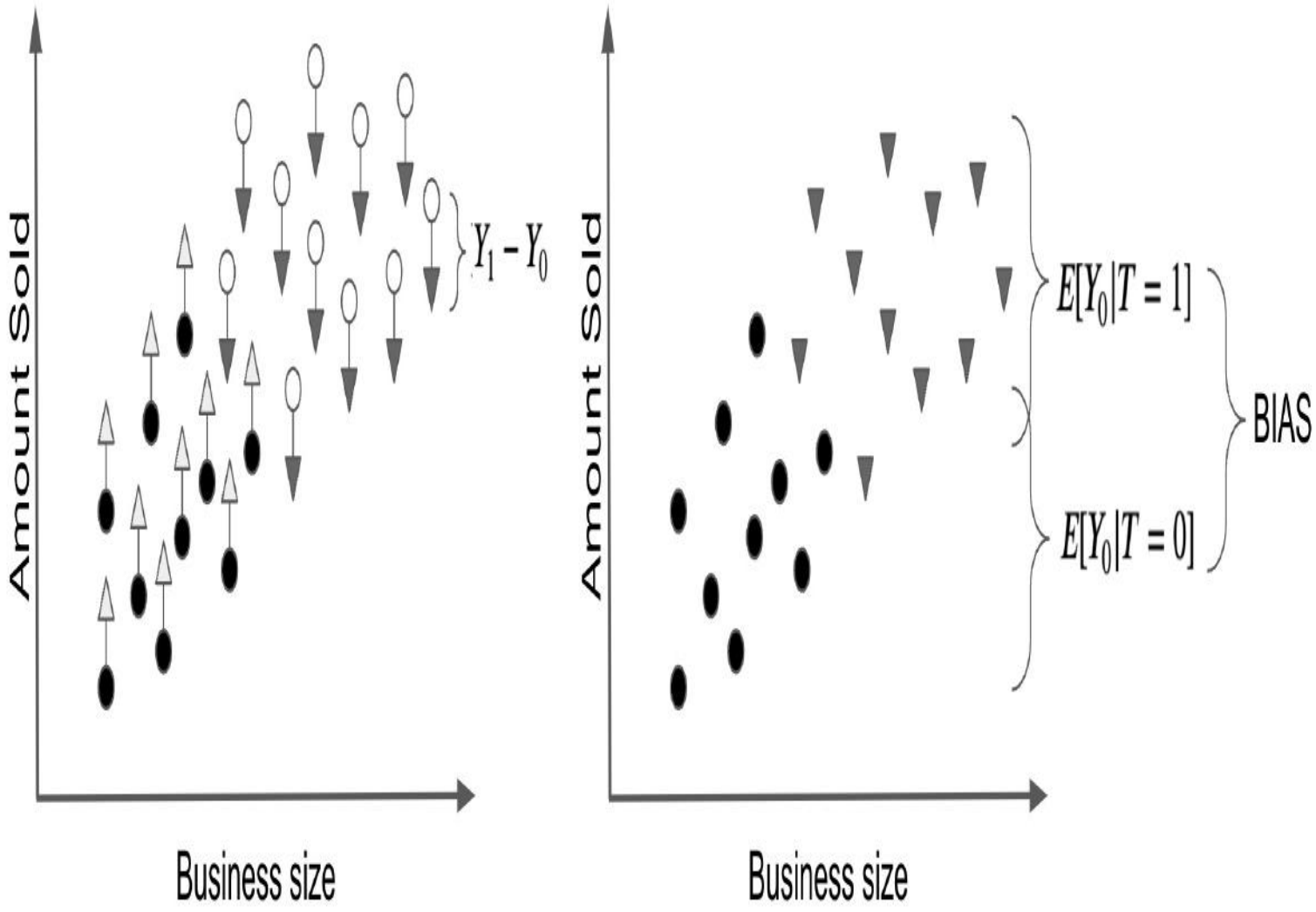


Notice how the difference in amount sold between the two groups can (and probably do) have two causes:

1. The treatment effect. The increase in the amount sold which is caused by the price cut.
2. The business size. Bigger businesses are both able to sell more, and do more price cuts. This source of difference between the treated and untreated is NOT due to the price cut.

The challenge in causal inference is untangling both causes.

Contrast this with what you would see if you add both potential outcomes to the picture (counterfactual outcomes are denoted as triangles). The individual treatment effect is the difference between the unit's outcome and another theoretical outcome that the same unit would have if it got the alternative treatment. The average treatment effect you would like to estimate is the average difference between the potential outcomes for each individual unit, $Y_1 - Y_0$. Notice that these individual differences are much smaller than the difference you saw in the previous plot, between treated and untreated groups. The reason for this is bias, which is depicted in the right plot below.



You can represent the bias by setting everyone to not receive the treatment. In this case, you are only left with the Y_0 potential outcome. Then, you can see how the treated and untreated groups differ on those potential outcomes under no treatment. If they do, something other than the treatment is causing the treated and untreated to be different. This is precisely the bias I've been talking about. It is what shadows the true treatment effect.

Identifying the Treatment Effect

Now that you understand the problem it's time to look at the (or at least one) solution. Identification is the first step in any causal inference analysis. You'll see much more of it in chapter three, but for now, it's worth knowing what it is. Recall that you can't observe the causal quantities, since only one potential outcome is observable. This means you can't directly estimate something like $E[Y_1 - Y_0]$, since you can't observe this difference for any data point. But perhaps you can find some other quantity, which is observable, and can be used to recover the causal quantity you care about. This is identification: **is figuring out how to recover causal quantities from observable data**. For instance, if, by some sort of miracle, $E[Y|T = t]$ did recover $E[Y_t]$ (identify $E[Y_t]$), you would be able to get $E[Y_1 - Y_0]$ by simply estimating $E[Y|T = 1] - E[Y|T = 0]$. This can be done by estimating the

average outcome for the treated and untreated, which are both observed quantities.

> **SEE ALSO**
>
> In the last decade, (2010-2020), an entire body of knowledge on causal identification was popularized by Judea Pearl and his team, as an attempt to unify the causal inference language. I use some of that language in this chapter - although probably an heretical version of it - and we will cover more about it in chapter 3. If you want to learn more about it, a short yet really cool paper to check out is *Causal Inference and Data Fusion in Econometrics*, by Paul Hünermund and Elias Bareinboim.

You can also see identification as the process of getting rid of bias. Using potential outcomes, you can also say what would be necessary to make association equal to causation. **If $E[Y_0|T=0] = E[Y_0|T=1]$, then, association IS CAUSATION!** Understanding this is not just remembering the equation. There is a strong intuitive argument here. To say that $E[Y_0|T=0] = E[Y_0|T=1]$ is to say that treatment and control group are comparable regardless of the treatment. Or, had the treated not been treated, if you could observe its Y_0, it would be the same as that of the untreated, at least on average. Mathematically, the bias term would vanish, leaving only the effect on the treated:

$$E[Y|T=1] - E[Y|T=0] = E[Y_1 - Y_0|T=1] = ATT$$

Also, if the treated and the untreated respond similarly to the treatment, that is, $E[Y_1 - Y_0|T=1] = E[Y_1 - Y_0|T=0]$, then (pay close attention), **difference in means BECOMES the average causal effect**:

$$E[Y|T=1] - E[Y|T=0] = ATT = ATE = E[Y_1 - Y_0]$$

Despite the seemingly fancy schmancy math here, all it's saying is that, **once you make treated and control group interchangeable**, expressing the causal effect in terms of observable quantities in the data becomes trivial. Tying this to our example, if businesses that do and don't do price cuts are similar to each other - that is, exchangeable - then, the difference in amount sold between the ones on sale and those not on sale can be entirely attributed to the price cut.

The Independence Assumption

This exchangeability is the key assumption in causal inference. Since it's so important, different scientists found different ways to state it. I'll start with one way, probably the most common, which is the **independence assumption**. Here, I'll say that the potential outcomes are independent of the treatment

$$(Y_0, Y_1) \perp T$$

This independence means that $E[Y_0|T] = E[Y_0]$, or, in other words, that the treatment gives you no information about the potential outcomes. The fact that a unit was treated doesn't mean it would have a lower or higher outcome, had it not been treated (Y_0). This is just another way of saying that $E[Y_0|T=1] = E[Y_0|T=0]$. In our business example, it simply means that you wouldn't be able to tell apart the businesses that chose to engage in sales from those that didn't, had they all not done any

sales. Except for the treatment and its effect on the outcome, they would all be similar to each other. Similarly, $E[Y_1|T] = E[Y_1]$ means that you also wouldn't be able to tell them apart, had they all engaged in sales. Simply put, it means that treated and untreated groups are comparable and indistinguishable, regardless of whether they all received the treatment or not.

Identification with Randomization

Here, you are treating independence as an assumption. That is, you know you need to make associations equal to causation, but you have yet to learn how to make this condition hold. Recall that a causal inference problem is often broken down into two steps:

1. Identification, where you figure out how to express the causal quantity of interest in terms of observable data.
2. Estimation, where you actually use data to estimate the causal quantity identified earlier.

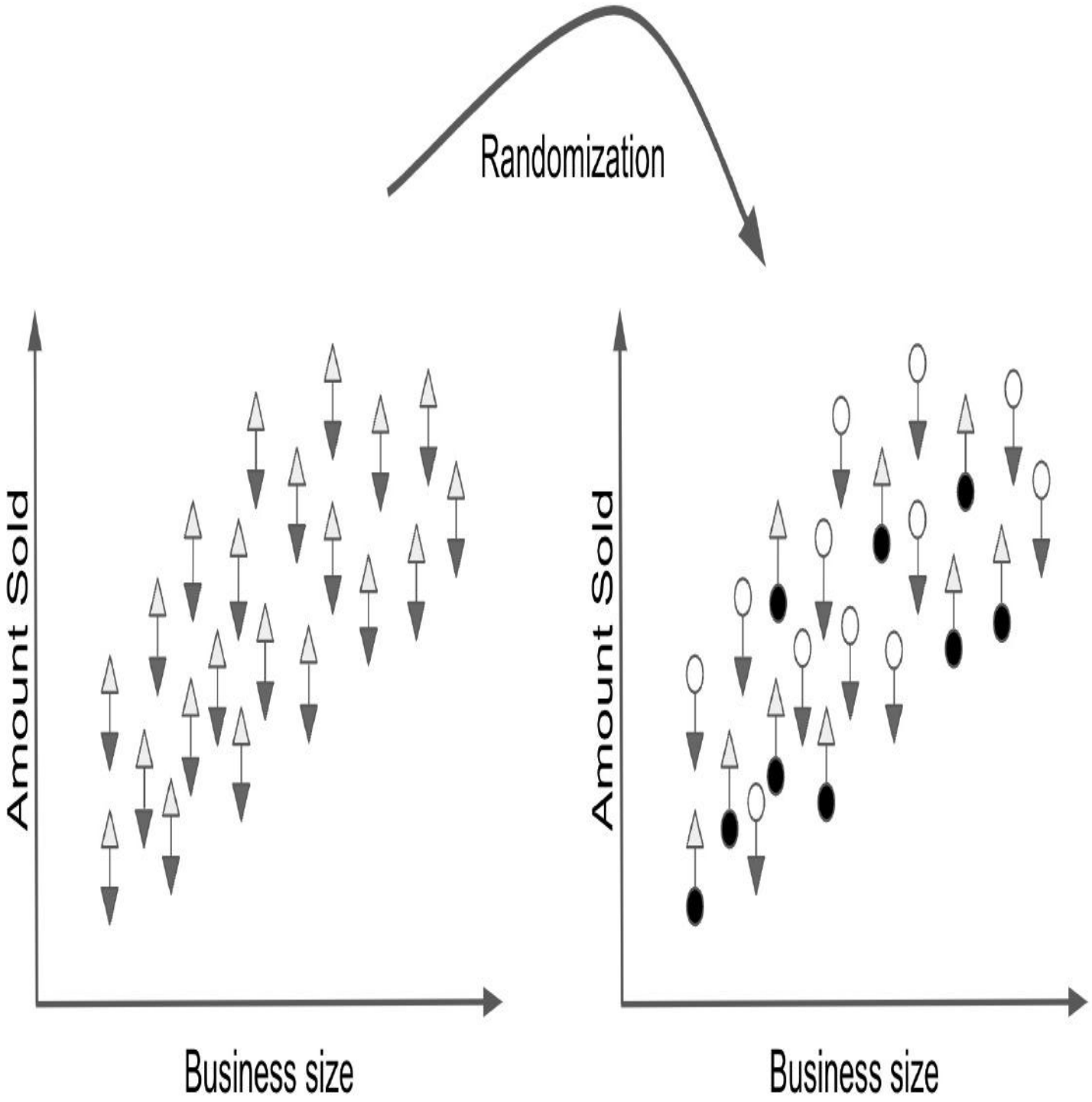
To illustrate this process with a very simple example, let's suppose that you can randomize the treatment. I know I said earlier that, in the online marketplace you work in, businesses had full autonomy on setting prices, but you can still find a way to randomize the treatment $isOnSales$. For instance, let's say that you negotiate with the businesses the right to force them to cut prices, but the marketplace will pay for the price difference you've forced. Ok, you now have a way to randomize sales, so what? This is a huge deal, actually!

First, randomization ties the treatment assignment to a coin flip, so variations in it become completely unrelated to any other factors in the causal mechanism.

$$IsOnSales \leftarrow rand(t)$$
$$AmountSold \leftarrow f_y(IsSales, u_y)$$

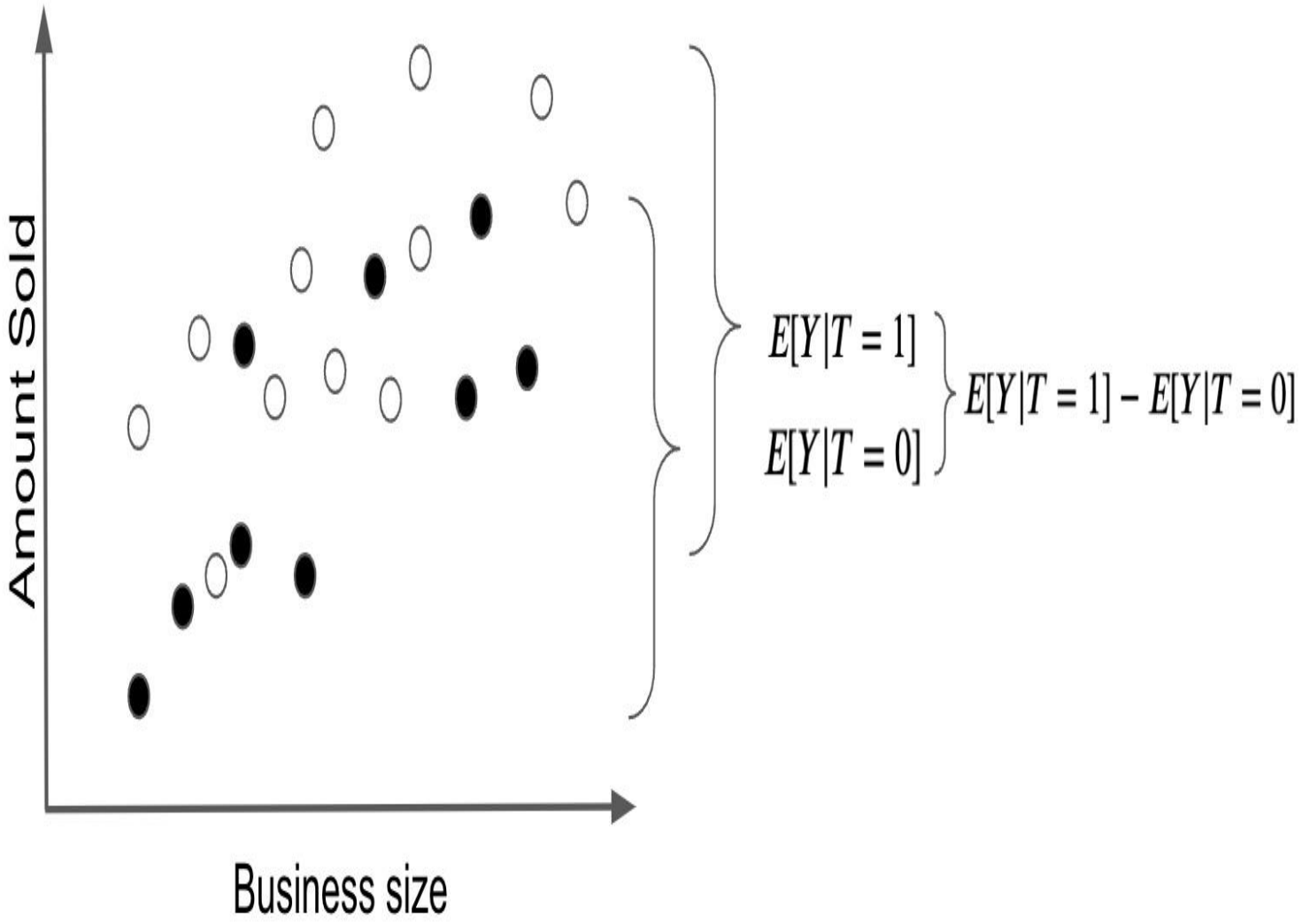
Under randomisation, u_t vanished from our model, since the assignment mechanism of the treatment becomes fully known. Moreover, since the treatment is random, it becomes independent from anything, including the potential outcomes. This means it pretty much forces independence, $(Y_0, Y_1) \perp T$, to hold.

To make this crystal clear, let's see how randomisation pretty much annihilates bias. We start before the treatment assignment. It's a world full of potential yet to be realized. This is depicted by the image on the left.

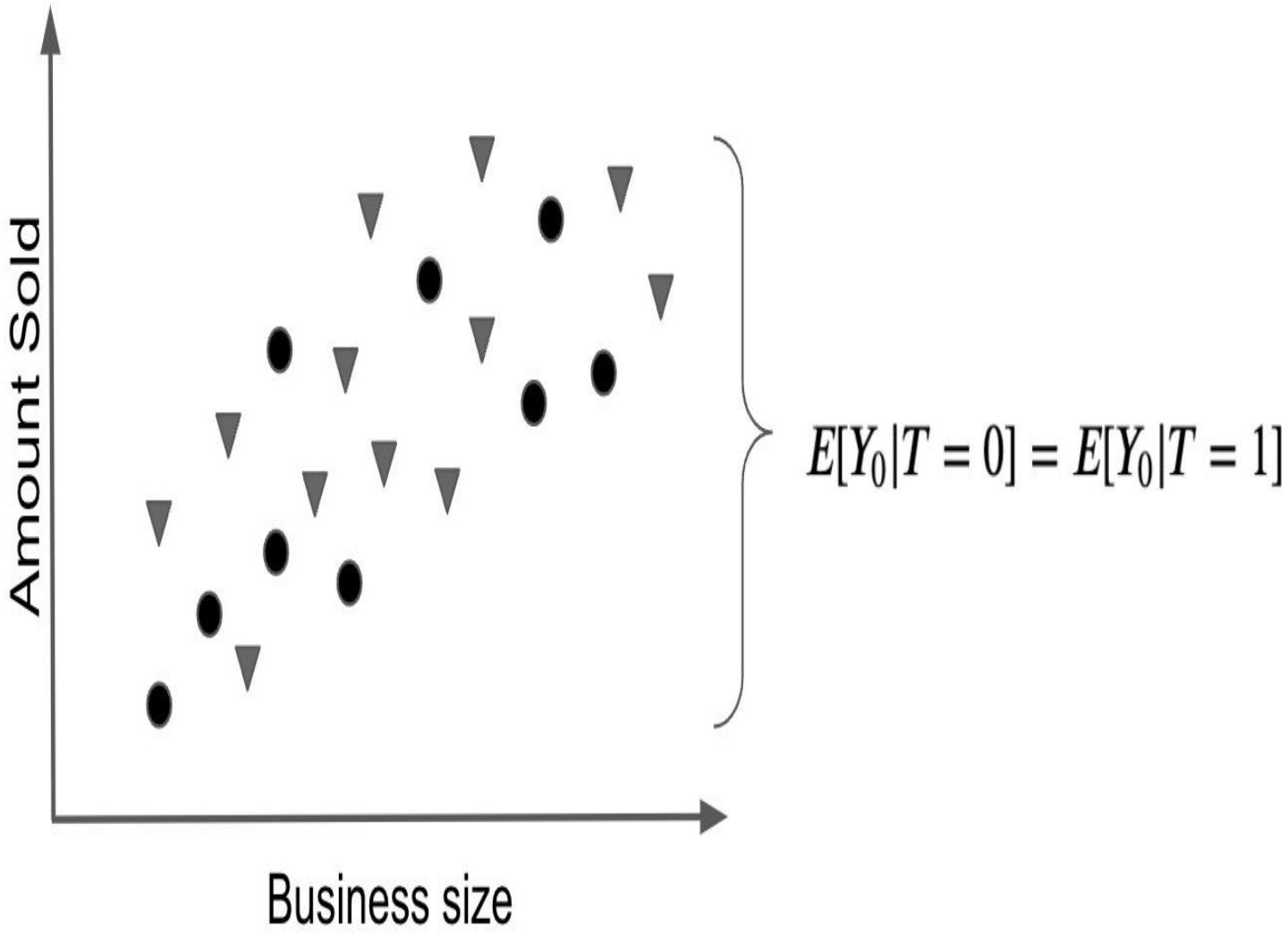


Then, at random, the treatment realizes one or the other potential outcome.

Next, let’s get rid of the clutter (the unrealized potential outcomes) and compare treated to untreated.



In this case, the difference in the outcome between treated and untreated **is** the average causal effect. This happens because there is no other source of difference between them, other than the treatment itself. All the differences you see must be attributed to the treatment. Or, simply put, there is no bias. If you set everyone to not receive the treatment so that you only observe the Y_0s, you would find no difference between the treated and untreated groups.



RANDOMIZED VS OBSERVATIONAL

In causal inference, we use the term **randomized** to talk about data where the treatment was randomized or when the assignment mechanism is fully known. In contrast to that, the term **observational** is used to describe data where you can see who got what treatment, but you don't know how that treatment was assigned.

This is what the herculean task of causal identification is all about. It's about finding clever ways of removing bias and making the treated and the untreated comparable so that all the difference you see is only the average treatment effect. Importantly, **identification is only possible if you know how the treatment was distributed or assigned**. This is why I said earlier that data alone cannot answer causal questions. Sure, data is important for estimating the causal effect. But, besides data, you'll always require a statement about how the data - specifically, the treatment - came to be. You get that statement using your expert knowledge or by intervening in the world, influencing the treatment and observing how the outcome changes in response.

Ultimately, causal inference is about figuring out how the world works, stripped of all delusions and misinterpretations. And now that you understand this, you can move forward to mastering some of the most powerful methods to remove bias, the instruments of the brave and true, to identify the causal

effect.

Chapter Key Ideas

We learned the mathematical language we will use to talk about causal inference. Importantly, we defined potential outcomes as the outcome we would observe for a unit had that unit took a specific treatment $T = t$

$$Y_{it} = Y_i|do\left(T = t\right)$$

Potential outcomes were very useful in understanding why association is different from causation. Namely, when treated and untreated are different due to reasons other than the treated, $Y_0\left|T = 1 \neq Y_0\right|T = 0$, and the comparison between both groups will not yield the true causal effect, but a biased estimate. We also used potential outcomes to see what we would need to make association equal to causation:

$$(Y_0, Y_1) \perp T$$

When treated and control group are interchangeable or comparable, like when we randomize the treatment, a simple comparison between the outcome of the treated and untreated group will yield the treatment effect

$$E\left[Y_1 - Y_0\right] = E\left[Y|T = 1\right] - E\left[Y|T = 0\right]$$

Other Examples

To really fix the concepts reviewed above, I'm including this short list of broad examples. I recommend you check them out while trying to tie them to what you just learned.

A Glass of Wine a Day Keeps the Doctor Away

A popular belief is that Wine, in moderation, is good for your health. The argument is that mediterranean cultures, like italians and spanish, are famous for drinking a glass of wine every day and also display high longevity.

After reading this chapter, you should be suspicious about this claim. To attribute the extended lifespan to drinking wine, those that drink and those that don't would need to be exchangeable and we know that they are not. For instance, Italy and Spain have both generous healthcare systems and elevated IDH, compared to the rest of the world. In technical terms, $E\left[Lifespan_0|WineDrinking = 1\right] > E\left[Lifespan_0|WineDrinking = 0\right]$, so bias might be clouding the true causal effect.

An Incredible Membership Program

A big online retailer implemented a membership program, where members pay an extra fee to have access to more discounts, faster deliveries, return fee waiver and amazing customer service. To understand the impact of the program, the company rolled it out to a random sample of the customers, which could opt in for paying the fee to get the membership benefits. After a while, they saw that customers in the membership program were much more profitable than those in the control group.

Customers not only bought from the company, but they also spent less time with customer service. Should we then say that the membership program was a huge success in increasing sales and decreasing time spent serving customers?

Not really. Although the eligibility to the program was randomized, the random chunk which could opt in is still self-selected into the program. In other words, randomization of program eligibility ensures that people who were able to get the program are comparable to those that weren't. But, out of the eligible, only a fraction choose to participate. This choice was not random. Probably, only the more engaged customers choose to participate, while the casual ones dropped out. So, even though eligibility to the program was randomized, participation in the program was not. The result is that those that participated are not comparable to those that didn't.

If you think about it, out of the eligible customers, the ones that actually choose to participate probably opted in for the program precisely because they already spent a lot in the online company, which made the extra discounts something worth paying for. This would imply that $E[Revenues_0|OptIn = 0 > Revenues_0|OptIn = 0]$, meaning those customers that opted in probably generate more revenues regardless of the program.

Chapter 2. Randomized Experiments and Stats Review

A NOTE FOR EARLY RELEASE READERS

With Early Release ebooks, you get books in their earliest form—the author's raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the 2nd chapter of the final book. Please note that the GitHub repo will be made active later on.

If you have comments about how we might improve the content and/or examples in this book, or if you notice missing material within this chapter, please reach out to the editor at *vwilson@oreilly*.

Now that you know the basics about causality, its time to talk about the inference part in causal inference. This chapter will first recap some of the concepts from previous chapters in the context of randomized experiments. Randomized experiments are the golden standard for causal inference, so it is really important that you understand what makes them special. Even when randomization is not an option, having it as an ideal to strive for will be immensely helpful when thinking about causality.

Next, I'll use randomized experiments to review some important statistical concepts and tools, such as error, confidence interval, hypothesis tests, power and sample size calculations. If you know about all of this, I'll make it clear when the review will start so you can skip it.

Brute Force Independence with Randomization

In the previous chapter, you saw why and how association is different from causation. You also saw what is required to make association equal to causation.

$$E[Y|T=1] - E[Y|T=0] = \underbrace{E[Y_1 - Y_0|T=1]}_{ATT} + \underbrace{\{E[Y_0|T=1] - E[Y_0|T=0]\}}_{BIAS}$$

To recap, association becomes causation if there is no bias. There will be no bias if $E[Y_t|T=0] = E[Y_t|T=1]$. In other words, association will be causation if the treated and control are equal or comparable, except for their treatment. Or, in slightly more technical terms, when the potential outcomes of the treated are equal to the potential outcomes of the untreated, at least in expectations. Remember that potential outcome Y_{ti} is the outcome you would see had unit i received treatment t.

In the previous chapters, I also briefly touched on how to make association equal to causation by virtue of randomizing the treatment. Essentially, when you randomize the treatment T, you are trying it to the

flip of a coin, that is, a random mechanism completely known to us. Notice that this coin doesn't have to be fair. You could assign the treatment to just 10% of the units, 1%, or even less. As long as the assignment mechanism is random, you can get the right conditions to identify the treatment effect. Randomization brute force your way towards independence between the treatment and the potential outcomes:

$$(Y_0, Y_1) \perp T$$

Importantly, notice that I'm **not** talking about the independence between the treatment and the outcome. If that was the case, the treatment would have no impact on the outcome for you to measure. For example, let's say that the treatment is a new feature in your App and the outcome is time spent in that App. Saying that $Feature \perp TimeSpent$ means that time spent in the app is the same in both treated and untreated groups. In other words, the new feature has simply no effect.

Instead, what you want is for the **potential outcomes** to be independent of the treatment. There is an important distinction here. Saying that $(Y_0, Y_1) \perp T$ means that the outcome of both treatment and untreated **would be the same**, had they all received the treatment ($Y_1 \perp T$) or had they all received the control ($Y_0 \perp T$). It is saying that, if you didn't roll out the new feature to the treated group, this rollout group would spend the same time in the app as the control group ($Y_0 \perp T$). Additionally, had you chosen to give the feature to everyone, treated and control group would also have the same outcome ($Y_1 \perp T$).

Another simpler way of putting this is that the independence assumptions implies that treatment and control groups are comparable. Or that knowing the treatment assignment doesn't give me any information on how the outcome was, previous to the treatment. Consequently, $(Y_0, Y_1) \perp T$ means that the treatment is the only thing generating a difference in visible outcome, between the treated and in the control groups.

$$E[Y_0|T=0] = E[Y_0|T=1] = E[Y_0]$$

And

$$E[Y_1|T=0] = E[Y_1|T=1] = E[Y_1]$$

Which, as you've seen, makes it so that a simple comparison in averages between treated and control group is able to identify the average treatment effect

$$E[Y|T=1] - E[Y|T=0] = E[Y_1 - Y_0] = ATE$$

Let's now talke all this math and go over an example so you can see that it is actually quite simple. In the next section, I'll use randomized control trials (RCT) in order to understand the impact of cross sell emails.

An A/B Testing Example

A common strategy among companies is to have a cheap or even free product which isn't profitable, but serves as the doorway to attracting new customers. Once the company has those customers, it can then cross sell other products which are more profitable. Let's suppose you work for a coffee delivery company. Your main product is a low cost monthly subscription which allows the customers to have high quality and curated coffee delivered to them on a weekly basis. Beyond this basic and low cost subscription, your company provides a more premium one, with brewing perks and the world's finest coffee, from local producers in the small town of Divinolandia, Brazil (also known as my Mom's neighbor). This is by far your most profitable service and so, your goal is to increase selling it to the users who have already subscribed for your low cost, entry product.

In order to do that, your company has a marketing team which tries to sell the premium coffee delivery subscription to its customers. They do this mainly through cross sell emails. As the causal inference expert, your goal is to understand how effective those emails are.

When you look into the data to answer this question, you can clearly see that the customers who received an email were more likely to buy the premium subscription. In technical terms, when a customer buys the product you are trying to sell, you can say they converted. So, you can say that the customers who received an email converted more:
$E[Conversion|Email = 1] > E[Conversion|Email = 0]$.

Unfortunately, you also discover that the marketing team tends to send emails for the customers which they thought were more likely to convert in the first place. It is not entirely clear how they did this. Maybe they looked for customers which interacted the most with the company, or those that answered positively in a satisfaction survey. Regardless, this is very strong evidence that
$E\left[Conversion_0|Email = 1\right] > E\left[Conversion_0|Email = 0\right]$, which means that a simple comparison in means is a biased estimate of the true causal effect of the cross sell email.

To solve that, you need to make the treated and untreated comparable:
$E\left[Y_0|T = 1\right] = E\left[Y_0|T = 0\right]$. One way to force this is by randomly assigning the emails. If you manage to do that, the treatment and untreated will have, on average, the same conversion, except for the treatment they receive. So that is what you do. You select three random samples from your customer base. To one of them, you don't send any emails. To the other, you send a large and beautifully written email about the premium subscription. To the last sample, you send a short and to the point email about the premium subscription. After some time collecting data, you have something that looks like this

```
(323, 4)
```

	gender	cross_sell_email	age	conversion
0	0	short	15	0
1	1	short	27	0
2	1	long	17	0
3	1	long	34	0
4	1	no_email	14	0

You can see that you have 323 samples. It's not exactly big data, but something you can work with.

> **SIMULATED VS REAL WORLD DATA**
>
> When teaching about causal inference, it is very helpful to use simulated data. First, because causal inference is always accompanied by a statement about how the treatment was assigned. Simulated allows me to talk about this assignment mechanism without any uncertainty. Second, causal inference involves counterfactual quantities which I can choose to show in order to give a better explanation of what is going on. However, so that the data doesn't look too artificial, I often take real world data and transform it to fit the example I'm trying to give. For instance, this example takes data from the paper *A Randomized Assessment of Online Learning* (2016), by Alpert, William T., Kenneth A. Couch, and Oskar R. Harmon, and transforms it to look like cross sell email data.

To estimate the causal effect, you can simply compute the average conversion for each of the treatment groups.

	gender	age	conversion
cross_sell_email			
long	0.550459	21.752294	0.055046
no_email	0.542553	20.489362	0.042553
short	0.633333	20.991667	0.125000

Yup. It's that simple. You can see that the group assigned to no email had a conversion rate of 4.2%, while the groups assigned to the long and short email had a conversion rate of 5.5% and whooping 12.5%, respectively. This means that the ATEs, measured as the difference between each treated group and the control group, $ATE = E[Y|T = t] - E[Y|T = 0]$, where 1.3 and 8.3 percentage points increase for the long and short email, respectively. Interestingly, sending an email which is short to the point seems better than an elaborated one.

The beauty of RCTs is that you no longer have to worry if the marketing team somehow targeted

customers who were likely to convert or, for that matter, you don't have to worry that the customers from the distinct treatment groups are different in any systematic way, other than the treatment they received. By design, the random experiment is made to wipe out those differences, making $(Y_0, Y_1) \perp T$, at least in theory.

Checking for Balance

In practice, a good sanity check to see if the randomisation was done right (or if you are looking at the correct data) is to check if the treated are equal to the untreated in pre-treatment variables. For example, you have data on gender and age and you can see whether these two characteristics are balanced across treatment groups.

When you look at age, groups seem very much alike, but there seems to be a difference in gender ($woman = 1, man = 0$). It seems that the group that received the short email had 63% men, compared to 54% in the control group and 55% in the group which got the long email. This is somewhat unsettling, as the treatment group in which you found the highest impact also appears to be different from the other groups. So, even if independence should hold in theory in RCTs, it does not necessarily hold in practice. It could be that the large effect you saw for the short email was due to the fact that, for whatever reason, $E[Y_0|man] > E[Y_0|woman]$.

This should draw your attention to what happens with a small dataset. Even under randomisation, it could be that, by chance, one group is different from another. In large samples, this difference tends to disappear. It also brings forth the issue of how much difference is enough for you to conclude that the treatments are indeed effective and not just due to chance, which is something I'll address shortly.

The Ideal Experiment

Randomized experiments or Randomized Controlled Trials (RCTs) are the most reliable way to get causal effects. It's a straightforward technique and absurdly convincing. It is so powerful that most countries have it as a requirement for showing the effectiveness of new drugs. To make a terrible analogy, you can think of RCT as Aang, from Avatar: The Last Airbender, while other causal inference techniques are more like Sokka. Sokka is cool and can pull some neat tricks here and there, but Aang can bend the four elements and connect with the spiritual world. Think of it this way, if you could, RCT would be all you would ever do to uncover causality. A well designed RCT is the dream of any scientist and decision maker.

Unfortunately, they tend to be either very expensive - both in money, but more importantly, in time - or just plain unethical. Sometimes, you simply can't control the assignment mechanism. Imagine yourself as a physician trying to estimate the effect of smoking during pregnancy on baby weight at birth. You can't simply force a random portion of moms to smoke during pregnancy. Or say you work for a big bank, and you need to estimate the impact of the credit line on customer churn. It would be too expensive to give random credit lines to your customers. Or that you want to understand the impact of increasing the minimum wage on unemployment. You can't simply assign countries to have one or another minimum wage. Moreover, as you will see in chapter three, there are some situations (selection biased ones) where not even RCTs can save you.

Still, I would like you to think about random experiments beyond a tool for uncovering causal effects. Rather the goal here is to use it as a benchmark. Whenever you do causal inference without RCTs, you should always ask yourself what is the experiment you would like to have in order to answer your question. Even if that ideal experiment is not feasible, it serves as a valuable benchmark. It often helps you to shed some light on how we can discover the causal effect even without an experiment.

The Most Dangerous Equation

Now that you understand the value of an experiment, it's time to review what it means to not have infinite data. Causal inference is a two step process. RCTs are invaluable in helping with identification, but if the sample size of an experiment is small, you'll struggle with the second step: inference. In order to understand this, it's worth reviewing some statistical concepts and tools. If you are already familiar with them, feel free to skip to the next chapter.

In his famous article of 2007, Howard Wainer writes about very dangerous equations:

> *"Some equations are dangerous if you know them, and others are dangerous if you do not. The first category may pose danger because the secrets within its bounds open doors behind which lies terrible peril. The obvious winner in this is Einstein's iconic equation* $E = MC^2$*, for it provides a measure of the enormous energy hidden within ordinary matter. […] Instead I am interested in equations that unleash their danger not when we know about them, but rather when we do not. Kept close at hand, these equations allow us to understand things clearly, but their absence leaves us dangerously ignorant."*

The equation he talks about is Moivre's equation:

$$SE = \frac{\sigma}{\sqrt{n}}$$

where SE is the standard error of the mean, σ is the standard deviation, and n is the sample size. This piece of math is definitely something you should master, so let's get to it.

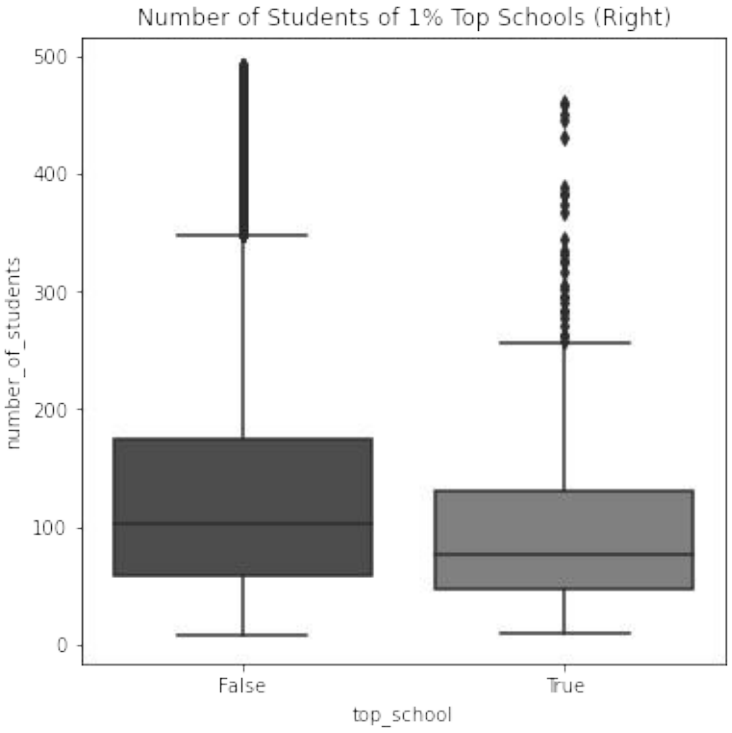
To see why not knowing this equation is very dangerous, let's look at some education data. I've compiled data on ENEM scores (Brazilian standardized high school scores, similar to SATs) from different schools over a 3 year period. I've also cleaned the data to keep only the information relevant to you in this section. The original data can be downloaded on the Inep website.

If you look at the top-performing school, something catches the eye: those schools have a reasonably small number of students.

```
import pandas as pd
import numpy as np
from scipy import stats

df = pd.read_csv("data/enem_scores.csv")
df.sort_values(by="avg_score", ascending=False).head(10)
```

	year	school_id	number_of_students	avg_score
16670	2007	33062633	68	82.97
16796	2007	33065403	172	82.04
16668	2005	33062633	59	81.89
16794	2005	33065403	177	81.66
10043	2007	29342880	43	80.32
18121	2007	33152314	14	79.82
16781	2007	33065250	80	79.67
3026	2007	22025740	144	79.52
14636	2007	31311723	222	79.41
17318	2007	33087679	210	79.38

Looking at it from another angle, you can separate only the 1% of top schools and study them. What are they like? Perhaps you can learn something from the best and replicate it elsewhere. And sure enough, if you look at the top 1% of schools, you'll figure out they have, on average, fewer students.

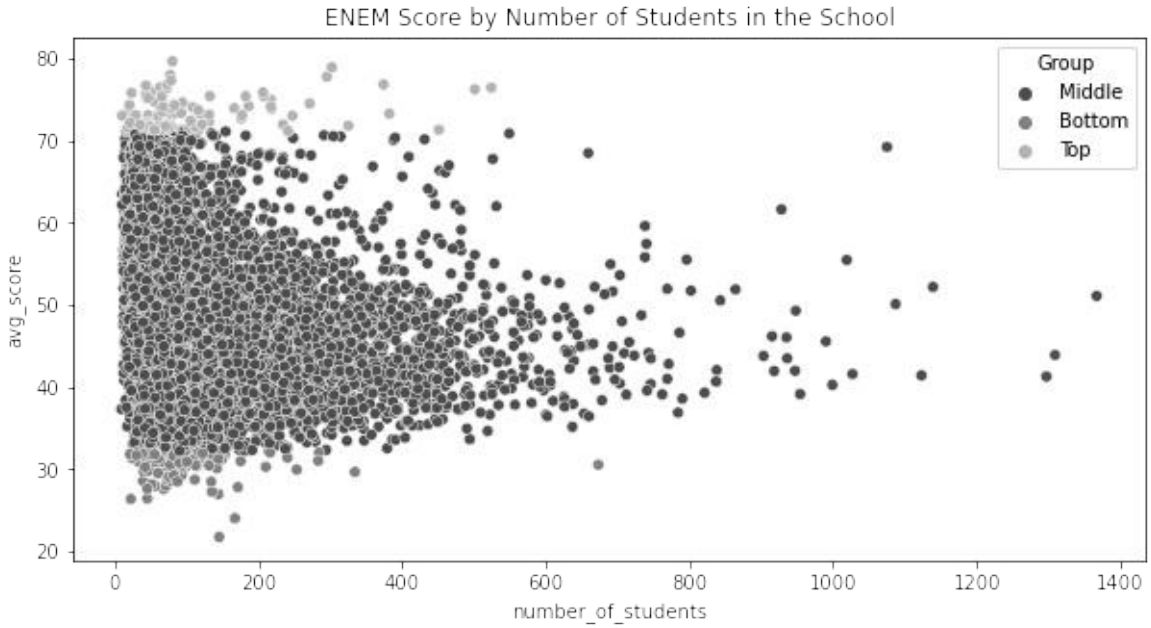


One natural conclusion is that small schools lead to higher academic performance. This makes intuitive sense, since we believe that fewer students per teacher allows the teacher to give focused attention to each student. But what does this have to do with Moivre's equation? And why is it dangerous?

Well, it becomes dangerous once people start to make important and expensive decisions based on this information. In his article, Howard continues:

“In the 1990s, it became popular to champion reductions in the size of schools. Numerous philanthropic organizations and government agencies funded the division of larger schools because students at small schools are overrepresented in groups with high test scores.”

What people forgot to do was to also look at the bottom 1% of schools. If you do that, lo and behold! They also have very few students!



What you see here is precisely what’s expected according to the Moivre’s equation. As the number of students grows, the average score becomes more and more precise. Schools with very few students (low sample size) can have very high and low scores simply due to chance. This is less likely to occur in large schools. Moivre’s equation talks about a fundamental fact about the reality of information and records in the form of data: it is always imprecise. The question then becomes how imprecise. And what can you do to take those inaccuracies into account?

Statistics is the science that deals with these imprecisions, so they don’t catch you off-guard. As Taleb puts it in his book, Fooled by Randomness:

> *Probability is not a mere computation of odds on the dice or more complicated variants; it is the acceptance of the lack of certainty in our knowledge and the development of methods for dealing with our ignorance.*

One way to quantify our uncertainty is the **variance of our estimates**. Variance tells you how much observation deviates from its central value. As Moivre’s equation indicates, this uncertainty shrinks as the amount of data you observe increases. This makes sense, right? If you see many students performing excellently at a school, you can be more confident that this is indeed a good school. However, if you see a school with only 10 students and 8 of them perform well, you need to be more suspicious. By chance, it could be that the school got some above-average students.

The beautiful triangular plot you see above tells precisely this story. It shows you how your estimate of the school performance has a huge variance when the sample size is small. It also indicates that variance shrinks as the sample size increases. This is true for the average score in a school, but it is also true

about any summary statistics you might have, including the ATE you often want to estimate. Back to our cross sell email application, if you had thousands of customers in each treatment group, instead of hundreds, you would be much more confident that the difference in conversion you saw between treated and control groups are not simply due to chance.

The Standard Error of Our Estimates

Since this is just a review of statistics, I'll take the liberty to go a bit faster. If you are not familiar with distributions, variance, and standard errors, please read on, but keep in mind that you might need some additional resources. I suggest you google any MIT course on introduction to statistics. They are usually quite good and you can watch them for free on YouTube.

In the previous section, you estimated the average treatment effect $E[Y_1 - Y_0]$ as the difference in the means between the treated and the untreated $E[Y|T = 1] - E[Y|T = 0]$. Specifically, you figured out the ATE for two types of cross sell emails on conversion. You then saw that the short email had a very impressive lift, of more than 8 percentage points, while the long email had a smaller impact, of just 1.3 percentage points increase. The question you now need to answer is if those effects are large enough so you can be confident that they are not just due to chance. In technical terms, you need to see if they are statistically significant.

To do so, you first need to estimate the SE, according to the equation I've shown earlier. n is pretty easy to get. You just need the `len` of each treatment. Or, you can use Pandas `groupby` followed by a `size` aggregation.

```
short_email = data.query("cross_sell_email=='short'")["conversion"]
long_email = data.query("cross_sell_email=='long'")["conversion"]
email = data.query("cross_sell_email!='no_email'")["conversion"]
no_email = data.query("cross_sell_email=='no_email'")["conversion"]

data.groupby("cross_sell_email").size()
```

```
cross_sell_email
long        109
no_email     94
short       120
dtype: int64
```

To get the estimate for the standard deviation, you can apply the following equation

$$\widehat{\sigma}^2 = \sqrt{\frac{1}{N-1}\sum_{i=0}^{N}(x - \overline{x})^2}$$

where $\overline{x}$ is the mean of x. Fortunately for you, most programming software already implements this. In Pandas, you can use the method std. Putting it all together, you have the following function for the standard error.

```
def se(y: pd.Series):
    return y.std() / np.sqrt(len(y))
```

```
print("SE for Long Email:", se(long_email))
print("SE for Short Email:", se(short_email))
```

```
SE for Long Email: 0.021946024609185506
SE for Short Email: 0.030316953129541618
```

Knowing this formula is incredibly handy (we'll come back to it multiple times, trust me), but know that Pandas also has a built-in method for calculating the standard error, `.sem()`, as in `standard error of the mean`.

```
print("SE for Long Email:", long_email.sem())
print("SE for Short Email:", short_email.sem())
```

```
SE for Long Email: 0.021946024609185506
SE for Short Email: 0.030316953129541618
```

Confidence Intervals

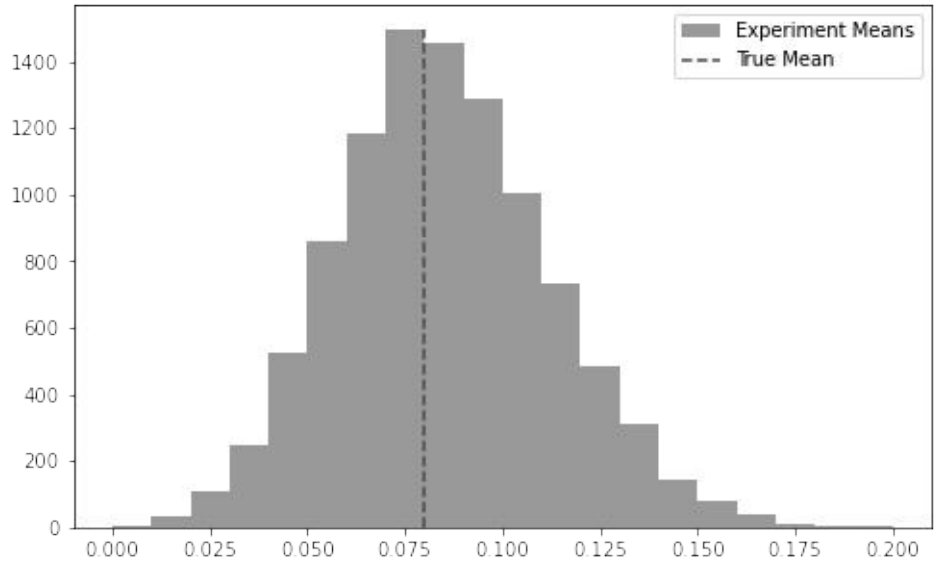
The standard error of your estimate is a measure of confidence. You need to go into turbulent and polemic statistical waters to understand precisely what it means. For one view of statistics, the frequentist view, we would say that our data is nothing more than a manifestation of an accurate data-generating process. This process is abstract and ideal. It is governed by true parameters that are unchanging but also unknown to us. In the context of cross sell semail, if you could run multiple experiments and collect multiple datasets with the conversion rate associated with each email, all of those experiments would resemble the true underlying data generating process which dictates how the emails drive conversion. However, no experiment would be exactly like it. This is very much like Plato's writing on the Forms:

> *Each [of the essential forms] manifests itself in a great variety of combinations, with actions, with material things, and with one another, and each seems to be many*

To understand this, let's suppose you have the true abstract distribution of conversion for the short cross sell email. Because conversion is either zero or one, it follows a Bernoulli distribution and let's say that the probability of success in this distribution is 0.08. This means that whenever a customer receives the short email, it has an 8% chance of converting. Next, let's pretend you can run 10 thousand experiments. On each one, you collect a sample of 100 customers, send them the short email and observe the average conversion, giving you a total of 10 thousand conversion rates. If you plot the 10 thousand conversion rates on an histogram, you can see that they are distributed around the true mean of 0.08. Some experiments will have a conversion rate lower than the true one, and some will be higher, but the mean of the 10 thousand conversion rate will be pretty close to the true mean.

```
n = 100
conv_rate = 0.08

def run_experiment():
    return np.random.binomial(1, conv_rate, size=n)

np.random.seed(42)
```

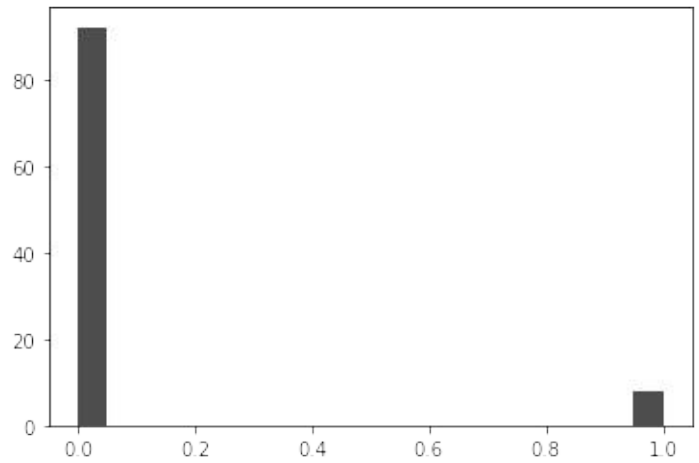
```
experiments = [run_experiment().mean() for _ in range(10000)]
```



Notice that I'm talking about the mean of means here, since the conversion rate is just the average conversion of a sample. So, by chance, you could have an experiment where the mean is somewhat below or above the true mean. This is to say that you can never be sure that the mean of our experiment matches the true platonic and ideal mean. However, **with the standard error, you can create an interval that will contain the true mean in 95% of the experiments you run**.

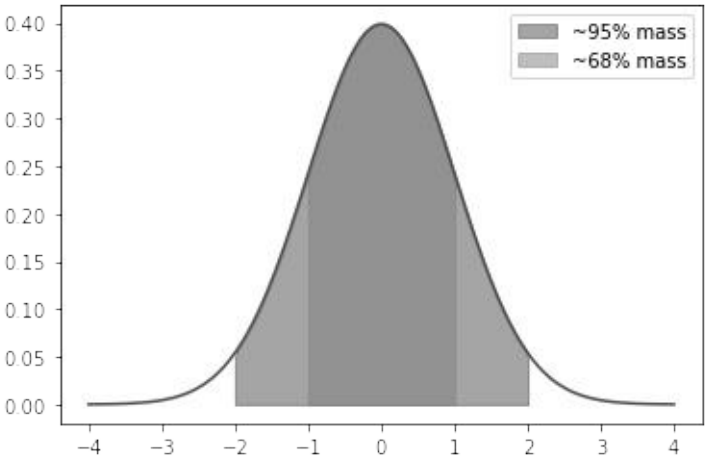
In real life, you don't have the luxury of simulating the same experiment with multiple datasets. You often only have one. But you can draw on the intuition above to construct what is called a **confidence interval**. Confidence intervals come with a probability attached to them. The most common one is 95%. This probability tells you that if you were to do multiple experiments and construct the 95% confidence interval in each one of them, the true mean would fall inside the interval 95% of the time.

To calculate the confidence interval, you'll use what is perhaps the most mind blowing result in statistics. Take a closer look at the distribution of conversion rates you've just plotted. Now, remember that conversion is either zero or one and hence follows a Bernoilli distribution. If you plot this Bernoulli distribution in a histogram, it will have a huge bar at 0 and small bar at 1, since the success rate is only 8%. This looks nothing like a normal distribution right?



This is where that mind blowing result comes to play. Even though the distribution of the data is not normally distributed (like in the conversion case, which follows a bernoulli distribution), the **average of**

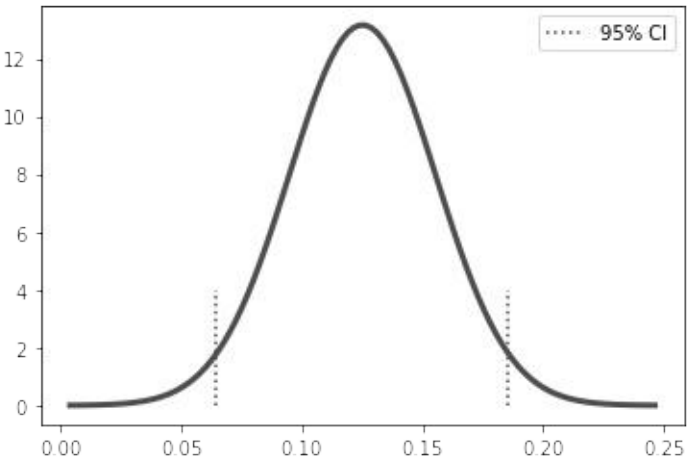
the data is always normally distributed (Ok, not always, but in most situations). This means that, if you collect data on conversion multiple times and calculate the average conversion each time, those averages will follow a normal distribution. This is very neat, because normal distribution is very well known and you can do all sorts of interesting things with it. For example, for the purpose of calculating the confidence interval, you can leverage knowledge from statistical theory that 95% of the mass of a normal distribution falls between 2 standard deviations above and below the mean (technically, 1.96, but 2 is a good approximation which is easier to remember).



Back to your cross sell experiments, you now know that the conversion rate from this experiment follows a normal distribution. The best estimate you have for the mean of that (unknown) distribution is the mean from your small experiment. Moreover, the standard error serves as your estimate of the standard deviation that unknown distribution of the experiment means. So, if you multiply the standard error by 2 and add and subtract it from the mean of your experiments, you will construct a 95% confidence interval for the true mean.

```
exp_se = short_email.sem()
exp_mu = short_email.mean()
ci = (exp_mu - 2 * exp_se, exp_mu + 2 * exp_se)
print("95% CI for Short Email: ", ci)
```

```
95% CI for Short Email:  (0.06436609374091676, 0.18563390625908324)
```



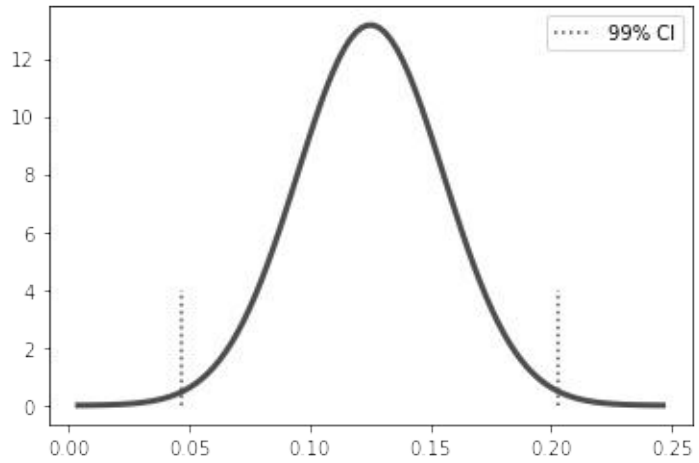
Of course, you don’t need to restrict yourself to the 95% confidence interval. You could generate the 99% interval by finding what you need to multiply the standard deviation by so that the interval contains 99% of the mass of a normal distribution.

To do that, you can use the `ppf` function from `scipy`. This function gives you the inverse of the CDF (cumulative distribution function) of a standard normal distribution. So, `ppf(0.5)` will return 0.0, saying that 50% of the mass of the standard normal distribution is below 0.0; `ppf(0.975)` ($1 - 0.5/2$) will give you 1.96, saying that 97.5% of the mass of the standard normal distribution is below 1.96. Since the normal distribution is symmetric, the fact that 97.5% of its mass falls below 1.96 also means that 2.5% of its mass falls below -1.96, meaning that 95% of the mass falls between -1.96 and 1.96, which is what you used to construct the 95% confidence interval. To get the 99% CI, you can use `ppf(0.995)`, which will give a `z` value such that 99% of the distribution's mass falls between `-z` and `z`.

```
z = stats.norm.ppf(.995)
print(z)
ci = (exp_mu - z * exp_se, exp_mu + z * exp_se)
ci
```

```
2.5758293035489004
```

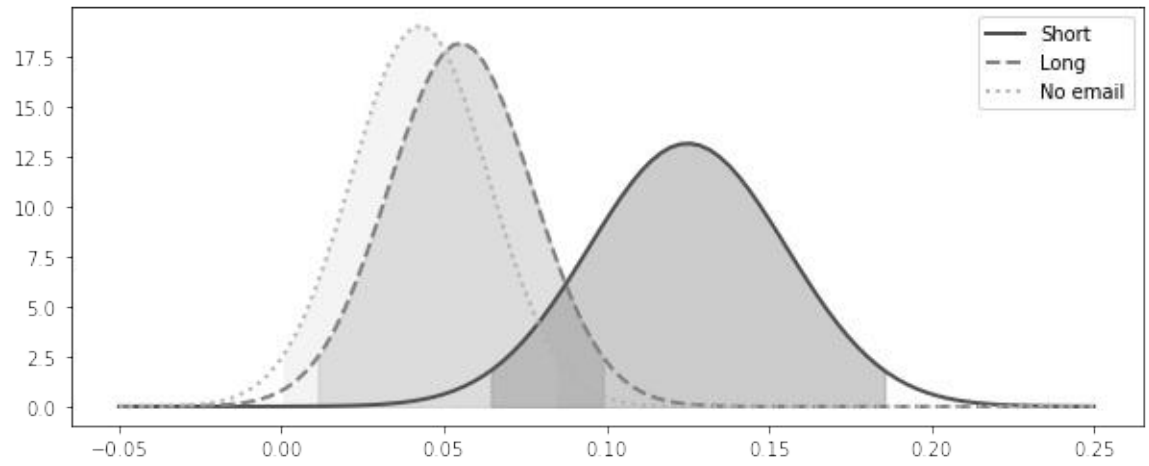
```
(0.04690870373460816, 0.20309129626539185)
```



This shows you the 99% of the conversion rate of sending a short email to customers. But you could very well show the 95% CI for the conversion rate of the customers that got the long email, as well as for those in the control group.

```
def ci(y: pd.Series):
    return (y.mean() - 2 * y.sem(), y.mean() + 2 * y.sem())

print("95% CI for Short Email:", ci(short_email))
print("95% CI for Long Email:", ci(long_email))
print("95% CI for No Email:", ci(no_email))
```

```
95% CI for Short Email: (0.06436609374091676, 0.18563390625908324)
95% CI for Long Email: (0.01115382234126202, 0.09893792077800403)
95% CI for No Email: (0.0006919679286838468, 0.08441441505003955)
```



You can see that the 95% CI of the three groups overlap with each other. If they didn't, you would be able to conclude that the difference in conversion between the groups is not simply by chance. In other words, you would be able to say that sending a cross sell email causes a statistically significant difference in conversion rates. But since the intervals do overlap, you can't say that. At least not yet. Importantly, overlapping confidence intervals is not enough to say that the difference between the groups is not statistically significant, however, if they didn't overlap, that would mean they are statistically different. In other words, non overlapping confidence intervals are conservative evidence for statistical significance.

To recap, before you move on, confidence intervals are a way to place uncertainty around our estimates. The smaller the sample size, the larger the standard error, and hence, the wider the confidence interval. Since they are super easy to compute, lack of confidence intervals signals either some bad intentions or simply lack of knowledge, which is equally concerning. Finally, you should always be suspicious of measurements without any uncertainty metric.

One final word of caution here. Confidence intervals are trickier to interpret than at first glance. For instance, I **shouldn't** say that a particular 95% confidence interval contains the true mean with 95% chance. In frequentist statistics that use confidence intervals, the population mean is regarded as a true population constant. This constant is either inside or outside a particular confidence interval. In other words, a specific confidence interval either contains or doesn't contain the true mean. If it does, the chance of containing it would be 100%, not 95%. If it doesn't, the chance would be 0%. Instead, in confidence intervals, the 95% refers to the frequency that such confidence intervals, computed in many studies, contain the true mean. 95% is our confidence in the algorithm used to calculate the 95% CI, not on the particular interval itself.

Now, having said that, as an Economist (statisticians, please look away now), I think this purism is not very useful. In practice, you will see people saying that the particular confidence interval contains the true mean 95% of the time. Although wrong, this is not very harmful, as it still places a visual degree of uncertainty in our estimates. What I mean by this is that I would rather you have a confidence interval around your estimate and interpret it wrong than not avoiding the confidence interval in fear of misinterpretation. I don't care if you say they contain the true mean 95% of the time. Just, please, never forget to place them around your estimates; otherwise, you will look silly.

Hypothesis Testing

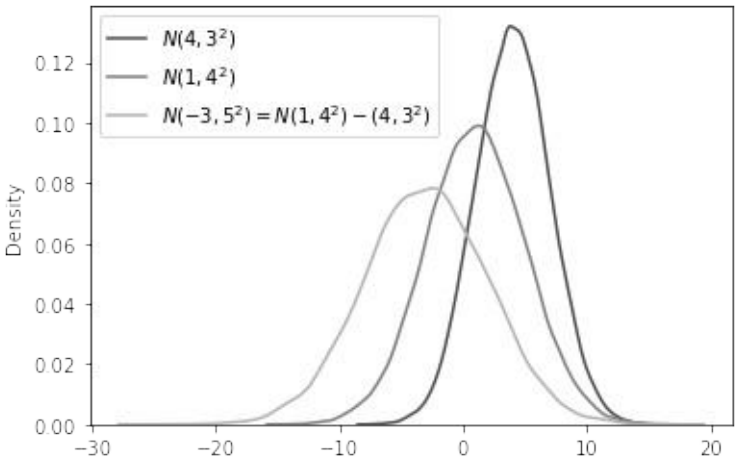
Another way to incorporate uncertainty is to state a hypothesis test: is the difference in means between two groups statistically different from zero (or any other value)? In order to answer these type of questions, you need to recall that the sum or difference of 2 independent normal distributions is also a normal distribution. The resulting mean will be the sum or difference between the two distributions, while the variance will always be the sum of the variance:

$$N\left(\mu_1, \sigma_1^2\right) - N\left(\mu_2, \sigma_2^2\right) = N\left(\mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2\right)$$

$$N\left(\mu_1, \sigma_1^2\right) + N\left(\mu_2, \sigma_2^2\right) = N\left(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2\right)$$

If you don't recall, it's OK. You can always use code and simulated data to check for yourself:

```
np.random.seed(123)

n1 = np.random.normal(4, 3, 30000)
n2 = np.random.normal(1, 4, 30000)
n_diff = n2 - n1

sns.distplot(n1, hist=False, label="$N(4,3^2)$")
sns.distplot(n2, hist=False, label="$N(1,4^2)$")
sns.distplot(n_diff, hist=False, label=f"$N(-3, 5^2) = N(1,4^2) - (4,3^2)$")
plt.legend()
plt.show()
```



If you take two groups, each with a distribution attached to it, and subtract one from the other, you'll end with a third distribution. The mean of this final distribution will be the difference in the means, and the standard deviation of this distribution will be the square root of the sum of the variances, which are just the squared standard errors. Since you are talking about the distributions of the data averages, you can think about the standard deviation of these as the standard error of the mean.

$$\mu_{diff} = \mu_1 - \mu_2$$

$$SE_{diff} = \sqrt{SE_1^2 + SE_2^2}$$

You can take this idea and apply it to the problem of comparing the conversion from our cross sell email experiment. If you take the estimated distribution of two groups - let's say, the short email and the no email group - and subtract one from the other, you get the distribution of the difference. Then, with the resulting distribution, you can easily construct a 95% confidence interval for the difference.

```
diff_mu = short_email.mean() - no_email.mean()
diff_se = np.sqrt(no_email.sem()**2 + short_email.sem()**2)

ci = (diff_mu - 1.96*diff_se, diff_mu + 1.96*diff_se)
print(f"95% CI for the different (short email - no email):\n{ci}")
```

```
95% CI for the different (short email - no email):
(0.01023980847439844, 0.15465380854687816)
```

Null Hypothesis

With this interval, you can answer questions about what is called a null hypothesis. For example, you can state the hypothesis that there is no difference in conversion between the random sample of customers who receive no email and the random sample of customers who received the short email. You'll usually use H_0 to talk about the null hypothesis.

$$H_0 : Conv_{no\_email} = Conv_{short\_email}$$

Once you have this hypothesis, it's time to ask yourself, "is it likely that I would observe such a difference if the null hypothesis was true?". This means you'll look at the data and see if it conforms to our null hypothesis. If it doesn't, you'll say that seeing such data would be too weird, if the null hypothesis was true and hence, you should reject it. One way to do this is with the confidence intervals you have just constructed.

Notice how the 95% confidence interval above does not contain zero. Also, recall that this is the CI of the difference between conversion rates. Since the null hypothesis states that this difference is zero, but you can see that the confidence interval is entirely outside zero, you can say that the probability of seeing such a result would be too low, if the null hypothesis was true. Hence, you can reject the null hypothesis with 95% confidence.

Of course you can also formulate other null hypotheses, besides the one one which states no difference at all. For example, let's say there is some cost to sending emails, which is very realistic. Even if there is no significant monetary cost, if you send too many emails to customers, eventually they will flag you as spammers, which will shut down this communication channel with them, leading to lower sales in the future. Under this situation, perhaps the marketing team is only willing to roll out the cross sell email if the lift in conversion rate is higher than 1%. Then, you can state the null hypothesis as follows: "the difference in conversion rate is 1% or lower". To test this hypothesis, all you need to do is shift the confidence interval by subtracting 1% from the difference in means

```
diff_mu_shifted =  short_email.mean() - no_email.mean() - 0.01 # shifting the CI
diff_se = np.sqrt(no_email.sem()**2 + short_email.sem()**2)

ci = (diff_mu_shifted - 1.96*diff_se, diff_mu_shifted + 1.96*diff_se)
print(f"95% CI 1% difference between (short email - no email):\n{ci}")
```

```
95% CI 1% difference between (short email - no email):
(0.00023980847439844521, 0.14465380854687815)
```

Since this 95% CI is also above zero, you can also reject this other null hypothesis. Again, this means

that the chance of observing such an extreme result is lower than 5%, if the null hypothesis was indeed true. However, now the 95% CI is very close to zero, meaning you would not be able to reject the null hypothesis of the effect being lower than something like 2%, at least not with a 95% confidence.

Test Statistic

Besides confidence intervals, sometimes is useful to think about rejecting the null hypothesis in terms of a **test statistic**. These statistics are often constructed such that higher values point towards rejection of the null. One of the most commonly used test statistics is the t-statistic. It can be defined by normalizing the distribution that gives rise to the confidence interval.

$$t_\Delta = \frac{\mu_\Delta - H_0}{SE_\Delta} = \frac{(\mu_1 - \mu_2) - H_0}{\sqrt{\sigma_1^2/n_1 + \sigma_2^2/n_2}}$$

Notice how the numerator is simply the difference between the observed average difference and the null hypothesis. This means that, if the null was true, the average of this numerator would be zero: $E[\mu_\Delta - H_0] = 0$. The denominator is simply the standard error, which normalizes the statistic to have unit variance. It ensures that t_Δ follows a standard normal distribution, if the null is true. Since t_Δ is normally distributed under the null, values above or below the 1.96 should be extremely unlikely (appear less than 95% of the time). This means you can also reject the null hypothesis if you see such an extreme t-statistics. In our running example, the statistics associated with H_0 of no effect is greater than 2, meaning you can reject it at a 95% confidence level.

```
t_stat = (diff_mu - 0) / diff_se
t_stat
```

Additionally, since t-statistic is normally distributed under the null, you can use it to easily compute P-values.

P-values

Previously, I've said that there is less than a 5% chance we would observe such an extreme difference if the difference in conversion between the no email and the short email groups were actually zero. But can you precisely estimate what that chance is? How likely are you to observe such an extreme value? Enter p-values!

Like with confidence intervals (and most frequentist statistics, as a matter of fact), the true definition of p-values can be very confusing. So, to not take any risks, I'll copy the definition from Wikipedia: "the p-value is the probability of obtaining test results at least as extreme as the results actually observed during the test, assuming that the null hypothesis is correct".

To put it more succinctly, the p-value is the probability of seeing such data, if the null hypothesis were true. It measures how unlikely is that measurement you are seeing, considering that the null hypothesis is true. Naturally, this often gets confused with the probability of the null hypothesis being true. Note

the difference here. The p-value is NOT $P(H_0|data)$, but rather $P(data|H_0)$.

To get the p-value, all you need to do is compute the area under the standard normal distribution before the test-statistic for a one sided null hypothesis ("the difference is greater than x" or "the difference is smaller than x") and multiply the result by 2 for a two sided null hypothesis ("the difference is x").

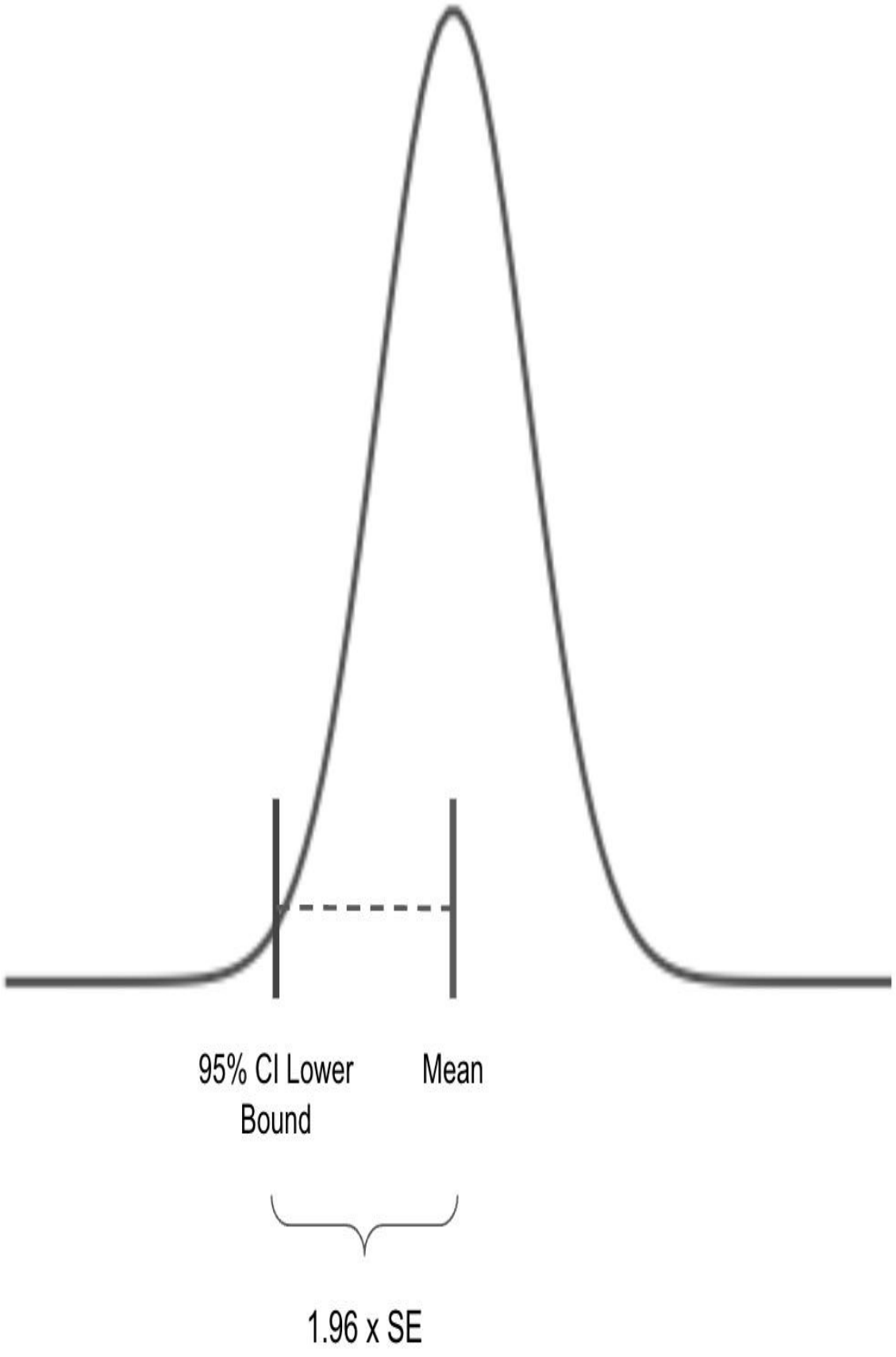
```
print("P-value:", (1 - stats.norm.cdf(t_stat))*2)
```

Notice how the p-value is interesting because it frees you from having to specify a confidence level, like 95% or 99%. But, if you wish to report one, from the p-value, you know precisely at which confidence our test will pass or fail. For instance, with a p-value of 0.025, you'll have significance up to the 2.5% level. So, while the 95% CI for the difference will not contain zero, the 99% CI will. This P-value also means that there is only a 2.5% chance of observing this extreme test statistic, if the difference was truly zero.

Power

So far, you've been looking into these statistical concepts from the perspective of a data analyst who's been presented with the data from an existing test. You are treating the data as given. But what if you are asked to design an experiment, instead of just reading one that was already designed? In this case, you need to decide the sample you would like to have for each variant. For example, what if you haven't yet run the cross sell email experiment, but, instead, you need to decide how many customers you should send the long email and how many, the short email and no email at all? From this perspective, the goal is to have a big enough sample so that you can correctly reject the null hypothesis of no effect, if it is indeed false. The probability that a test correctly rejects the null hypothesis is called Power of the test. It's not only a useful concept if you want to figure out the sample size you need for an experiment, but also for detecting issues in poorly run experiments.

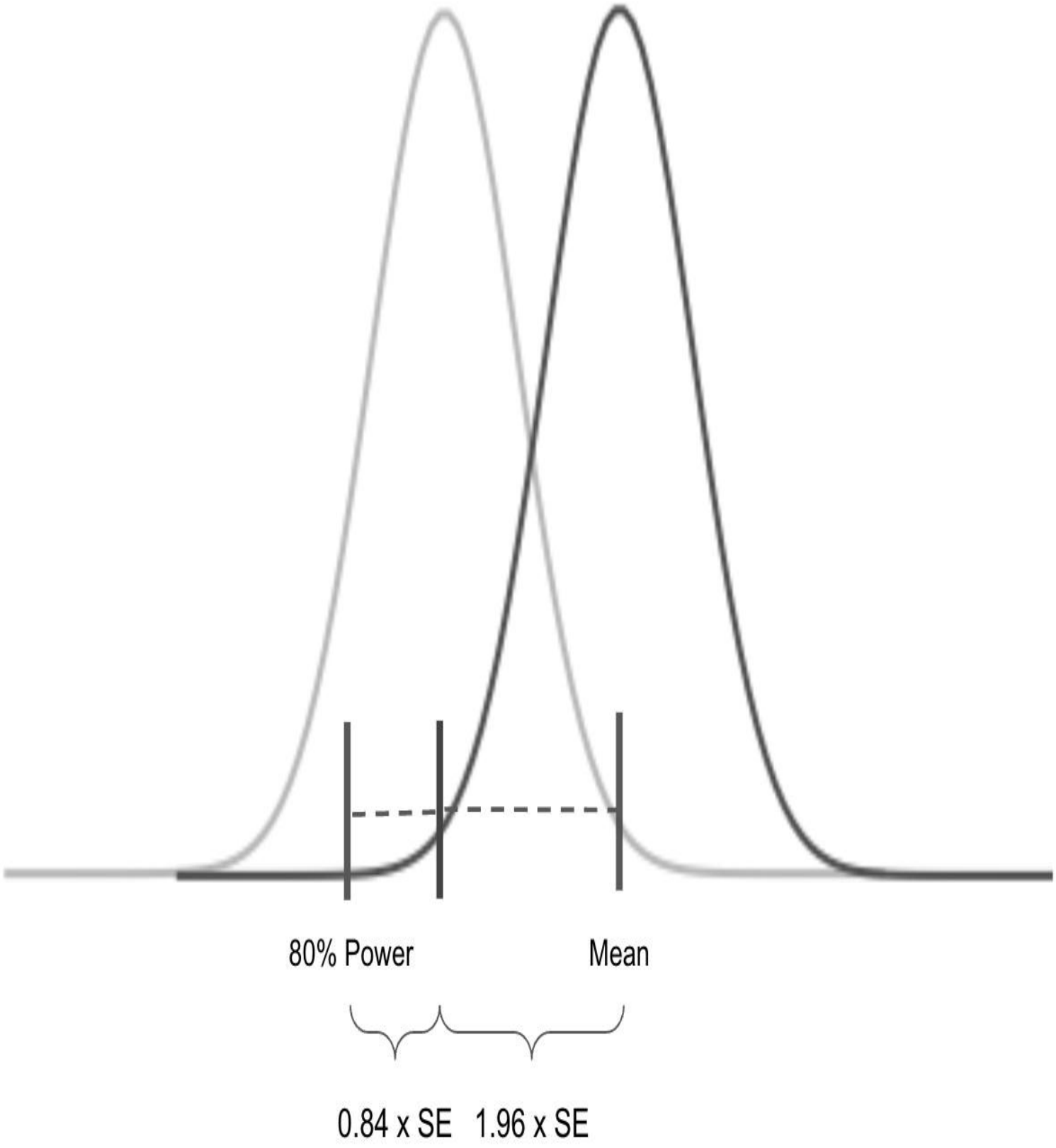
Power is closely related to statistical significance, which is sometimes referred to as $1-\alpha$. α is the chance of rejecting the null hypothesis when it is actually true. In other words, it is the false positive rate. You'll usually use $\alpha = 0.05$, which is why you tend to construct 95% confidence intervals. Recall how the 95% confidence interval means that 95% of the experiments will contain the true difference. This also means that 5% of them won't, which will cause you to falsely reject the null hypothesis 5% of the time. With $\alpha = 0.05$, you need the difference to be $1.96SE$ away from zero in order to conclude that it is statistically significant, since $\delta - 1.96SE$ will be the lower end of the 95% confidence interval.



Ok, so you need $\delta - 1.96SE > 0$ to claim the result as significant. But how likely are you to see this significant difference? This is where you need to think about Power. Power is the chance of correctly rejecting the null, or $1 - \beta$, with β being the probability of not rejecting the null when it is false (probability of false negative). The industry standard for power is 80%, meaning that you'll only have a

20% ($\beta = 0.2$) chance of not rejecting the null, when it is indeed false. To achieve 80% power, you need to reject the null hypothesis 80% of the time it is false. Since rejecting the null means that $\delta - 1.96SE > 0$, you need to get this big difference 80% of the time. In other words, you need to get the lower end of the 95% CI above zero, 80% of the time.

What is striking (or not) is that the lower end of the 95% confidence interval also follows a normal distribution. Just like the distribution of the sample average, the distribution of the lower end of the 95% CI has variance equal to the SE, but now the mean is $\delta - 1.96SE$. This is just the distribution of the sample average, shifted by $1.96SE$. This means that, in order to have $\delta - 1.96SE > 0$ 80% of the time (80% Power), you need the difference to be $(1.96 + 0.84)SE$ above zero. 1.96 to give you the 95% CI and 0.84 so that the lower end of this interval falls above zero 80% of the time.




```
stats.norm.cdf(0.84)
```

Sample Size Calculation

Another way of looking at this is that the detectable effect is $2.8SE = (1.96SE + 0.84SE)$. So, if you want to craft a cross sell email experiment where you want to detect a 1% difference, you must have a sample size which gives you at least $1\% = 2.8SE$. If you open up the SE formula for the

difference, you have $SE_\Delta = \sqrt{SE_1^2 + SE_2^2}$. But recall that you are now speaking from the perspective of an analyst who has not seen the experiment, but is actually trying to design it. This means you don't have the SE of the treated group, but you can assume that the variance in both treated and control will be the same and hence $SE_\Delta = \sqrt{2SE^2} = \sqrt{2\sigma^2/n} = \sigma\sqrt{2/n}$. Plugging this in the detectable difference, you end up with a pretty simple formula for determining the sample size of each variant in a test if you want 80% power and 95% significance:

$$\delta = 2.8\sigma\sqrt{2/n}$$

$$n = 2 * 2.8^2\sigma^2/\delta^2 \approx 16\sigma^2/\delta^2$$

where δ is the detectable difference and I've round $2 * 2.8^2$ to be conservative. Applying this formula to your data, using the variance of the control group as our best guess for σ^2, you end up with the following required sample size

> **SEE ALSO**
>
> This very simple way of calculating sample size was taken from *A/B Testing Intuition Busters: Common Misunderstandings in Online Controlled Experiments* (2022), by Ron Kohavi, Alex Deng and Lukas Vermeer. This sample size formula is only one of the many very interesting and useful things presented in the article, so I definitely recommend you check it out

```
np.ceil(16 * no_email.std()**2/0.01)
```

```
data.groupby("cross_sell_email").size()
```

This is of course invaluable in terms of experiment design, but is also good news for the cross sell experiment we currently have. In it, we have more than 100 samples for both treatment groups and 94 samples for the control, which indicates a properly powered test.

Key Ideas

The idea of this chapter was to link causal identification with estimation (and also review some important statistical concepts). Recall that the goal of causal inference is to learn about causal quantities from data. The first step in the process is identification, where you use key assumptions to go from unobservable causal quantities to observable statistical quantities you can estimate from data.

For example, the ATE is a causal quantity, since it is defined by the unobservable potential outcomes $ATE = E[Y_1 - Y_0]$. In order to identify the ATE, you use the independence assumption, $T \perp (Y_0, Y_1)$, which allows you to write it in terms of observable quantities, $E[Y|T=1]$, and $E[Y|T=0]$. That is, under the independence assumption,

$$E\left[Y_1 - Y_0\right] = E\left[Y|T=1\right] - E\left[Y|T=0\right]$$

You also saw how you could use Randomized Control Trials (RCTs) to make this assumption more plausible. If you randomize the treatment, you are brute forcing it to be independent from the potential outcomes Y_t.

But identification is just the first step in causal inference. Once you are able to write the causal quantities in terms of statistical quantities, you still need to estimate those statistical quantities. For instance, even though you can write the ATE in terms of $E[Y|T=1]$ and $E[Y|T=0]$, you still need to estimate them. You can do this by estimating the expectations with the empirical average in the data. But since you don't have infinite data, there is always some uncertainty in this estimation. If you want to use the ATE to make some decision, as it is often the case, you need to know how confident you are about its estimate. You can quantify this uncertainty by placing confidence intervals around your estimates. For example, say you estimate the average effect of sending a cross sell email to your customers in terms of conversion to a new product. You then find that effect to be positive. Still, there is always a possibility that this positive result was simply because, by pure chance, more customers that would convert anyways fell in the group that got the cross sell email, compared to the control group. Statistical tools help you deal with that chance, be it by using concervative levels of α (chance of a test finding a difference when there is none) when reading the test and by designing tests which are properly powered.

Other Examples

These examples are used to solidify what you learned in this chapter. They use concepts and techniques you just learned and are presented in a fast paced manner. As you read through them, try to link what they tell you with what you learned in this chapter. Also, use them to try to generalize the ideas outlined here to applications in your own work.

The Effectiveness of COVID19 Vaccines

Randomized Control Trials are incredibly important for the pharmaceutical industry. Perhaps the most widely known examples are the test conducted in order to determine the effectiveness of COVID19 vaccines, given the tremendous impact those had on almost everyone on the planet. Here is the result section from the study *Efficacy and Safety of the mRNA-1273 SARS-CoV-2 Vaccine*, published in 2020:

> *The trial enrolled 30,420 volunteers who were randomly assigned in a 1:1 ratio to receive either vaccine or placebo (15,210 participants in each group). More than 96% of participants received both injections, and 2.2% had evidence (serologic, virologic, or both) of SARS-CoV-2 infection at baseline. Symptomatic Covid-19 illness was confirmed in 185 participants in the placebo group (56.5 per 1000 person-years; 95% confidence interval (CI), 48.7 to 65.3) and in 11 participants in the mRNA-1273 group (3.3 per 1000 person-years; 95% CI, 1.7 to 6.0); vaccine efficacy was 94.1% (95% CI, 89.3 to 96.8%; P<0.001).*

Here is my take on how to interpret these results in the light of the concepts you've been learning. First, they defined the treatment and control (placebo) groups, saying that the treatment was randomly assigned, which ensures independence of the treatment from the potential outcomes. This will allow them to identify the causal effect of the vaccine from statistical the quantities $E[Y|T=0]$ and $E[Y|T=1]$. Next, they define the outcome as being the presence of symptomatic Covid-19 per 1000

person-years. Finally, they report the 95% CI for the estimate of $E[Y|T=0]$ and $E[Y|T=1]$ as being 48.7 to 65.3 and 1.7 to 6.0 respectively. This tells you that symptomatic Covid-19 was detected far less in those with the vaccine, compared to those that got the placebo. They report the efficacy of the vaccine, $E[Y|T=0]/E[Y|T=1]$, as well as the 95% confidence interval around it, 89.3 to 96.8%.

Face-to-Face vs Online Learning.

Besides the direct impact of the virus, the 2020 pandemic brough other important issues with it. Chief among those was the fact that kids could not go to school, so learning was taken to the online environment for as much as two years. It's hard to estimate the generational impact this will have, since the decision to come back from the online environment to a face-to-face one was not randomized. In Brazil, for example, public schools took longer to open up, compared to private schools.

However, although certainly not trivial, one can design an experiment to test the impact of online versus face-to-face learning, and it is precisely what was done in a study by Figlio, Rush, and Yin (2013). Here is the abstract of their paper, *Is it Live or is it Internet? Experimental Estimates of the Effects of Online Instruction on Student Learning*

> *Students in a large introductory microeconomics course at a major research university were randomly assigned to live lectures versus watching these same lectures in an internet setting, where all other factors (e.g., instruction, supplemental materials) were the same. Counter to the conclusions drawn by a recent U.S. Department of Education meta-analysis of non-experimental analyses of internet instruction in higher education, we find modest evidence that live-only instruction dominates internet instruction.These results are particularly strong for Hispanic students, male students, and lower-achieving students.We also provide suggestions for future experimentation in other settings.*

Notice that this study was conducted at a University in the US. Which means it is hard to say those results will generalize to basic education and to other countries. In technical terms, we say that the study has internal validity, as treatment and control groups are comparable due to randomization. But this study might not have external validity in terms of generalizing its results to other settings, since the people in it were not a random sample of the population, but rather economics students from a US university.

Chapter 3. Graphical Causal Models

A NOTE FOR EARLY RELEASE READERS

With Early Release ebooks, you get books in their earliest form—the author's raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the 3rd chapter of the final book. Please note that the GitHub repo will be made active later on.

If you have comments about how we might improve the content and/or examples in this book, or if you notice missing material within this chapter, please reach out to the editor at *vwilson@oreilly*.

In chapter 1 you saw how causal inference can be broken down into two problems: identification and estimation. In this chapter, you'll dive deeper into the identification part, which is arguably the most challenging one. This chapter is mostly theoretical, as you will be playing with graphical models without necessarily estimating them with data. Don't let this fool you. Identification is the heart of causal inference, so learning its theory is fundamental for tackling causal problems in real life. In this chapter, you will

- Introduction to graphical models, where you will learn what is a graphical model for causality, how associations flows in a graph and how to query a graph using off-the-shelf software
- Revisit the concept of identification through the lens of graphical models
- Discuss two very common sources of bias that hinder identification, their causal graph structure and what you can do about them.

Thinking About Causality

Have you ever noticed how those cooks in YouTube videos are excellent at describing food? "Reduce the sauce until it reaches a velvety consistency". If you are just learning to cook, you have no idea what this even means. Just give me the time I should leave this thing on the stove, will you! With causality, it's the same thing. Suppose you walk into a bar and hear folks discussing causality (probably a bar next to an economics department). In that case, you will hear them say how the confounding of income made it challenging to identify the effect of immigration on that neighborhood unemployment rate, so they had to use an instrumental variable. And by now, you might not understand what they are talking about. That's OK. You've only scratched the surface when it comes to understanding the language of causal inference. You've learned a bit about counterfactual outcomes and biases. Enough so you could understand the key issue causal inference is trying to solve. Enough to appreciate what's going on behind the most powerful tool of causal inference: Randomized Controlled Trials. But this tool won't always be available or simply won't work (as you'll soon see in the selection bias section). As you

encounter more challenging causal inference problems, you'll also need a broader understanding of the causal inference language, so you can properly understand what you are facing and how to deal with it.

A well articulated language allows you to think clearly. This chapter is about broadening your causal inference vocabulary. You can think of graphical models as one of the fundamental languages of causality. It is a powerful way of structuring a causal inference problem and making identification assumptions explicit, or even veisual. Graphical models will allow you to make your thoughts transparent to others and to yourself.

> **STRUCTURAL CAUSAL MODEL**
>
> Some scientists use the term structural causal model (SCM) to refer to a unifying language of causal inference. These models are composed of graphs and causal equations. Here, I'll mostly focus on the graph aspect of SCMs.

As a starting point into the fantastic world of graphs, let's take our previous example of estimating the impact of emails on conversion. In that example, the treatment T is cross sell email and the outcome Y is if a customer converted to a new product or not.

	gender	cross_sell_email	age	conversion
0	0	short	15	0
1	1	short	27	0
2	1	long	17	0
3	1	long	34	0
4	1	no_email	14	0

Let's also recall from the previous chapter that, in this problem, T is randomized. Hence, you can say that the outcome is independent from the potential outcomes, $(Y_0, Y_1) \perp T$, which makes association equal to causation:

$$E[Y_1 - Y_0] = E[Y|T = 1] - E[Y|T = 0]$$

Importantly, there is absolutely no way of telling that the independence assumption holds just by looking at the data. You can only say that it does because you have information about the treatment assignment mechanism. That is, you know that emails were randomized.

Visualizing Causal Relationships

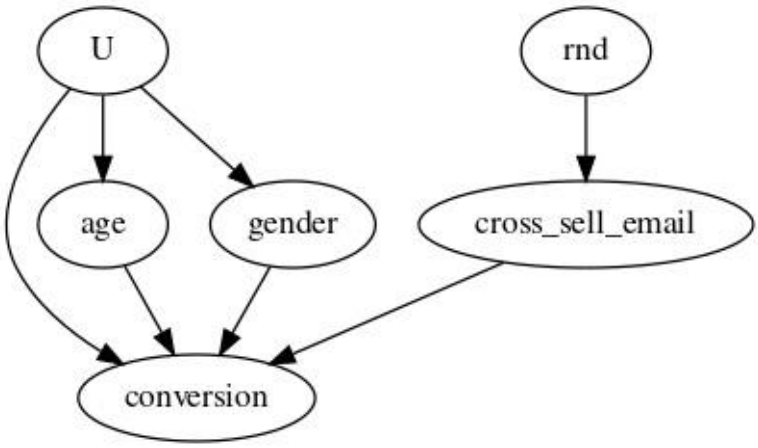
You can encode this knowledge in a graph, which captures your beliefs about what causes what. In this simple example, let's say you believe that cross sell emails cause conversion. You also believe that the

other variables you measured, age and gender, also cause conversion. Moreover, you can also add variables you didn't measure to the graph. We usually denote them by the letter U, since they are unobserved. There are probably many unobserved variables that cause conversion (like customer income, social background), gender (complex biological mechanism, culture) and age (how our product appeals to different demographics, the city the company is operating in). But since you don't measure them, you can bundle everything into a U node which represents all that unmeasured stuff. Finally, you can add a random node pointing to T, representing our knowledge of the fact that the cross email was randomized.

> **NOTE**
>
> You might find people referring to causal graphs as **DAGs**. The acronym stands for Directed Acyclic Graph. The directed part tells you that the edges have a direction, as opposed to undirected graphs, like a social network, for example. The acyclic tells you that the graph has no loops or cycles. Causal graphs are usually directed and acyclic because causality is non reversible.

To add of those beliefs you have to a graph and literally see them, you can use `graphviz`:

```
import graphviz as gr

g_cross_sell = gr.Digraph()

g_cross_sell.edge("U", "conversion")
g_cross_sell.edge("U", "age")
g_cross_sell.edge("U", "gender")

g_cross_sell.edge("rnd", "cross_sell_email")
g_cross_sell.edge("cross_sell_email", "conversion")
g_cross_sell.edge("age", "conversion")
g_cross_sell.edge("gender", "conversion")

g_cross_sell
```

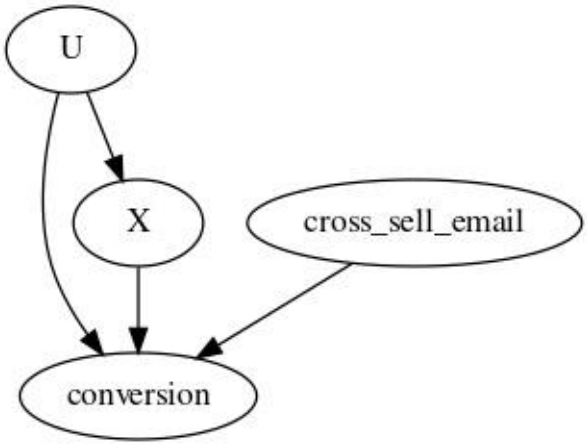


Each node in the graph is a random variable. You can use arrows, or edges, to show if a variable causes another. In the graphical model above, you are saying that email causes conversion, that U causes age, conversion and gender and so on and so forth. This language of graphical models will help us clarify your thinking about causality, as it makes your beliefs about how the world works explicit. If you are pondering how impractical this is, after all, there is non way you are going to encode all the hundreds of

variables that are commonly present in today's data application, rest assured you won't need to. In practice, you can radically simplify things, by bundling up nodes, while also keeping the general causal story you are trying to convey. For example, you can take the graph above and bundle the observable variables into a X node. Since they both are caused by U and cause conversion, your causal story remains intact by joining them.

Also, when you are representing variables that have been randomized or intervened on, you can just remove all incoming arrows from it.

```
g_cross_sell = gr.Digraph()

g_cross_sell.edge("U", "conversion")
g_cross_sell.edge("U", "X")

g_cross_sell.edge("cross_sell_email", "conversion")
g_cross_sell.edge("X", "conversion")

g_cross_sell
```

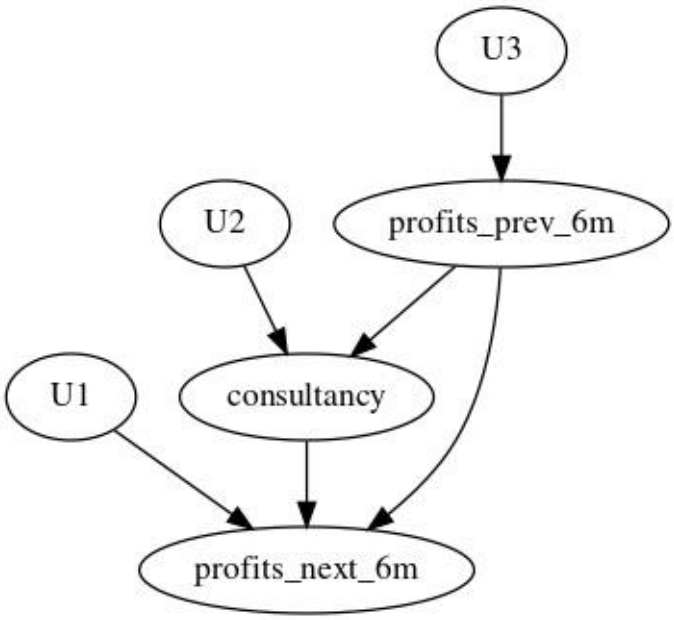


Just like with every language you learn, you are probably looking into this and thinking it doesn't all make complete sense. That's normal. I could just throw at you a bunch of rules and best practices to represent causal relationships between variables in a graph. But that is probably the least efficient way of learning. Instead, my plan is to simply expose you to lots and lots of examples and encourage you to try to use them for yourself. With time, you will get the hang of it. For now, I just want you to keep in mind that graphs are a very powerful tool for understanding why association isn't causation.

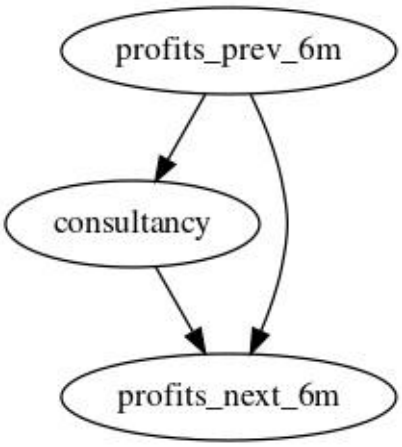
Are Consultants Worth it?

To see that, let's consider a more interesting example, where the treatment is not randomized. Let's suppose you are the manager of a company contemplating the decision of whether to bring in some top notch consultants. You know that they are expensive, but you also know that they have expert knowledge from working with the best companies in the business. To make things more complicated, you are not sure if the top notch consultants will improve your business or if it is just the case that only very profitable businesses can afford those consultants, which is why their presence correlates with strong business performance. It would be awesome if someone had randomized the presence of consultants, as this would make answering the question trivial. But of course you don't have that luxury, so you will have to come up with something else. As you can probably see by now, this is a problem of

untangling causation from association. To understand it, you can encode your beliefs about its causal mechanisms in a graph.



Notice how I've added U nodes to each of these variables to represent the fact that there are other things we can't measure causing them. Since graphs usually represent random variables, it is expected that a random component will cause all the variables, which is what those U represent. However, they won't add anything to the causal story I'm going to tell, so I might just as well omit them.



Here, I'm saying that the past performance of a company causes the company to hire a top notch consultant. If the company is doing great, it can afford to pay the expensive service. If the company is not doing so great, it can't. Hence, past performance (measured here by past profits), is what determines the odds of a company hiring a consultant. Remember that this relationship is not necessarily deterministic. I'm are just saying that companies that are doing well are **more likely** to hire top notch consultants.

Not only that, companies that did well in the past 6 months are very likely to also do well in the next 6 months. Of course, this doesn't always happen, but, on average, it does, which is why you also have an edge from past performance to future performance. Finally, I've added an edge from consultancy to the firm's future performance. Our goal is to know the strengths of this connection. This is the causal relationship I care about. Does consultancy actually cause company performance to increase?

Answering this question is not straightforward because there are two sources of association between consultancy and future performance. One is causal and the other is not. But in order to understand and

untangle them, you first need to take a quick look at how association flows in causal graphs.

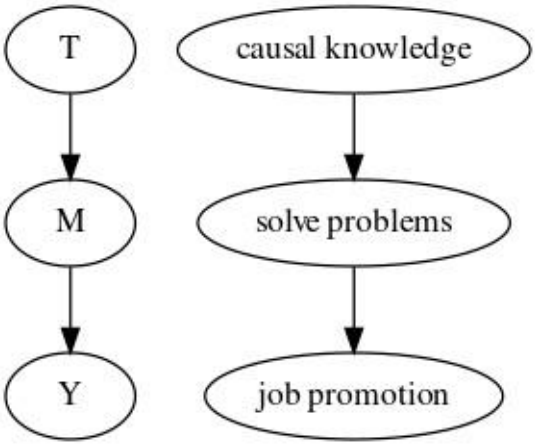
Crash Course in Graphical Models

Some say that graphical models are composed of an indefinite and perhaps infinite number of nodes, edges, paths and rules about those paths. Galleries, with vast air shafts between, surrounded by very low railings, through where passes a spiral stairway, which sinks abysmally and soars upwards to remote distances. It is not uncommon for man and woman to gaze in wonder at it, often getting lost to the point where there is no return. To find what you are looking for in a graph and to use it for your needs can be an art; I will now teach you that art and, hopefully, by finding the associations you need, you will also end up finding yourself.

There are whole semesters on graphical models. By all means, if you want to go deep in graphical models, it will be very beneficial for your understanding of causal inference. But, for the purpose of this book, it is just (utterly) important that **you understand what kind of independence and conditional independence assumptions a graphical model entails**. As you'll see, associations flow through a graphical model as water flows through a stream. You can stop this flow or enable it, depending on how you treat the variables in the graph. To understand this, let's examine some common graphical structures and examples. They will be pretty straightforward, but they are the sufficient building blocks to understand everything about the flow of association, independence and conditional independence on graphical models.

Chains

First, look at this very simple graph. It's called a chain. Here T causes M which causes Y. You can sometimes refer to the intermediary node as a mediator, because it mediates the relationship between T and Y.



In this first graph, although causation only flows in the direction of the arrows, association flows both ways. To give a more concrete example, let's say that knowing about causal inference improves your problem solving skills, and problem solving increases your chances of getting a promotion. So causal knowledge causes your problem solving skills to increase, which in turn causes you to get a job promotion. You can say here that job promotion is dependent on causal knowledge. The greater the causal expertise, the greater your chances of getting a promotion. Also, the greater your chances of promotion, the greater your chance of having causal knowledge. Otherwise, it would be difficult to get a

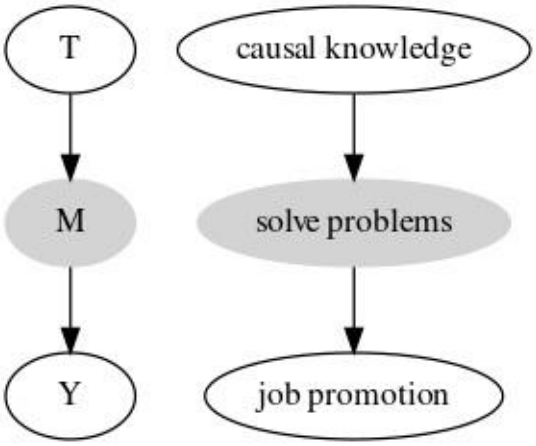
promotion. In other words job promotion is associated with causal inference expertise the same way that causal inference expertise is associated with job promotion, even though only one of the directions is causal. When two variables are associated with each other, you can say they are dependent or not independent.

$$T \neg \perp Y$$

Now, let's hold the intermediary variable fixed. You could do that by looking only at people with the same M, or problem solving skills in our example. Formally, you can say you are conditioning on M. In this case, the dependence is blocked. So, T and Y are independent given M. You can write this mathematically as

$$T \perp Y | M$$

To indicate that we are conditioning on a node, I'll fill it with gray.



To see what this means in our example, think about conditioning on people's problem solving skills. If you look at a bunch of people with the same problem solving skills, knowing which ones are good at causal inference doesn't give any further information about their chances of getting a job promotion. In mathematical terms,
$E[Promotion|Solve\ problems, Causal\ knowledge] = E[Promotion|Solve\ problems$
. The inverse is also true; once I know how good you are at solving problems, knowing about your job promotion status gives me no further information about how likely you are to know causal inference.

As a general rule, if you have a chain like in the graph above, association flowing in the path from T to Y is blocked when you condition on an intermediary variable M. Or,

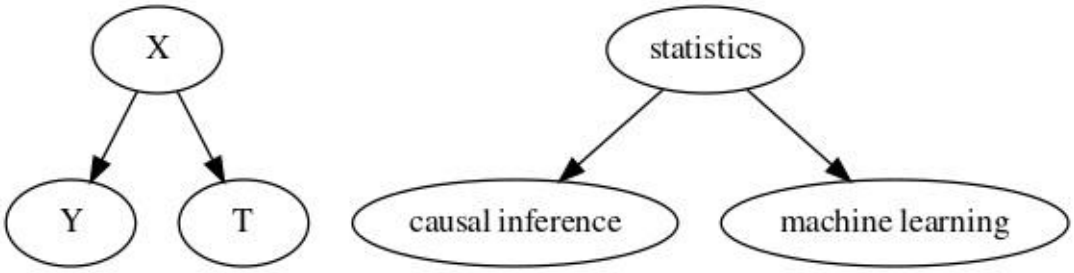
$T \neg \perp Y$

but

$T \perp Y | M$

Forks

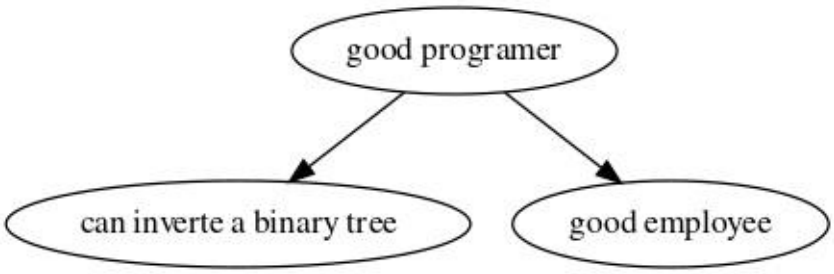
Moving on, let's consider a fork structure. In this structure, you have a common cause: the same variable causes two other variables down the graph. In forks, association flows backwards through the arrows.



For example, let’s say your knowledge of statistics causes you to know more about causal inference and about machine learning. However, knowing causal inference doesn’t help you with machine learning and vice versa, so there is no edge between those variables.

This graph is telling you that if you don’t know someone’s level of statistical knowledge, then knowing that they are good at causal inference makes it more likely that you are also good at machine learning, even if causal inference doesn’t help you with machine learning. That is because even if you don’t know someone’s level of statistical knowledge, you can infer it from your causal inference knowledge. If they are good at causal inference, they are probably good at statistics, making it more likely that they are good at machine learning. The variables at the tip of a fork move together even if they don’t cause each other, simply because they are both caused by the same thing. In the causal inference literature, when we have a common cause between a treatment and the outcome, we call that common cause a confounder.

The fork structure is so important in causal inference that it deserves another example. Do you know how tech recruiters sometimes ask you to solve problems that you’ll probably never find in the job you are applying for? Like when they ask you to invert a binary tree or count duplicate elements in Python? Well, they are essentially leveraging the fact that association flows through a fork structure in the following way graph:



The recruiter knows that good programmers tend to be top performers. But when they interview you, they don’t know if you are a good programmer or not. So they ask you a question that you’ll only be able to answer if you are. That question doesn’t have to be about a problem which you’ll encounter in the job you are applying for. It just needs signals whether you are a good programmer or not. If you can answer the question, you will likely be a good programmer, which means you will also likely be a good employee.

Now, let’s say that the recruiter already knows that you are a good programmer. Maybe they know you from previous companies or you have an impressive degree. In this case, knowing if you can answer or not the application process questions gives no further information on whether you will be a good employee or not. In technical terms, you can say that answering the question and being a good employee are independent, once your condition on being a good programmer.

More generally, if you have a fork structure, two variables that share a common cause are dependent,

but independent when you condition on the common cause. Or

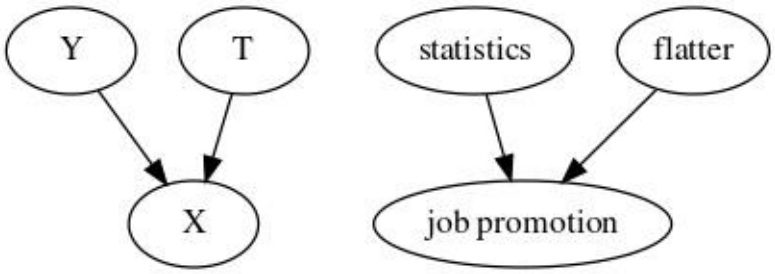
$$T \not\perp Y$$

but

$$T \perp Y | X$$

Immorality or Collider

The only structure that is missing is the immorality (and yes, this is a technical term). An immorality is when two nodes share a child, but there is no direct relationship between them. Another way of saying this is that two variables share a common effect. This common effect is often referred to as a collider, since two arrows collide at it.



In an immorality, the two parent nodes are independent of each other. But they become dependent if you condition on the common effect. For example, consider that there are two ways to get a job promotion. You can either be good at statistics or flatter your boss. If I don't condition on your job promotion, that is, I don't know if you will or won't get it, then your level of statistics and flattering are independent. In other words, knowing how good you are at statistics tells me nothing about how good you are at flattering your boss. On the other hand, if you did get a job promotion, suddenly, knowing your level of statistics tells me about your flattering level. If you are bad at statistics and did get a promotion, you will likely be good at flattering your boss. Otherwise, it will be very unlikely for you to get a promotion. Conversely, if you are good at statistics, it is likely that you are bad at flattering, as being good at statistics already explains your promotion. This phenomenon is sometimes called **explaining away**, because one cause already explains the effect, making the other cause less likely.

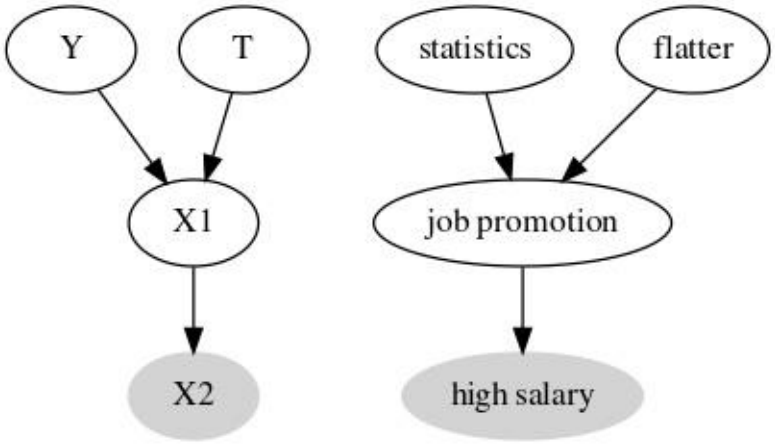
As a general rule, conditioning on a collider opens the association path, making the variables dependent. Not conditioning on it leaves it closed. Or

$$T \perp Y$$

and

$$T \not\perp Y | X$$

Importantly, we can open the same dependence path if instead of conditioning on the collider, you condition on a cause (direct or not) of the collider. Continuing with our example, let's now say that getting a job promotion massively increases your salary, which gives you the next graph.



In this graph, even if you don’t condition on the collider, but condition on a cause of it, the causes of the collider become dependent. For instance, even if I don’t know about your promotion, but I do know about your massive salary, your knowledge about statistics and boss flattering become dependent: having one makes it less likely that you also have the other.

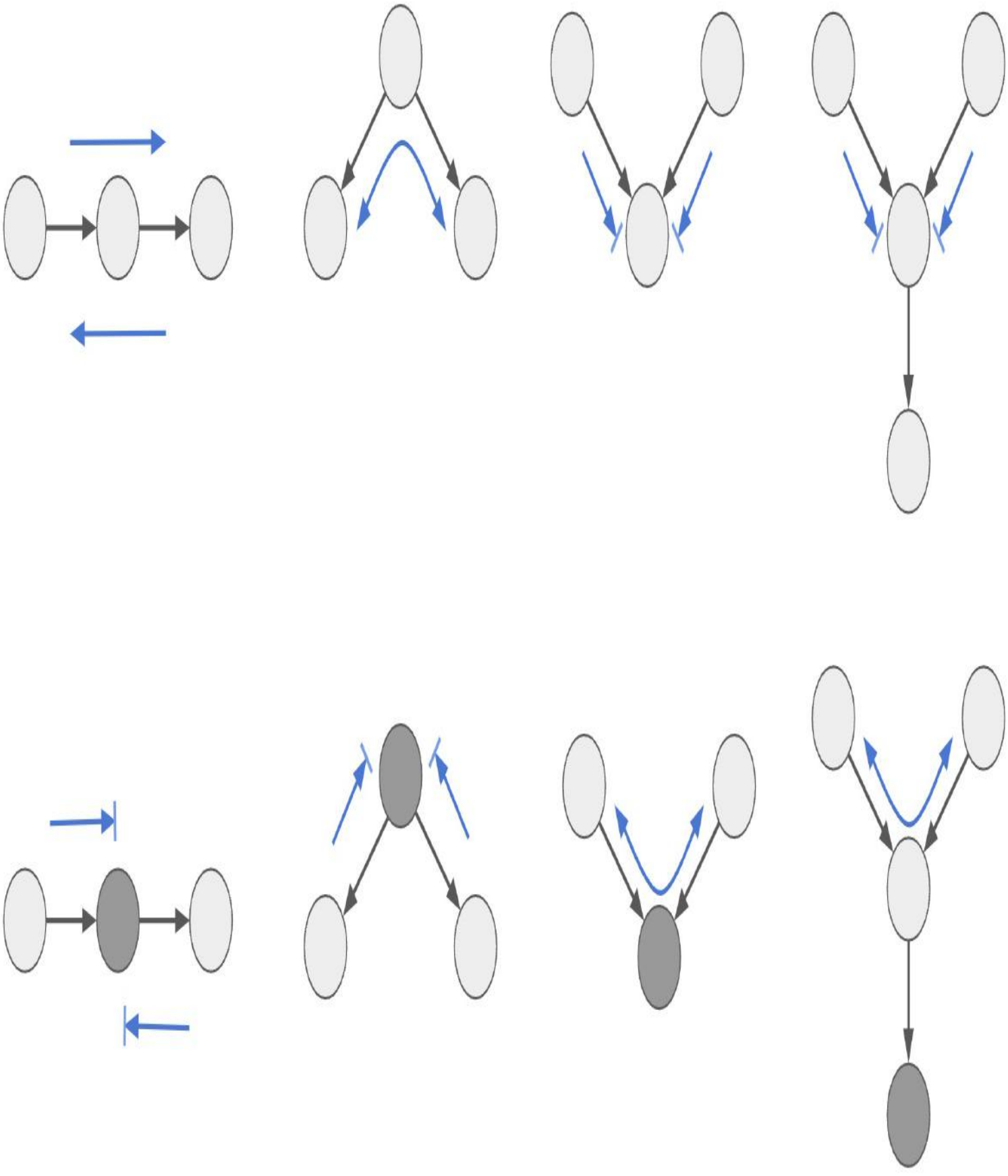
The Flow of Association Cheat Sheet

Knowing these three structures - chains, forks and immoralities - you can derive an even more general rule about independence and the flow of association in a graph.

A path is blocked if and only if:

1. It contains a non collider that has been conditioned on
2. It contains a collider that has not been conditioned on and has no descendants that have been conditioned on.

Here is a cheat sheet about how dependence flows in a graph.

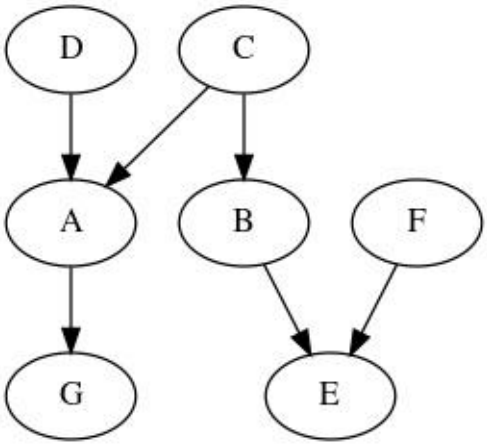


If these rules seem a bit opaque or hard to grasp, now is a good time for me to tell you that, thankfully, you can use off-the-shelf algorithms to check if two variables in a graph are associated with each other

or if they are independent. To tie everything you learned together, let's go over a final example so I can show you how to code it up.

Querying a Graph in Python

Take the following graph:



You can input such a graph into a `BayesianNetwork`, from `pgmpy`. `pgmpy` is a library to handle graphical models and has a bunch of handy algorithms that will help us inspect this graph.

> **NOTE**
>
> These graphical models are also called Bayesian networks. So when you see that name, you already know what it is about.

```
from pgmpy.models import BayesianNetwork

model = BayesianNetwork([
    ("C", "A"),
    ("C", "B"),
    ("D", "A"),
    ("B", "E"),
    ("F", "E"),
    ("A", "G"),
])
```

As a starter, let's take D and C. They form the immorality structure you saw earlier, with A being a collider. From the rule about independence in an immorality structure, you know that D and C are independent. You also know that if you condition on the collider A, association starts to flow between them. The method `is_dconnected` tells us if association flows between two variables in the graph (d-separation is another way of expressing the independence between two variables in a graph). To condition on a variable, you can add it to the observed set. For example, to check if D and C are dependent given A, you can use `is_dconnected` and pass `observed=["A"]`.

```
print("Is D and C dependent?", model.is_dconnected("D", "C"))
print("Is D and C dependent given A?", model.is_dconnected("D", "C", observed=["A"]))
```

```
Is D and C dependent? False
Is D and C dependent given A? True
```

Next, notice that D, A and G form a chain. You know that association flows in a chain, so D is not independent from G. However, if you condition on the intermediary variable A, you block the flow of association.

```
print("Is G and D dependent?", model.is_dconnected("G", "D"))
print("Is G and D dependent given A?", model.is_dconnected("G", "D", observed=["A"]))
```

```
Is G and D dependent? True
Is G and D dependent given A? False
```

The last structure you need to review is the fork. You can see that A, B and C form a fork, with C being a common cause of A and B. You know that association doesn't flow through a fork, so A and B are not independent. However, if you condition on the common cause, the path of association is opened and association starts to flow.

```
print("Is A and B dependent?", model.is_dconnected("A", "B"))
print("Is A and B dependent given A?", model.is_dconnected("A", "B", observed=["C"]))
```

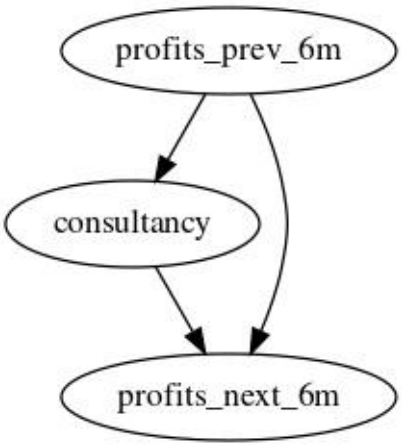
```
Is A and B dependent? True
Is A and B dependent given A? False
```

Finally, let's put everything together and talk about G and F. Does association flow between them? Let's start at G. You know that association flows between G and E, since they are in a fork. However, association stops at the collider E, which means that G and F are independent. However, if you condition on E, association starts to flow through the collider and the path opens, connecting G and F.

```
print("Is G and F dependent?", model.is_dconnected("G", "F"))
print("Is G and F dependent given E?", model.is_dconnected("G", "F", observed=["E"]))
```

```
Is G and F dependent? False
Is G and F dependent given E? True
```

This is great. Not only did you learn the three basics structures in graphs, you also saw how to use off-the-shelf algorithms to check for independences in the graph. But what does this have to do with causal inference? It's time to go back to the problem we were exploring at the beginning of the chapter. Recall that we were trying to understand the impact of hiring expensive, top notch consultants on business performance, which we depicted as the following graph:



You can use your newly acquired skills to see why association is not causation in this graph. Notice that you have a fork structure in this graph. Hence, there are two flows of association between consultancy and company's future performance: a direct causal path and a backdoor path, where association flows throw the common cause, past performance. The existence of this opened backdoor path is why, in this case, association is not causation.

Understanding how associations flow in a graph through non causal paths will allow you to be much more precise when talking about the difference between association and causation. For this reason, it pays to revisit the concept of identification, now under the new light of graphical models.

Identification Revisited

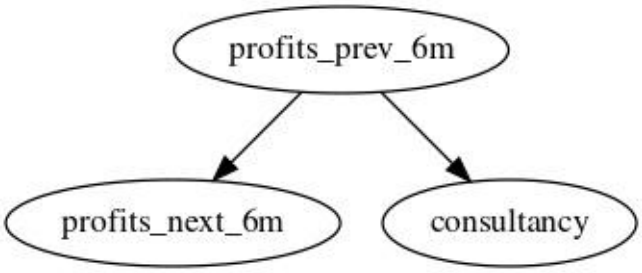
So far, in the absence of randomization, the argument I've been using to explain why it is so hard to find the causal effect is that treated and untreated are not comparable to each other. For example, companies that hire consultants usually have better past performance than those that don't hire expensive consultants. This results in the sort of bias you've seen before

$$E[Y|T=1] - E[Y|T=0] = \underbrace{E[Y_1 - Y_0|T=1]}_{ATET} + \underbrace{\{E[Y_0|T=1] - E[Y_0|T=0]\}}$$

Specifically, you can even say that $E[Y_0|T=1] > E[Y_0|T=0]$, since only companies that are doing well to begin with can afford expensive consultants.

Now that you've learned about causal graphs, you can be more precise about the nature of that bias and, more importantly, you can understand what you can do to make it go away. Identification is intimately related to independence in a graphical model. If you have a graph which depicts the causal relationship between the treatment, the outcome and other relevant variables, you can think about identification as the process of isolating the causal relationship between the treatment and the outcome in that graph. During the identification phase, you will essentially close all non-causal flows of association

Take our consultancy graph. As you saw earlier, there are two association paths between the treatment and the outcome, but only one of them is causal. You can check for bias by creating a causal graph which is just like our original one, but with the causal relationship removed. If treatment and outcome are still connected in this graph, it must be due to a non-causal path, which indicates the presence of bias.

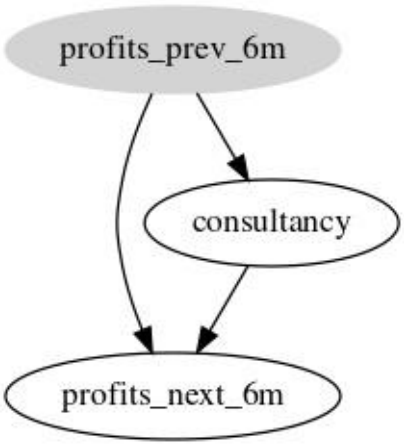



```
consultancy_model_severed = BayesianNetwork([
    ("profits_prev_6m", "profits_next_6m"),
    ("profits_prev_6m", "consultancy"),
#     ("consultancy", "profits_next_6m"), # causal relationship removed
])
```

```
consultancy_model_severed.is_dconnected("consultancy", "profits_next_6m")
```

```
True
```

These non causal flows of association are referred to as backdoor paths. To identify the causal relationship between T and Y, you need to close them so that only the causal path one remains. In the consultancy example, you know that conditioning on the common cause, company's past performance, closes that path.



CIA and The Adjustment Formula

You just saw that conditioning on `profits_prev_6m` blocks the non-association flow between the treatment, consultancy, and the outcome, company's future performance. This means that if you look at a group of companies with similar past performance and inside those groups compare the future performance of those that hired consultants with those that didn't, the difference can be entirely attributed to the consultants. This makes intuitive sense right? The difference in future performance between the treated (companies that hired consultants) and the untreated is 1) due to the treatment itself and 2) due to the fact that companies that hire consultants tend to be doing well to begin with. If you just compare treated and untreated companies that are doing equally well, the second source of difference disappears.

Of course, like with everything in causal inference, you are making an assumption here. Specifically, you are assuming that all sources of non-causal associated between the treated and the outcome is due to the common causes you can measure and condition on. This is very much like the independence assumption you saw earlier, but in its weaker form.

$$(Y_0, Y_1) \perp T|X$$

This **Conditional Independence Assumption** (CIA) states that, if you compare units (companies, in our example) with the same level of covariates X, their potential outcomes will be, on average, the same. Another way of saying this is that treatment looks **as if it was randomized**, if you look at units with the same values of covariate X.

> **NOTE**
>
> The CIA permeates a lot of causal inference research and it goes by many names, like ignorability and exogeneity.

The CIA also motivates a very simple way to identify the causal effect from observable quantities in the data. If treatment looks as good as random within groups of X, all you need to do is compare treated and untreated groups inside each of the X defined groups and average the result using the size of the group as weights.

$$ATE = E_X\left[E[Y|T=1] - E[Y|T=0]\right]$$

$$\begin{aligned} ATE &= \sum_x \left\{(E[Y|T=1, X=x] - E[Y|T=0, X=x])P(X=x)\right\} \\ &= \sum_x \left\{E[Y|T=1, X=x]P(X=x) - E[Y|T=0, X=x]P(X=x)\right\} \end{aligned}$$

This is called the **adjustment formula** or conditionality principle. It says that, if you condition on or control for X, the average treatment effect can be identified as the weighted average of in-group differences between treated and control. Again, if conditioning on X blocks the flow of association through the non-causal paths in the graph, a causal quantity, like the ATE, becomes identifiable, meaning that you can compute it from observable data.

Positivity Assumption

The adjustment formula also highlights the importance of positivity. Since you are averaging the difference between treatment and outcome over X, you must ensure that, for all groups of X, there are some units in the treatment and some in the control, otherwise the difference is undefined. More formally, you can say that you need that the the probability of the treatment given the covariates to be strictly positive and below 1: $1 > P(T|X) > 0$. Identification is still possible when positivity is violated, but it will require us to make dangerous extrapolations.

> **NOTE**
>
> Since the positivity assumption is also very popular in causal inference, it too goes by many names, like common support or overlap.

An Identification Example with Data

Since this might be getting a bit abstract, let's see how it all plays out with some data. To keep our example, let's say you've collected data on 6 companies, three of which had low profits (1 million USD) in the past six months and three of which had high profits. Just like you suspected, highly profitable companies are more likely to hire consultants. Two out of the three high profit companies

hired them, while only one out of the three low profit companies hired consultants (if the low sample bothers you, please pretend that each data point here actually represents ten thousand companies).

```
df
```

	profits_prev_6m	consultancy	profits_next_6m
0	1.0	0	1.0
1	1.0	0	1.1
2	1.0	1	1.2
3	5.0	0	5.5
4	5.0	1	5.7
5	5.0	1	5.7

If you simply compare `profits_next_6m` of the companies that hired consultants with those that didn't, you'll get a difference of 1.66 MM in profits. But you know better. This is not the causal effect of consultancy on a company's performance, since the companies which performed better in the past are over-represented in the group which hired consultants.

```
(df.query("consultancy==1")["profits_next_6m"].mean()
 - df.query("consultancy==0")["profits_next_6m"].mean())
```

```
1.666666666666667
```

To get an unbiased estimate of the effect of consultants, you need to look at companies with similar past performance. As you can see, this yields more modest results.

```
(df.query("consultancy==1").groupby("profits_prev_6m")["profits_next_6m"].mean() -
 df.query("consultancy==0").groupby("profits_prev_6m")["profits_next_6m"].mean())
```

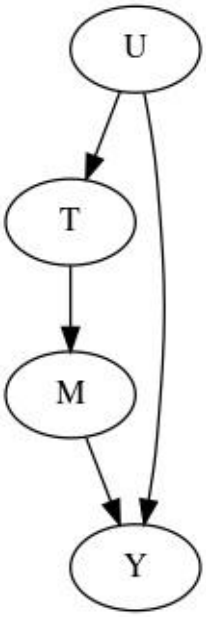
```
profits_prev_6m
1.0    0.15
5.0    0.20
Name: profits_next_6m, dtype: float64
```

If you take the weighted average of these effects, where the weights are the size of each group, you end up with an unbiased estimate of the ATE. Here, since the two groups are of equal size, this is just a simple average, giving us an ATE of 175 thousands. Hence, if you are a manager deciding whether to hire consultants and you are presented with the above data, you can conclude that the impact of consultants on future profits is about 175k USD. Of course, in order to do that, you have to invoke the CIA. That is, you have to assume that past performance is the only common cause of hiring consultants

and future performance.

FRONT DOOR ADJUSTMENT

The backdoor adjustment is not the only possible strategy to identify causal effects. One can leverage the knowledge of causal mechanisms to identify the causal effect via a front door, even in the presence of unmeasured common causes. With this strategy, you must be able to identify the effect of the treatment on a mediator and the effect of that mediator on the outcome. Then, the identification of the effect of treatment on the outcome becomes the combination of those two effects. However, it's hard to find an application where such a graph is plausible, which is why the front door adjustment is not so popular.



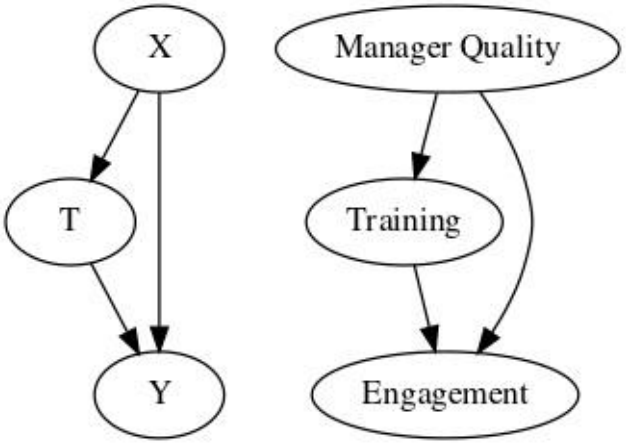
You just went through a whole example of encoding our beliefs about a causal mechanism into a graph and using that graph to find out which variables you needed to condition on in order to estimate the ATE, even without randomizing the treatment. Then, you saw what that looked like with some data, where you estimated the ATE, following the adjustment formula and assuming conditional independence. The tools used here are fairly general and will inform us of many causal problems to come. Still, I don't think we are done yet. Some graphical structures - and the bias they entail - are much more common than others. It is worth going through them so you can start to get a feeling for the difficulties that lie ahead in your causal inference journey.

Confounding Bias

The first significant cause of bias is confounding. It's the bias we've been discussing so far. Now, we are just putting a name to it. **Confounding happens when the treatment and the outcome share a common cause**. For example, let's say that you work in HR and you want to know if your new management training program is increasing employers' engagement. However, since the training is optional, you believe only managers that are already doing great attend the program and those who need it the most, don't. When you measure engagement of the teams under the managers that took the training, they are much higher than that of the teams under the managers who didn't attend the training. But it's hard to know how much of this is the causal effect of the training. Since there is a common cause between treatment and outcome (the training and being a good manager, respectively) they would

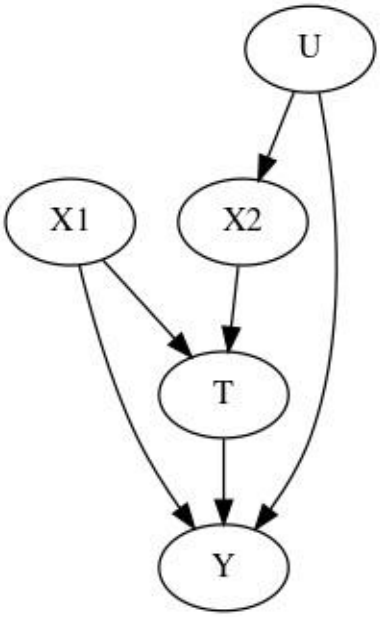
move together regardless of a causal effect.

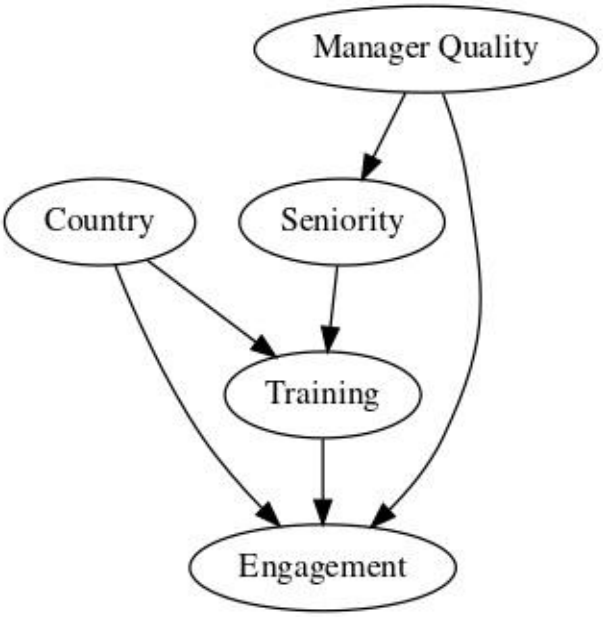
To identify that causal effect, you need to close all backdoor paths between the treatment and the outcome. If you do so, the only effect that will be left is the direct effect T->Y. In our example, you could somehow control for the manager's quality prior to taking the training. That is, you could compare teams whose managers are equally talented but some took the training and some didn't. In that situation, the difference in the outcome will be only due to the training, since manager quality prior to the training would be held constant between treatment and control. Simply put, **to adjust for confounding bias, you need to control for all common causes of the treatment and the outcome**.



Unfortunately, it is not always possible to control all common causes. Sometimes, there are unknown causes or known causes that you can't measure. The case of manager quality is one of them. Despite all the effort, we still haven't yet figured out how to measure management quality. If you can't observe manager quality, then you can't condition on it and the effect of training on engagement is not identifiable.

Now, notice that you don't need to measure all direct common causes in order to block all backdoor paths. For instance, let's suppose that your training is only being offered to managers at a certain seniority level and only in some countries in which our business operates. Also, let's assume manager quality doesn't cause managers to opt in for the training directly, but only indirectly, via the seniority level. That is, better managers achieve high seniority, which allows them to join the training. Also, let's assume that countries have equally good managers, so there is no direct link between the two variables. In this case, even if you can't control for the unmeasurable manager's quality, you can condition on seniority and country, which closes that backdoor path that contains manager quality.

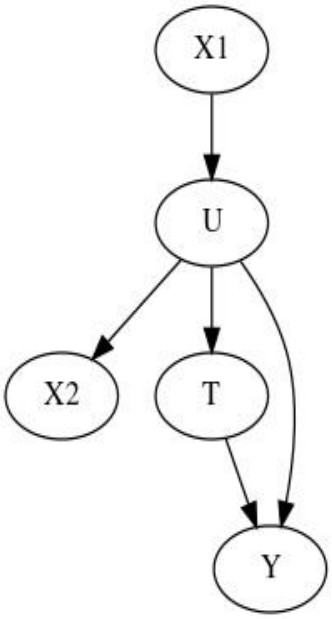


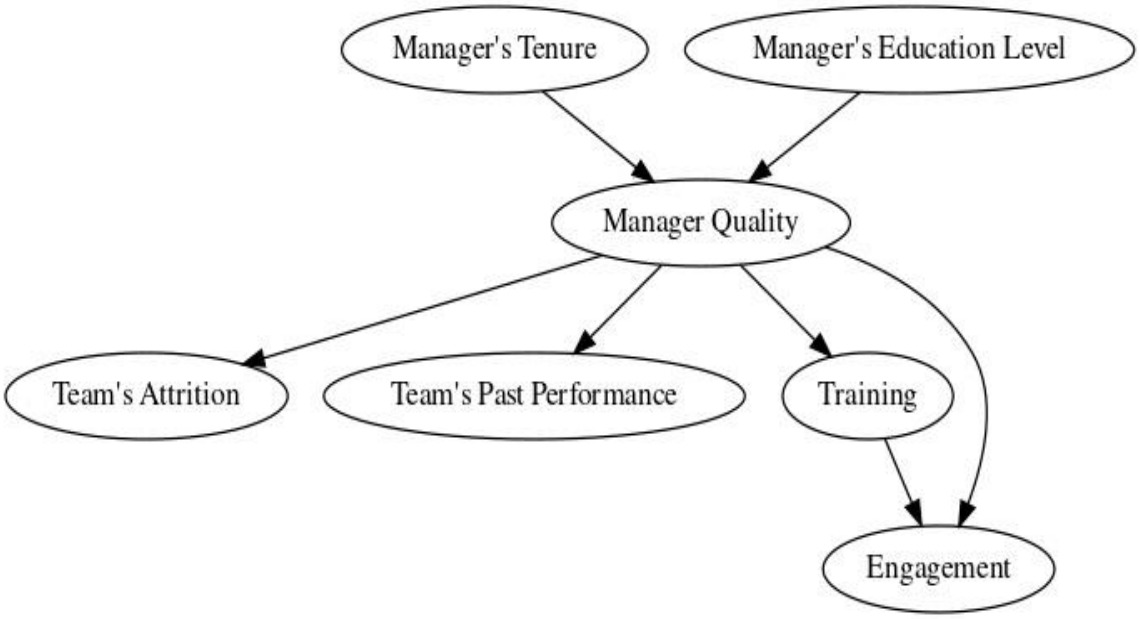


In general, even if you can't measure all common causes, you can still block backdoor paths between the treatment and the outcome by conditioning on measurable variables that mediate the effect of the unmeasured confounder on the treatment.

But what if that is not the case? What if the unmeasured variable causes the treatment and the outcome directly? In the following example, once again, manager quality causes managers to opt in for the training and team's engagement. So there is confounding in the relationship between the treatment (training) and the outcome (team's engagement). But in this case, you can't condition on the confounder, because it is unmeasurable. In this case, the causal effect of the treatment on the outcome is not identifiable due to confounder bias. However, you have other measured variables that can act as proxies for the confounder manager's quality. Those variables are not in the backdoor path, but controlling for them will help reduce the bias (even though it won't eliminate it). Those variables are sometimes referred to as surrogate confounders.

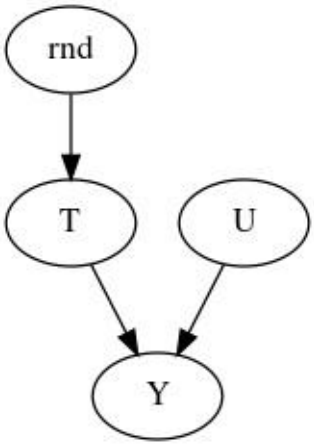
In this example, you can't measure manager's quality, but you can measure some of its causes - like the manager's tenure and manager's education degree - and some of its effects - like team's attrition or team's past performance. Controlling for those surrogate variables is not sufficient to eliminate bias, but it sure helps.





Randomization Revisited

In many important and very relevant research questions, confounders are a major issue, since you can never be sure that we've controlled for all of them. But if you are planning to use causal inference mostly in the industry, I have good news for you. In the industry, you are mostly interested in learning the causal effects of things that your company can control - like prices, customer service, marketing budget - so that you can optimize them. In those situations, it is fairly easy to know what the confounders are, because the business usually knows what information it used to allocate the treatment. Not only that, even when it doesn't, it's almost always an option to intervene on the treatment variable. This is precisely the point of A/B tests. When you randomize the treatment, you can go from a graph with unobservable confounders to one where the only cause of the treatment is randomness.



This means that, besides trying to see what variables you need to condition on in order to identify the effect, you should also be asking yourself what are the possible interventions you could make that would change the graph into one where the causal quantity of interest is identifiable.

I also want to mention that not all is lost, when you have unobserved confounders. In part IV, I'll cover methods that can leverage time structure in the data to deal with unobserved confounders. Part V will cover the use of instrumental variables for the same purpose. Moreover, when you can't measure all common causes, instead of simply giving up, it is often much more fruitful to shift the discussion from "am I measuring all confounders?" to "how strong should the unmeasured confounders be to change my analysis significantly?". This is the main idea behind sensitivity analysis.

Selection Bias

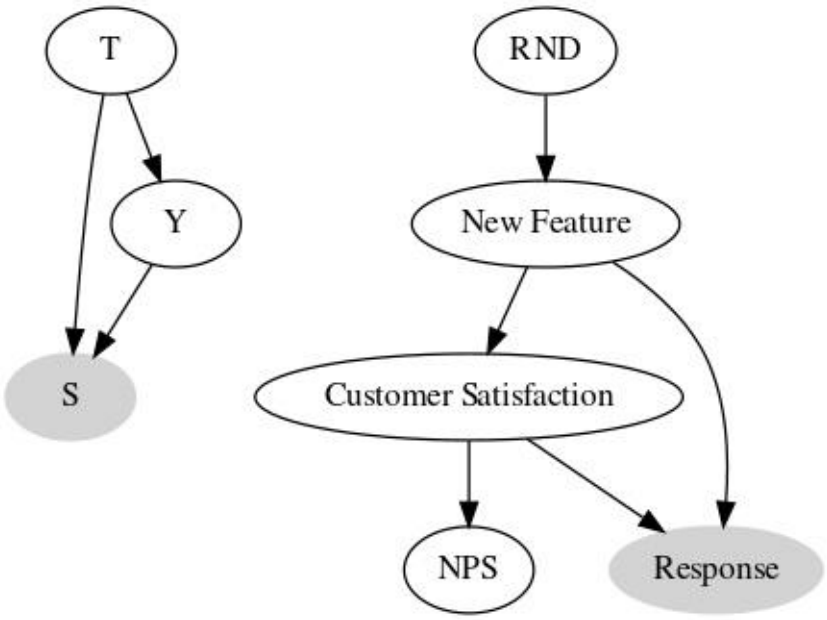
If you think confounding bias was already a sneaky little stone in your causal inference shoe, just wait until you hear about selection bias. While confounding bias happens when you don't control for common causes to the treatment and outcome, selection bias is more related to conditioning on common effects and mediators.

BIAS TERMINOLOGY

There isn't a consensus on the literature on the names of biases. For instance, economists tend to refer to all sorts of biases as selection bias. In contrast, some scientists like to further segment what I'm calling selection bias into collider bias and mediator bias. Here, I'll use the same terminology as in the book *Causal Inference: What If*, by Miguel A. Hernán and James M. Robins.

Conditioning on a Collider

Consider the case where you work as a software company and want to estimate the impact of a new feature you've just implemented. To avoid any sort of confounding bias, you do a randomized roll-out of the feature: 10% of the customers are randomly chosen to get the new feature, while the rest don't. You want to know if this feature made your customer happier and more satisfied. Since satisfaction isn't directly measurable, you use NPS (Net Promoter Score) as a proxy for it. To measure NPS, you send a survey to the customers in the roll-out (treated) and in the control groups, asking them if they would recommend your product. When the results arrive, you see that the customers who had the new feature and responded to the NPS survey had higher NPS score than the ones that didn't have the new feature and also responded to the NPS survey. Can you say that this difference is entirely due to the causal effect of the new feature on NPS? To answer this question, you should start with the graph that represents this situation.



To cut to the chase, sadly, the answer is no. The issue here is that you can only measure NPS for those that responded to the NPS survey. This means you are estimating the difference between treated and control while also conditioning on customers that responded to the NPS survey. Even though randomization allows you to identify the ATE as the difference in outcome between treated and control, once you condition on the common effect, you also introduce selection bias. To see this, you can recreate the graph above and delete the causal path from the new feature to customer satisfaction, which also closes the direct path to NPS. Then, you can check if NPS is still connected to the new features, once you condition on the response. You can see that it is, meaning that association flows between the two variables via a non causal path, which is precisely what bias means. Just keep in mind that selection bias is very sneaky. This approach of deleting the causal path and checking if the treatment and outcome

are still connected won't always work with selection bias.

```
nps_model = BayesianNetwork([
   ("RND", "New Feature"),
#     ("New Feature", "Customer Satisfaction"),
    ("Customer Satisfaction", "NPS"),
    ("Customer Satisfaction", "Response"),
    ("New Feature", "Response"),
])

nps_model.is_dconnected("NPS", "New Feature", observed="Response")

True
```

To gain further intuition into this bias let's get your godlike powers back and pretend you can see into the world of counterfactual outcomes. That is, you can see both the NPS customers would have under the control, NPS_0, and under the treatment, NPS_1, for all customers, even those that didn't answer the survey. Lets also simulate data in such a way that we know the true effect. Here, I'm saying that the new feature increases NPS by 0.4 (which is an absurdly high number for any business standards, but bear with me for the sake of the example). Let's also say that both the new feature and customer satisfaction increases the chance of responding to the NPS survey, just like we've shown in the graph above. With the power to measure counterfactuals, this is what you would see if you aggregated the data by the treated and control groups:

	responded	nps_0	nps_1	nps
new_feature				
0	0.183715	-0.005047	0.395015	-0.005047
1	0.639342	-0.005239	0.401082	0.401082

First, notice that 63% of those with the new feature responded to the NPS survey, while only 18% of those in the control responded to it. Next, if you look at both treated and control rows, you'll see an increase of 0.4 by going from NPS_0 to NPS_1. This simply means that the effect of the new feature is 0.4 for both groups. Finally, notice that the difference in NPS between treated and control is about 0.4. Again, this means that, if you could see the NPS of those that did not respond to the NPS survey, you could just compare treated and control groups to get the true ATE.

Of course, in reality, you can't see the columns NPS_0 and NPS_1. You also also can't see the NPS column like this, because you only have NPS for those that responded to the survey (18% of the control rows and 63% of the treated rows).

	responded	nps_0	nps_1	nps
new_feature				
0	0.183715	NaN	NaN	NaN
1	0.639342	NaN	NaN	NaN

If you further break down the analysis by respondents, you get to see the NPS of those where $Response = 1$. But notice how the difference between treated and control in that group is no longer 0.4, but only about half of that (0.22). How can that be? This is all due to selection bias.

		nps_0	nps_1	nps
responded	**new_feature**			
0	**0**	NaN	NaN	NaN
	1	NaN	NaN	NaN
1	**0**	NaN	NaN	0.314073
	1	NaN	NaN	0.536106

Adding back the unobservable quantities, you can see what is going on (focus on the respondents group here).

		nps_0	nps_1	nps
responded	**new_feature**			
0	**0**	-0.076869	0.320616	-0.076869
	1	-0.234852	0.161725	0.161725
1	**0**	0.314073	0.725585	0.314073
	1	0.124287	0.536106	0.536106

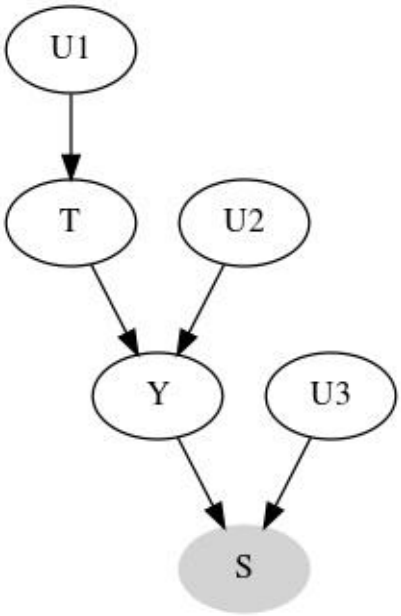
Initially, treated and control groups were comparable, in terms of their baseline satisfaction Y_0. But once you condition on those that responded the survey, those in the treatment group have lower baseline satisfaction $E[Y_0|T=0,R=1] > E[Y_0|T=1,R=1]$. This means that a simple difference in averages between treated and control does not identify the ATE, once you condition on those that

responded.

$$E[Y|T=1,R=1] - E[Y|T=1,R=1] = \underbrace{E[Y_1 - Y_0|R=1]}_{ATE} + \underbrace{E[Y_0|T=0,R=1] - E[Y_0|T=1,R=1}_{SelectionBias}$$

That bias term there won't be zero if the outcome, customer satisfaction, affects the response rate. Since satisfied customers are more likely to answer the NPS survey, identification is impossible in this situation. If the treatment increases satisfaction, then, the control group will contained more customers whose baseline satisfaction ($Y_0|R=1$) is higher than the control group. That's because the treated group will have those who were satisfied (high baseline satisfaction) plus those who had low baseline satisfaction, but due to the treatment, became more satisfied and answered the survey.

This means you don't even need the arrow from the treatment to the outcome to have this sort of selection bias. To see this, you can remove the arrow T -> S and expand the graph to contain the sources of variation in each variable: U_1 to U_3. If the treatment has non-zero effect, then the selection node is a descendant of a collider - the outcome itself.



I'll admit that the graph argument might not be that easy to follow here. This is a weird case that it is referred to as a virtual collider. But despite being weird, it is an incredibly common source of selection bias. Since it might be a bit counterintuitive, it's nice to hit it from different perspectives. So, let's take that graphical argument from above and transform it into something more visual. Ant to simplify things, let's pretend we can measure customer satisfaction directly. We'll still only be able to measure it for those that answer our survey, but now we don't need a proxy, like NPS.

Again, we'll rely on simulated data for which we know the true causal effect and the potential outcomes. In fact, simulating data is an incredible skill you should master in order to better understand causal inference. Whenever you fill like you don't understand what is going on, try to simulate data following the process you want to understand.

Here, I'm simulating data following the graph we've just discussed. I'm assuming the treatment (the

new app feature) is randomized, with 50% of the customers getting the treatment and 50%, the control. I'm also simulating a treatment effect of 0.4. I'm doing this by first simulating the Y_0 potential outcome and making it so that $Y_1 = Y_0 + 0.4$. Next, to make the observed outcome, I'll use a switch which will be Y_1 if the customer gets the treatment and Y_0, if the customer gets the control:

$$Y_i = T_i Y_{1i} + (1 - T_i) Y_{0i}$$

```
np.random.seed(2)
n = 100000

# 50% treated, 50% control
new_feature = np.random.binomial(1, 0.5, n)

# the new_feature (treatment) increases satisfaction by 0.4
satisfaction_0 = np.random.normal(0, 0.5, n)
satisfaction_1 = satisfaction_0 + 0.4
satisfaction = new_feature*satisfaction_1 + (1-new_feature)*satisfaction_0

# following the graph above, response is cause by satisfaction,
# but not by the new feature.
responded  = (np.random.normal(satisfaction, 1) > 1).astype(int)

tr_df = pd.DataFrame(dict(new_feature=new_feature,
                          responded=responded,
                          satisfaction_0=satisfaction_0,
                          satisfaction_1=satisfaction_1,
                          satisfaction=satisfaction))
```

Finally, I'll simulate the response. Here, just like in the graph you just saw, I'm making response dependent only of customer satisfaction. The treatment does not impact response directly, only through the outcome.

With all of that simulated, I'll store everything into a dataframe. It's important to remember that, in reality, we'll only be able to measure the satisfaction from the customers that responded. We are only simulating data as though everything was observable to better understand what is going on with selection bias. For instance, if we compare satisfaction between treated and control group in this data, we will easily find the correct effect of 0.4.

```
tr_df.groupby("new_feature").mean()
```

	responded	satisfaction_0	satisfaction_1	satisfaction
new_feature				
0	0.182337	-0.001696	0.398304	-0.001696
1	0.294694	0.001166	0.401166	0.401166

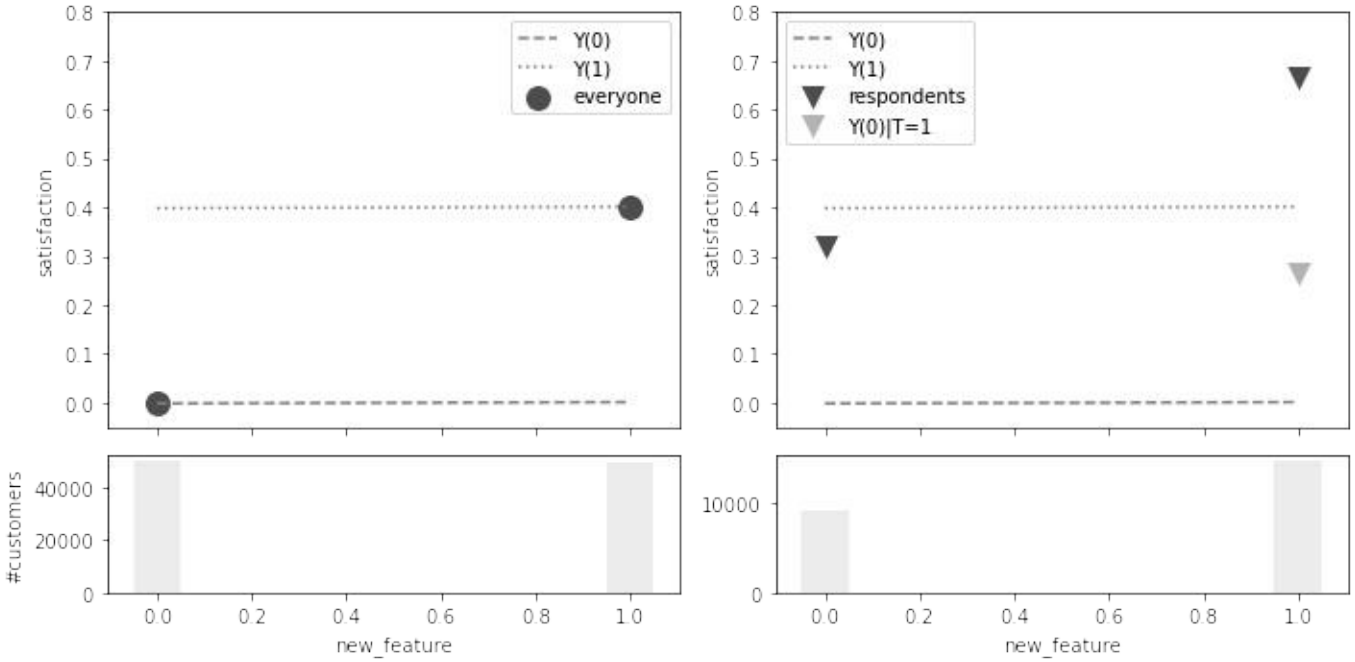
But if you only look at the customers that responded the survey, you can see that the difference in

satisfaction between treated and control is smaller than 0.4. This is the difference you would see in reality, since you would only be able to measure satisfaction for those that answered the survey.

```
tr_df.query("responded==1").groupby("new_feature").mean()
```

	responded	satisfaction_0	satisfaction_1	satisfaction
new_feature				
0	1.0	0.31957	0.71957	0.31957
1	1.0	0.26415	0.66415	0.66415

To see what is going on, you can plot the satisfaction of both treated and control considering you could measure it for everyone. This is what is shown in the first column of plots here. Notice that the observed outcome matches exactly the potential outcomes. In other words $E[Y|T=1] = Y_1$ and $E[Y|T=0] = Y_0$. Also, notice the number of customers. Due to randomization, we have 50% of them in the treated group and 50% of them in the control group.



However, once you condition on those that responded to the survey, the treatment is no longer random. Notice how there are fewer customers in the control group than in the treated group. Even though I've generated data so that the treatment didn't cause the response, the treatment does increase satisfaction, and only those with higher satisfaction tend to answer the survey. You can see this by the average satisfaction of the respondents in the control group. It went from 0 (everyone) to about 0.35 (respondents). More satisfaction means more response, so those that responded were more satisfied with the product. That is also true for the treated group. In it, the satisfaction went from 0.4 to about 0.65. So, if satisfaction in both groups went up once you condition on respondents, where is the bias coming from?

For us to have bias that reduces the measured effect, it must be that conditioning on response increased

the average satisfaction in the control group more than it did in the treated group. Or, similarly, that inherently satisfied customers (those that were satisfied regardless of the treatment) became overrepresented in the control group. You can see that this is the case by plotting the (unobserved) potential outcome $Y_0|T=1$. Notice that it is lower than that of the control group. This difference is what is causing the bias.

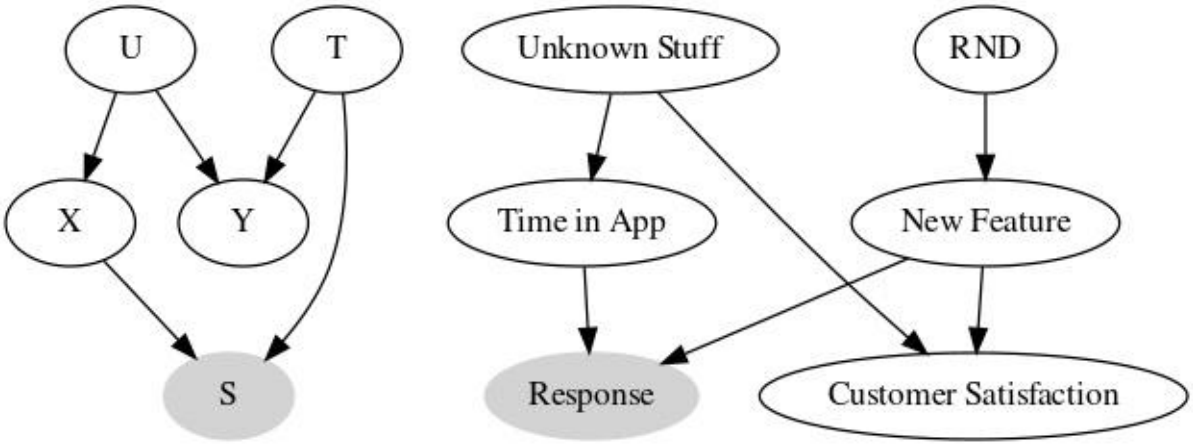
$$E\left[Y_0|T=0,R=1\right] > E\left[Y_0|T=1,R=1\right]$$

In other words, since satisfaction increases response rate, once you condition on respondents, treated and untreated are no longer comparable. The control group will have only those customers that were more satisfied with your product, regardless of the treatment. The treatment group will have those customers too, but also the ones that were dissatisfied at first, but became satisfied once they got the treatment. These initially dissatisfied customers will pull the average Y_0 down in the treatment group, which causes bias to arise.

Adjusting for Selection Bias

Unfortunately, correcting selection bias is not at all trivial. In the example we've been discussing, even with a randomized control trial, the ATE is not identifiable, simply because you can't close the non-causal flow of association between the new feature and satisfaction, once you condition on those that responded to the survey. In order to make some progress, you need to make further assumptions, and here is where the graphical model starts to shine. It allows you to be very explicit and transparent about those assumptions.

For instance, let's assume that the outcome doesn't cause selection. In our example, this would mean that customer satisfaction doesn't cause customers to be more or less likely to answer the survey. Instead, you would have some other variable (or variables) which cause both selection and the outcome. For example, it could be that the only thing which causes customers to respond to the survey is the time they spend in the App and the new feature. In this case, the non-causal association between treatment and control flows through Time in the App.



How strong of an assumption that is, it's something that only expert knowledge will be able to tell. But if it is correct, the effect of the new feature on satisfaction becomes identifiable once you control for time in the App. You can verify this by checking if the treatment and outcome are still connected in the graph with the causal path removed. As you've seen before, once you condition on response, you'll open a non-causal association path. However, conditioning on time in the App closes that path again.

```
nps_model = BayesianNetwork([
    ("RND", "New Feature"),
#     ("New Feature", "Customer Satisfaction"),
    ("Unknown Stuff", "Customer Satisfaction"),
    ("Unknown Stuff", "Time in App"),
    ("Time in App", "Response"),
    ("New Feature", "Response"),
])

print(nps_model.is_dconnected("Customer Satisfaction", "New Feature", observed=["Response"]))
print(nps_model.is_dconnected("Customer Satisfaction", "New Feature", observed=["Response", "Time in App"]))
```

```
True
False
```

If the data is generated according to the above graph, you still have selection bias, as you can see from the simple comparison in means that follows. The recovered effect is lower (about .25) than the true 0.4 effect.

```
tr_df_measurable.groupby(["new_feature"]).mean()
```

	responded	time_in_app	satisfaction
new_feature			
0	0.325975	0.498164	0.299958
1	0.669546	0.495711	0.541878

However, once you control for time in the app (and for the sake of simplicity, let's say you can group customer into just two groups: a lot of time spent in the app, $X = 1$ and a short amount of time spent in the app $X = 0$), you once again manage to recover the true ATE of 0.4.

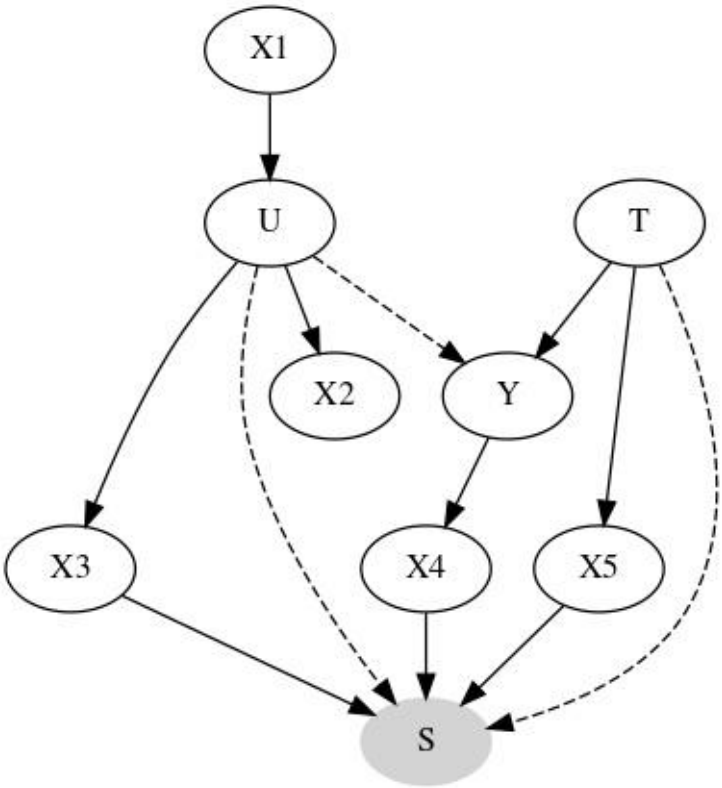
```
(tr_df_measurable
 .groupby(["time_in_app", "new_feature"])
 .mean())
```

		responded	satisfaction
time_in_app	**new_feature**		
0	**0**	0.156664	-0.564384
	1	0.499980	-0.151300
1	**0**	0.496534	0.574681
	1	0.842046	0.960587

Once again, you are applying the adjustment formula here. You are simply segmenting the data into groups defined by X (Time in App) so that treated and control groups become comparable within those segments. Then, you can just compute the weighted average of the in-group comparison between treated and control, using the size of each group as the weights. Only now, you are doing all of this while also conditioning on the selection variable (customers that answered the survey).

$$ATE = \sum_x \Big\{ (E[Y|T=1, R=1, X] - E[Y|T=0, R=1, X])P(X|R=1) \Big\}$$

Generally speaking, to adjust for selection bias, you have to adjust for whatever causes selection and you also have to assume that the neither the outcome not the treatment causes selection directly or shares a hidden common cause with selection. For instance, in the following graph, you have selection bias since conditioning on S opens a non-causal association path between T and Y.

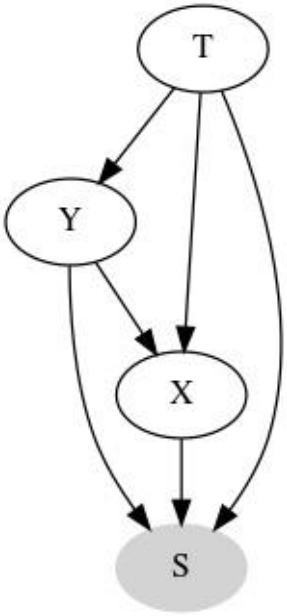


You can close two of these paths by adjusting for measurable variables that explain selection, $X3$, $X4$ and $X5$. However, there are two paths you cannot close (shown in dashed lines): Y -> S <- T and T -> S <- U -> Y. That's because the treatment causes selection directly and the outcome shares a hidden common cause with selection. You can mitigate the bias from this last path by further conditioning on $X2$ and $X1$, as they account for some variation in U, but that will not eliminate the bias completely.

This graph reflects a more plausible situation you will encounter when it comes to selection bias, like the response bias we've just used as an example. In these situations, the best you can do is to condition on variables that explain the selection. This will reduce the bias, but it won't eliminate it because, as you saw, 1) there are things which cause selection that you don't know or can't measure, 2) the outcome or the treatment might cause selection directly.

I also don't want to give you the false idea that just controlling for everything which causes selection is

a good idea. If you have a variable X, which does cause selection, but which is also a common effect of the treatment and the outcome, conditioning on it will close a path but open another.

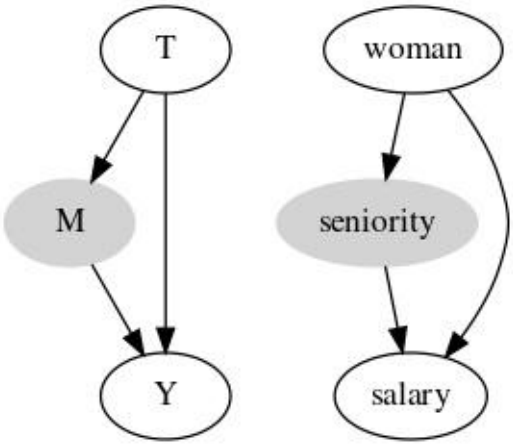


Here, conditioning on X closes the non causal path Y->S<-X<-T, but it opens another non causal path: Y->X<-T.

Conditioning on a Mediator

While the selection bias discussed so far is caused by unavoidable selection into a population (you were forced to condition on the respondent population), you can also cause selection bias inadvertently. For instance, let's suppose you are working in HR and you want to find out if there is a gender pay gap in your company. In other words, you wish to know if men and women are compensated equally well. But to do that analysis, you want to control for seniority level. In other words, you think that if men and women in the same position have different salaries, you will have evidence of a gender pay gap in your company.

The issue with this analysis is that the causal diagram probably looks something like this:

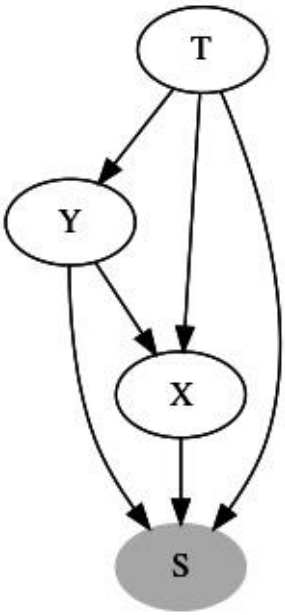


This means that the seniority level is a **mediator** in the path between the treatment (woman) and salary. Intuitively, the difference in salary between woman and man has a direct cause (the direct path, woman -> salary) and an indirect cause, which flows through the seniority (the indirect path woman -> seniority -> salary). What this graph tells you is that one way woman can suffer from discrimination is by being less likely to be promoted to higher seniorities. The difference in salary between men and women is

partly the difference in salary at the same seniority level, but also the difference in seniority. Simply put, the path woman -> seniority -> salary is also a causal bath between the treatment and the outcome, and you shouldn't close it in your analysis. But you will, if you try to measure the difference in salaries between men and women, while controlling for seniority.

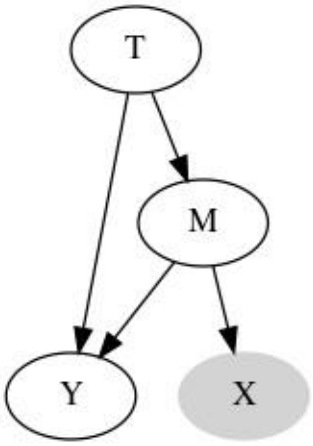
I can also give you a more visual argument to why conditioning on the mediator is problematic. The following plot has women as squares and men as circles. The size of each dot is proportional to the number of employees in that group. You can see that there are more men than women at higher seniority levels: the squares are bigger than the circles at lower seniority levels and the circles are bigger at higher seniority levels. The dashed line represents the average salary for men, which is about 30000 higher than the solid line, which represents the average salary for women.

	salary	diff
woman		
0	127.965634	NaN
1	95.344533	-32.621101



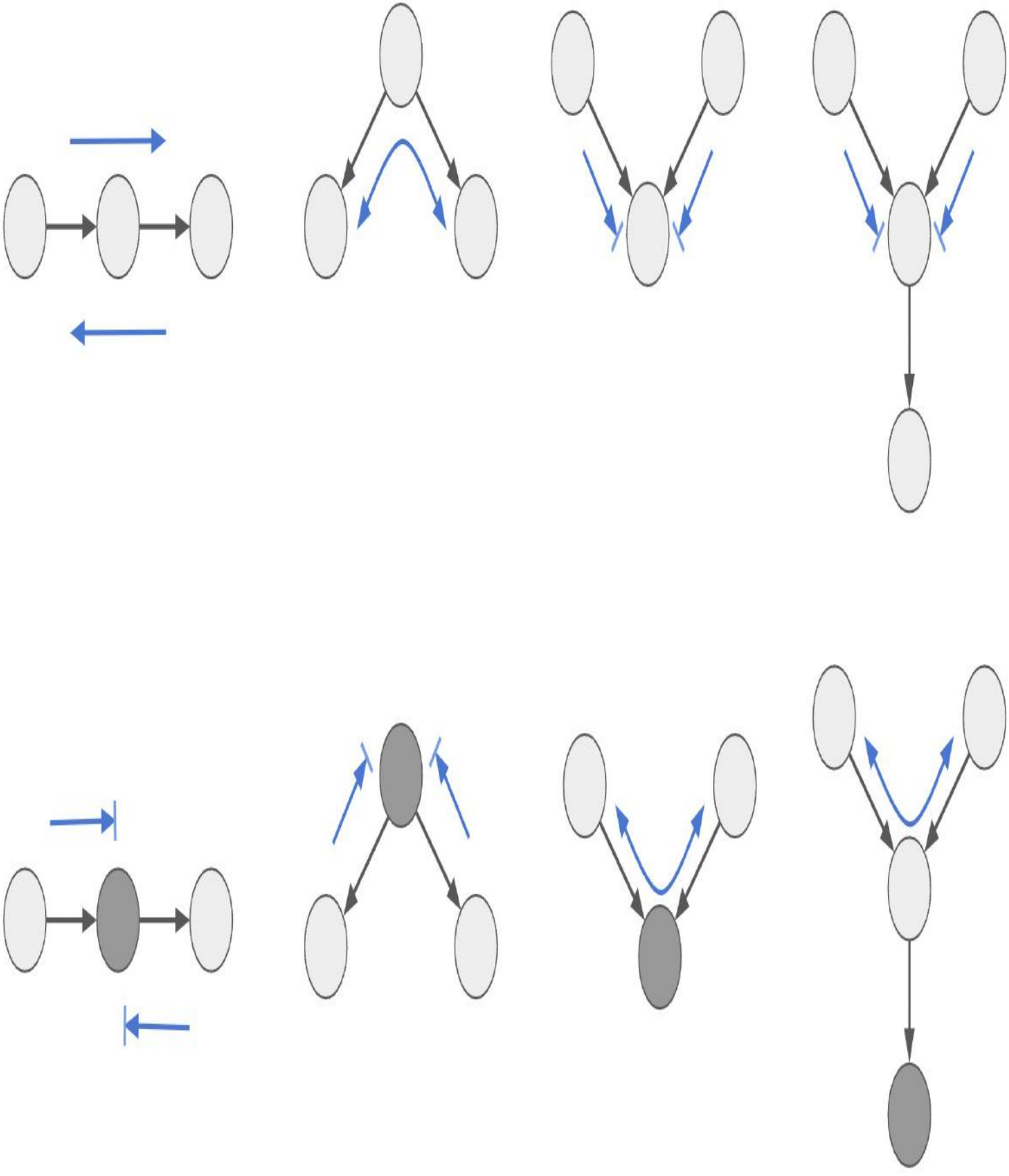
Even though the marginal difference in salary between genders is huge, if you condition on seniority, you would not see that. If you look inside each seniority level, part of the gender pay gap goes away.

It is also worth mentioning that conditioning on descendants of the mediator node also induces bias towards zero. This sort of selection doesn't completely shut the causal path, but it partially blocks it.



Key Ideas

In this chapter, you focused mostly on the identification part of causal inference. The goal was to learn how to use graphical models to be transparent about the assumptions you are making and to see what kind of association - causal or not - those assumptions entail. In order to do that, you had to learn how association flows in a graph. This cheat sheet is a good summary of those structures, so I recommend you keep it close by.

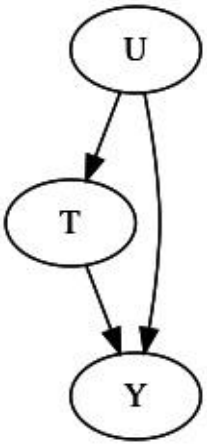


Then, you saw that identification amounts to isolating the causal flow of association from the non-causal ones in a graph. You could close non-causal paths of association by adjusting (conditioning) on

some variables or even intervening on a graph, like in the case where you do a randomized experiment. Bayesian Network software, like `pgmpy` is particularly useful here, as it aids you when checking if two nodes are connected in a graph. For instance, to check for confounder bias, you can simply remove the causal path in a graph and check if the treatment and outcome nodes are still connected, even with that path removed. If they are, you have a backdoor path that needs to be closed.

Finally, you went through two very common structures of bias in causal problems. Confounding bias happens when the treatment and the outcome share a common cause. This common cause forms a fork structure, which creates a non-association flow between the treatment and the outcome.

```
g = gr.Digraph()
g.edge("U", "T")
g.edge("U", "Y")
g.edge("T", "Y")

g
```



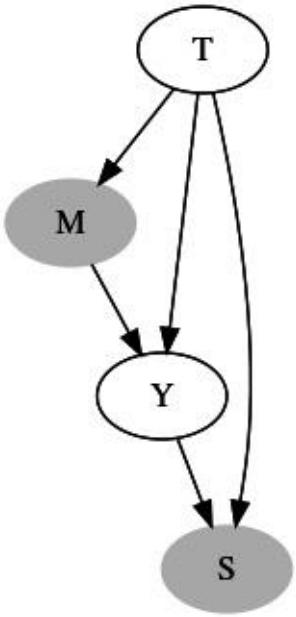
To fix confounding bias, you need to try to adjust for the common causes, directly or by the means of proxy variables. This motivated the idea of the adjustment formula,

$$ATE = \sum_x \{(E[Y|T=1, X=x] - E[Y|T=0, X=x])P(X=x)\},$$

and the Conditional Independence Assumption, which states that, if treatment is as good as randomly assigned within groups of variables X, then you can identify causal quantities by conditioning on X.

Alternatively, if you can intervene on the treatment node, confounding becomes a lot easier to deal with. For instance, if you design a random experiment, you'll create a new graph where the arrows pointing to the treatment are all deleted, which effectively annihilates confounding bias.

You also saw selection bias, which appears when you condition on a common effect (or descendant of a common effect) between the treatment and the outcome or when you condition on a mediator node (or a descendent of a mediator node). Selection bias is incredibly dangerous because it does not go away with experimentation. To make it even worse, it can be quite counterintuitive and hard to spot.



Again, it is worth mentioning that causal graph is like a new language. You'll learn most of it by seeing it again and again and by trying to use it.

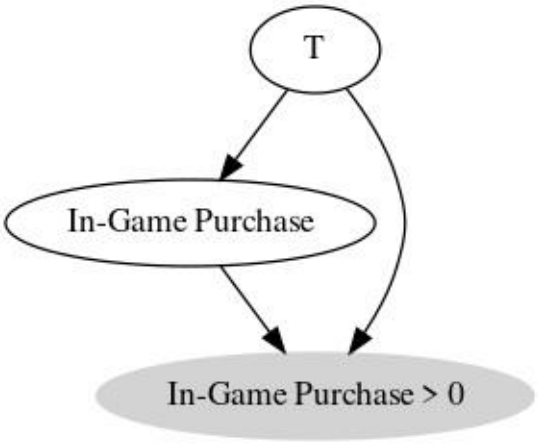
> **SEE ALSO**
>
> If you want very neat article to review this language, I strongly recommend you check out the paper **A Crash Course in Good and Bad Controls**, by Carlos Cinelli, Andrew Forney and Judea Pearl. It goes through everything covered in this chapter and more. The article is also written in clear language, making it easy to read.

Other Examples

These examples are used to solidify what you learned in this chapter. They use concepts and techniques you just learned and are presented in a fast paced manner. As you read through them, try to link what they tell you with what you learned in this chapter. Also, use them to try to generalize the ideas outlined here to applications in your own work.

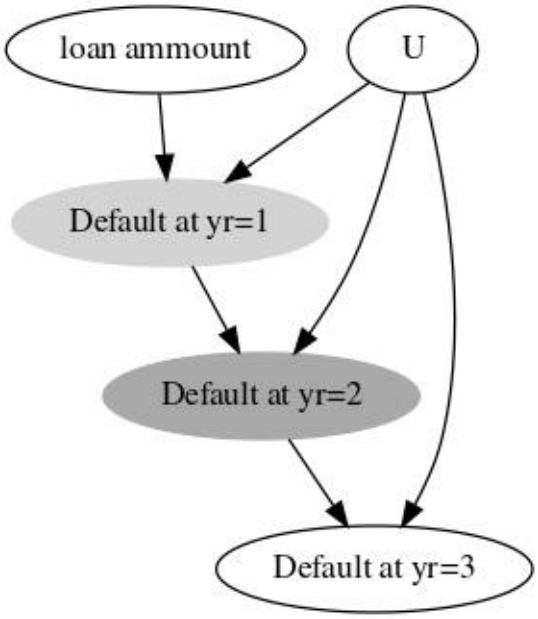
Conditioning on the Positives

Many tech companies are structured around a free product, which attracts the majority of the users but generates no revenue, and a payed option, which only a minority of the users select. For instance, a mobile game might be free to play, but users can choose to buy better items which enhances their playing experience. Since the vast majority of the users don't pay, it's common to have in-game purchases as a zero inflated distribution. This causes the - somewhat misguided - belief that the effect of any new feature introduced (treatment) will be diluted away. In an attempt to fix that, after randomizing the treatment, analysts condition the analysis of the effect on those customers whose in-game purchase was above zero. This of course induces selection bias.



The Hidden Bias in Survival Analysis

Survival analysis appears in many business applications that are not necessarily pharmaceutical. For instance, a bank is very interested in understanding how the size of a loan (loan amount) increases the chance of a customer defaulting on that loan. Consider a 3 year loan. The customer can default in the first, second or third year, or not default at all. The goal of the bank is to know how loan amount impacts $P(Default|yr = 1)$, $P(Default|yr = 2)$ and $P(Default|yr = 3)$. Here, for simplicity's sake, consider that the bank has randomized the loan amount. Noice how only customers that survived (did not default) in year 1 are observed in year 2 and only customers that survived years 1 and 2 are observed in year 3. This selection makes it so that only the effect of loan size in the first year is identifiable.



Intuitively, even if the loan amount was randomized, it only stays that way in the first year. After that, if loan amount increases the chance of default, customers with lower risk of defaulting will be overrepresented in the region with high loan amounts. Their risk will have to be low enough to offset the increase caused by a bigger loan amount, otherwise, they would have defaulted at year 1. If the bias is too extreme, it can even look like bigger loans cause risk to decrease in a year after year 1, which doesn't make any sense.

A simple solution for this selection bias problem is to focus on cumulative outcome (survival), $Y|time > t$, rather than yearly outcomes (hazard rates), $Y|time = t$. For example, even though you can't identify the effect of loan amount on default at year 2, $P(Default|yr = 2)$, you can easily

identify the effect on default up to year 2, $P(Default|yr \leq 2)$.

Chapter 4. The Unreasonable Effectiveness of Linear Regression

A NOTE FOR EARLY RELEASE READERS

With Early Release ebooks, you get books in their earliest form—the author's raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the 4th chapter of the final book. Please note that the GitHub repo will be made active later on.

If you have comments about how we might improve the content and/or examples in this book, or if you notice missing material within this chapter, please reach out to the editor at *vwilson@oreilly*.

In this chapter you'll add the first major debiasing technique in your causal inference arsenal: linear regression or Ordinary Least Squares (OLS) and orthogonalization. You'll see how linear regression can adjust for confounders when estimating the relationship between a treatment and an outcome. But, more than that, I hope to equip you with the powerful concept of treatment orthogonalization. This idea, born in linear regression, will come in handy later on when you start to use machine learning models for causal inference.

All You Need is Linear Regression

Before you skip to the next chapter because, "oh, regression is so easy! It's the first model we learn as data scientists" and yada yada, let me assure you that no, you actually don't know linear regression. In fact, regression is one of the most fascinating, powerful and dangerous models in causal inference. Sure, it's more than 100 years old. But, to this day, it frequently catches even the best causal inference researchers off guard.

OLS RESEARCH

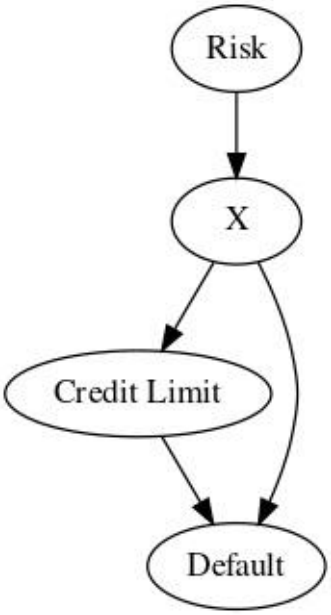
Don't believe me? Just take a look at the recent literature on the Diff-in-Diff method that you will see. A good place to start is the article *Difference-in-Differences with Variation in Treatment Timing*, by Andrew Goodman-Bacon, or the paper *Interpreting OLS Estimands When Treatment Effects Are Heterogeneous* by Tymon Słoczyński, or even the the paper *Contamination Bias in Linear Regressions* by Goldsmith-Pinkham, Hull & Kolesár

I assure you: not only is regression the most used model in causal inference, but it will be the one you'll use the most. Regression is also a major building block for more advanced techniques, like most of the panel data methods (like Difference-in-Differences and Two-Way Fixed Effects), machine learning methods (like Double-Debiased Machine Learning) and advanced identification techniques (like

instrumental variables).

Why We Need Models

Now that I hopefully convinced you to say, we can get down to business. To motivate the use of regression, let's consider a pretty challenging problem in banking and the lending industry in general: understanding the impact of loan amount or credit card limits on default rate. Naturally, increasing someone's credit card limit will increase (or at least not decrease) the odds of he or she defaulting on the credit card bill. But, if you look at any banking data, you will see a negative correlation between credit lines and default rate. Obviously, this doesn't mean that higher lines cause customers to default less on their credit card. Rather, it simply reflects the treatment assignment mechanism: banks and lending companies offer more credit to customers that have a lower chance of defaulting, as perceived by their underwriting models. The negative correlation you see is the effect of confounding bias.



Of course the bank doesn't know the inherent risk of default, but it can use proxy variables X - like income or credit scores - to estimate it. In the previous chapters, you saw how you could adjust for variables so that you could make the treatment look as good as randomly assigned. Specifically, you saw how the adjustment formula,

$$ATE = E_x\left\{E[Y|T=1,X=x] - E[Y|T=0,X=x]\right\},$$

which, together with the conditional independence assumption

$$(Y_0, Y_1) \perp T|X$$

Allows you to identify the causal effect even in the presence of bias. Or rather, adjust for it.

However, if you were to literally apply the adjustment formula, you'll see how things can get out of hand pretty quickly. First, you would need to partition your data into segments defined by the features X. This would be fine if you had very few discrete features. But what if there are many of them, with some being continuous? For example, let's say you know a bank used 10 variables, each with 3 groups, to underwrite customers and assign credit lines. That doesn't seem a lot right? Well, it will already

amount to 59049 cells, or 3^10. Estimating the ATE in each of those cells and average the result is only possible if you have massive amounts of data. This is the **curse of dimensionality**, a problem very familiar to most data scientists. In the context of causal inference, one implication of this curse is that a naive application of the adjustment formula will suffer if you have lots and lots of covariates.

CAUSAL INFERENCE VS MACHINE LEARNING LINGO

The literature on Machine Learning, which is what most data scientists are familiar with, uses different terms from the literature on causal inference, which usually comes from econometrics or epidemiology. So, in the event that you need to translate from one to the other, here are some of the main equivalences you will encounter:

- Feature: covariates or independent variables;
- Weights: parameters or coefficients;
- Target: outcome or dependent variable.

One way out of this dimensionality problem is to use an **outcome model** like linear regression, which can interpolate and extrapolate the many individual X defined cells. You can think about linear regression in this context as a sort of dimensionality reduction algorithm. It projects all the X variables into the outcome dimension and makes the comparison between treatment and control on that projection. It's quite elegant. But I'm getting ahead of myself. To really (and I mean truly, with every fiber of your heart) understand regression, you have to start small.

Regression in A/B Tests

Pretend you work for an online streaming company, perfecting their recommender system. Your team just finished a new version of this system, with cutting edge technology and the latest ideas from the machine learning community. While that's all very impressive, what your manager really cares about is if this new system will increase the watch time of the streaming service. In order to test that, you decide to do an A/B test. First, you sample a representative but small fraction of your customer base. Then, you deploy the new recommender to a random 1/3 of the customers in that sample, while the rest continue to have the old version of the recommender. After a month, you collect the results in terms of average watch time per day.

```
data.head()
```

	recommender	age	tenure	watch_time
0	challenger	15	1	2.39
1	challenger	27	1	2.32
2	benchmark	17	0	2.74
3	benchmark	34	1	1.92
4	benchmark	14	1	2.47

Since the version of recommender was randomized, a simple comparison of average watch time between versions would already give you the ATE of the new recommender. But then you had to go through all the hassle of computing standard errors to get confidence intervals in order to check for statistical significance. So, what if I told you that you can interpret the results of an A/B test with regression, which will give us, for free, all the inference statistics you need? The idea behind regression is that you'll estimate the following equation or model:

$$WatchTime_i = \beta_0 + \beta_1 challenger_i + e_i$$

Where $challenger$ is 1 for customers in the group that got the new version of the recommender and zero otherwise. If you estimate this model, the impact of the challenger version will be captured by the estimate of β_1, $\widehat{\beta_1}$.

To run that regression model in Python, you can use `statsmodels`' formula API. It allows you to express linear models succinctly, using R-style formulas. For example, you can represent the above model with the formula `'watch_time ~ C(recommender)'`. To estimate the model, just call the method `.fit()` and to read the results, call `.summary()` on a previously fitted model.

```
import statsmodels.formula.api as smf

result = smf.ols('watch_time ~ C(recommender)', data=data).fit()

result.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025
Intercept	2.0491	0.058	35.367	0.000	1.935
C(recommender) [T.challenger]	0.1427	0.095	1.501	0.134	-0.044

Notice that the outcome variable comes first, followed by a ~. Then, you add the explanatory variables. In this case, you'll just use the recommender variable, which is categorical with 2 categories (one for

the challenger and one for the old version). You can wrap that variable in `C(...)` to explicitly state that the column is categorical.

> **SEE ALSO**
>
> The formula synthetic sugar is an incredibly convenient way to do feature engineering. You can learn more about it in the patsy library.

Next, look at the results. First, you have the intercept. This is the estimate for the β_0 parameter in our model. It tells us the expected value of the outcome when the other variables in the model are zero. Since the only other variable here is the challenger, you can interpret the intercept as the expected watch time for those that received the old version of the recommender system. Here, it means that customers spend, on average 2.04 hours per day watching your streaming content, when with the old version of the recommender system. Finally, looking at the parameter estimate associated with the challenger recommender, $\widehat{\beta_1}$, you can see the increase in watch time due to this new version. If $\widehat{\beta_0}$ is the estimate for the watch time under the old recommender, $\widehat{\beta_0} + \widehat{\beta_1}$ tells us the expected watch time for those who got the challenger version. In other words, $\widehat{\beta_1}$ is an estimate for the ATE. Due to randomization, you can assign causal meaning to that estimate: you can say that the new recommender system increased watch time by 0.14 hours per day, on average. However, that result is not statistically significant.

> **HATS**
>
> I'll use hats to denote the estimate of parameters and predictions.

Forget the non significant result for a moment, because what you just did was quite amazing. Not only did you estimate the ATE, but also got, for free, confidence intervals and p-values out of it! More than that, you can see for yourself that regression is doing exactly what it supposed to do: estimating $E[Y|T]$ for each treatment.

```
(data
 .groupby("recommender")
 ["watch_time"]
 .mean())
```

```
recommender
benchmark     2.049064
challenger    2.191750
Name: watch_time, dtype: float64
```

Just like I've said, the intercept is mathematically equivalent to the average watch time for those in the control - the old version of the recommender.

These numbers are identical because, in this case, regression is mathematically equivalent to simply doing a comparison between two averages, while the $\widehat{\beta_1}$ is the average difference between the two

groups: $2.191 - 2.049 = 0.1427$. Ok, so you managed to, quite literally, reproduce group averages with regressions. But so what? It's not like you couldn't do this earlier, so what is the real gain here?

Adjusting with Regression

To appreciate the power of regression, let me take you back to our initial example: estimating the effect of credit lines on default. Bank data usually looks something like this, with a bunch of columns of customer features that might indicate credit worthiness, like monthly wage, lots of credit scores provided by credit bureaus, tenure at current company and so on. Then, there is the credit line given to that customer (the treatment in this case) and the column which tells us if a customer defaulted or not - the outcome variable.

> **NOTE**
>
> Once again, I'm building from real world data and changing it to fit the needs of this chapter. This time, I'm using the `wage1` data, curated by professor Jeffrey M. Wooldridge and available in the 'wooldridge' R package.

```
risk_data.head()
```

	wage	educ	exper	married	credit_score1
0	950.0	11	16	1	500.0
1	780.0	11	7	1	414.0
2	1230.0	14	9	1	586.0
3	1040.0	15	8	1	379.0
4	1000.0	16	1	1	379.0

First, notice that the treatment, credit limit, has way too many ordered categories. In this situation, it is better to treat it as a continuous variable, rather than a discrete one. Here, instead of representing the ATE as the difference between multiple levels of the treatment, you can represent it as the derivative of the expected outcome with respect to the treatment:

$$ATE = \frac{\partial}{\partial t} E\left[y|t\right]$$

Don't worry if this sounds fancy. It simply means the amount you expect the outcome to change given a unit increase in the treatment. In our example, it represents how much you expect the default rate to change given a 1 USD increase in credit lines.

One way to estimate such a quantity is to run a regression. Specifically, you can estimate the model

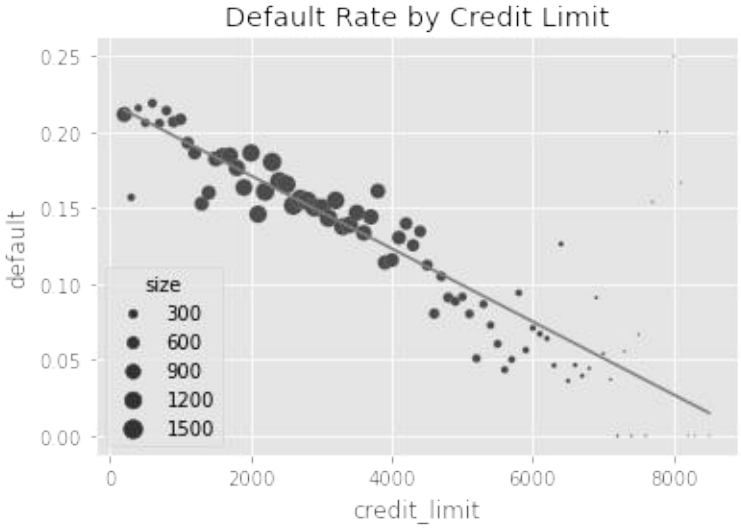
$$Default_i = \beta_0 + \beta_1 limit_i + e_i,$$

and the estimate $\widehat{\beta}_1$ can be interpreted as the amount you expect risk to change given a 1 USD increase in limit. This parameter has a causal interpretation if the limit was randomized. But as you know very well that is not the case, as banks tend to give higher lines to customers that are less risky. In fact, if you run the model above, you'll get a negative estimate for β_1.

```
model = smf.ols('default ~ credit_limit', data=risk_data).fit()
model.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025
Intercept	0.2192	0.004	59.715	0.000	0.212
credit_limit	-2.402e-05	1.16e-06	-20.689	0.000	-2.63e-05

That is not at all surprising, given the fact that the relationship between risk and credit limit is negative, due to confounding. If we plot the fitted regression line alongside the average default by credit limit, you can clearly see the negative trend.



To adjust for this bias, you could, in theory, segment your data by all the confounders, run a regression of default on credit lines inside each segment, extract the slope parameter and average the results. However, due to the curse of dimensionality, even if you try to do that for a moderate number of confounders - both credit scores - you will see that there are cells with only one sample, making it impossible for you to run your regression. Not to mention the many cells that are simply empty.

```
risk_data.groupby(["credit_score1", "credit_score2"]).size().head()
```

```
credit_score1  credit_score2
34.0           339.0           1
               500.0           1
52.0           518.0           1
69.0           214.0           1
               357.0           1
dtype: int64
```

Thankfully, once more, regression comes to your aid here. Instead of manually adjusting for the confounders, you can simply add them to the model you'll estimate with OLS.

$$Default_i = \beta_0 + \beta_1 limit_i + \boldsymbol{\theta} \boldsymbol{X}_i + e_i,$$

Here, $\boldsymbol{X}$ is a vector of confounder variables and $\boldsymbol{\theta}$ is the vector of parameters associated with those confounders. There is nothing special about $\boldsymbol{\theta}$ parameters. They behave exactly like β_1. I'm just representing them differently because they are just there to help us get an unbiased estimate for β_1. That is, you don't really care about their causal interpretation (they are technically called **nuisance parameters**).

In the credit example, you could add the credit scores and wage confounders to the model. It would look like this:

$$Default_i = \beta_0 + \beta_1 limit_i + \theta_1 wage_i + \theta_2 creditScore1_i + \theta_3 creditScore2_i + e_i,$$

I'll get into more details about how including variables in the model will adjust for confounders, but there is a very easy way to see it right now. The model above is a model for $E[y|t, X]$. Recall that you want $\frac{\partial}{\partial t} E\left[y|t, X\right]$. So what happens if you differentiate the model above with respect to the treatment - credit limit? Well, you simply get β_1! This means that β_1 can be seen as the partial derivative of the expected value of default with respect to credit limit. Or, more intuitively, it can be viewed as how much you should expect default to change, given a small increase in credit limit, **while holding fixed all other variables in the model**. This interpretation already tells you a bit of how regression adjusts for confounders: it holds them fixed while estimating the relationship between the treatment and the outcome.

To see this in action, you can estimate the model above. Just add some confounders and, like some kind of magic, the relationship between credit lines and default becomes positive!

```
model = smf.ols('default ~ credit_limit + wage + credit_score1  + credit_score2', data=risk_data).fit()
model.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025
Intercept	0.4037	0.009	46.939	0.000	0.387
credit_limit	3.063e-06	1.54e-06	1.987	0.047	4.16e-08
wage	-8.822e-05	6.07e-06	-14.541	0.000	-0.000
credit_score1	-4.175e-05	1.83e-05	-2.278	0.023	-7.77e-05
credit_score2	-0.0003	1.52e-05	-20.055	0.000	-0.000

Don't let the small estimate of β_1 fool you. Recall that limit is in the scales of 1000s while default is

either 0 or 1. So it is no surprise that increasing lines by 1 USD will increase expected default by a very small number. Still, that number is statistically significant and tells us that risk increases as you increase credit limit, which is much more in line with our intuition on how the world works.

Hold that though because you are about to explore it more formally. It's finally time to learn one of the greatest causal inference tools of all. An incredible way to get rid of bias which is sadly seldom known by Data Scientist: the **Frisch-Waugh-Lovell theorem**. But in order to do so, you must first quickly review a bit of regression theory.

Regression Theory

I don't intend to dive too deep into how linear regression is constructed and estimated. However, a little bit of theory will go a long way in explaining its power in causal inference. First of all, regression solves the best linear prediction problem. Let β^* be a vector of parameters:

$$\beta^* = \underset{\beta}{argmin}\ E\left[\left(Y_i - X_i'\beta\right)^2\right]$$

Linear regression finds the parameters that minimize the mean squared error (MSE). If you differentiate it and set it to zero, you will find that the linear solution to this problem is given by

$$\beta^* = E\left[X_i \hat{a} X_i\right]^{-1} E\left[X_i \hat{a} Y_i\right]$$

You can estimate this beta using the sample equivalent:

$$\widehat{\beta} = \left(X \hat{a} X\right)^{-1} X \hat{a} Y$$

But don't take my word for it. If you are one of those that understand code better than formulas, try for yourself. In the following code, I'm using the algebraic solution to OLS to estimate the parameters of the model you just saw (I'm adding the intercept as the final variables, so the first parameter estimate will be $\widehat{\beta_1}$).

ASSIGN

I tend to use the method `.assign()` from Pandas quite a lot. If you are not familiar with it, it just returns a new data frame with the newly created columns passed to the method.

```
new_df = df.assign(new_col_1 = 1,
                   new_col_2 = df["old_col"] + 1)

new_df[["old_col", "new_col_1", "new_col_2"]].head()
```

```
   old_col  new_col_1  new_col_2
0        4          1          5
1        9          1         10
2        8          1          9
3        0          1          1
4        6          1          7
```

```
X = risk_data[["credit_limit", "wage", "credit_score1", "credit_score2"]].assign(intercep=1)
y = risk_data["default"]

def regress(y, X):
    return np.linalg.inv(X.T.dot(X)).dot(X.T.dot(y))

beta = regress(y, X)
beta
```

```
array([ 3.06252773e-06, -8.82159125e-05, -4.17472814e-05, -3.03928359e-04,
        4.03661277e-01])
```

If you look back a bit, you will see that these are the exact same numbers you get earlier, when estimating the model with the `ols` function from `statsmodels`.

Single Variable Linear Regression

The formula above is pretty general. However, it pays off to study the case where you only have one regressor. In causal inference, you often want to estimate the causal impact of a variable T on an outcome y. So, you use regression with this single variable to estimate this effect.

With a single regressor variable T, the parameter associated to it will be given by

$$\hat{\tau} = \frac{Cov(Y_i, T_i)}{Var(T_i)}$$

If T is randomly assigned, β_1 is the ATE. Importantly, with this simple formula, you can see what regression is doing. It's finding out how the treatment and outcome move together (as expressed by the covariance in the numerator) and scaling this by units of the treatment, which is achieved by dividing by the variance of the treatment.

> **NOTE**
>
> You can also tie this to the general formula. Covariance is intimately related to dot products, so you can pretty much say that $\boldsymbol{X\hat{a}X}$ takes the role of the denominator in the covariance/variance formula, while $\boldsymbol{X\hat{a}y}$ takes the role of the numerator.

Multivariate Linear Regression

Turns out there is another way to see multivariate linear regression, beyond the general formula you saw earlier. This other way shed some light into what regression is doing.

If you have more than one regressor, you can extend the one variable regression formula to accommodate that. Let's say those other variables are just auxiliary and that you are truly interested in estimating the parameter τ associated to T.

$$y_i = \beta_0 + \tau T_i + \beta_1 X_{1i} + \ldots + \beta_k X_{ki} + u_i$$

τ can be estimated with the following formula

$$\hat{\tau} = \frac{Cov(Y_i, \widetilde{T_i})}{Var(\widetilde{T_i})}$$

where $\widetilde{T_i}$ is the residual from a regression of all other covariates $X_{1i} + ... + X_{ki}$ on T_i.

Now, let's appreciate how cool this is. It means that the coefficient of a multivariate regression is the bivariate coefficient of the same regressor **after accounting for the effect of other variables in the model**. In causal inference terms, τ is the bivariate coefficient of T after having used all other variables to predict it.

This has a nice intuition behind it. If you can predict T using other variables, it means it's not random. However, you can make T look as good as random once you control for other available variables. To do so, you can use linear regression to predict it from the other variables and then take the residuals of that regression $\widetilde{T}$. By definition, $\widetilde{T}$ cannot be predicted by the other variables X that you've already used to predict T. Quite elegantly, $\widetilde{T}$ is a version of the treatment that is not associated (uncorrelated) with any other variable in X.

I know this is a mouthful, but it is just amazing. In fact, it is already the work of the FWL theorem that I promised to teach you. So don't worry if you didn't quite get this multivariate regression part, as you are about to review it in a much more intuitive and visual way.

Frisch-Waugh-Lovell Theorem and Orthogonalization

FWL-style orthogonalization is the **first major debiasing technique you have at your disposal**. It's a simple yet powerful way to make non experimental data look as if the treatment has been randomized. FWL is mostly about linear regression, FWL-style orthogonalization has been expanded to work in more general contexts, as you'll see in part III.

Frisch-Waugh-Lovell theorem states that a multivariate linear regression model can be estimated all at once or in three separate steps. For example, you can regress `default` on `credit_limit`, `wage`, `credit_score1`, `credit_score2`, just like you already did,

```
model = smf.ols('default ~ credit_limit + wage + credit_score1  + credit_score2', data=risk_data).fit()
model.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025
Intercept	0.4037	0.009	46.939	0.000	0.387
credit_limit	3.063e-06	1.54e-06	1.987	0.047	4.16e-08
wage	-8.822e-05	6.07e-06	-14.541	0.000	-0.000
credit_score1	-4.175e-05	1.83e-05	-2.278	0.023	-7.77e-05
credit_score2	-0.0003	1.52e-05	-20.055	0.000	-0.000

But, according to FWL, you can also break down this estimation into

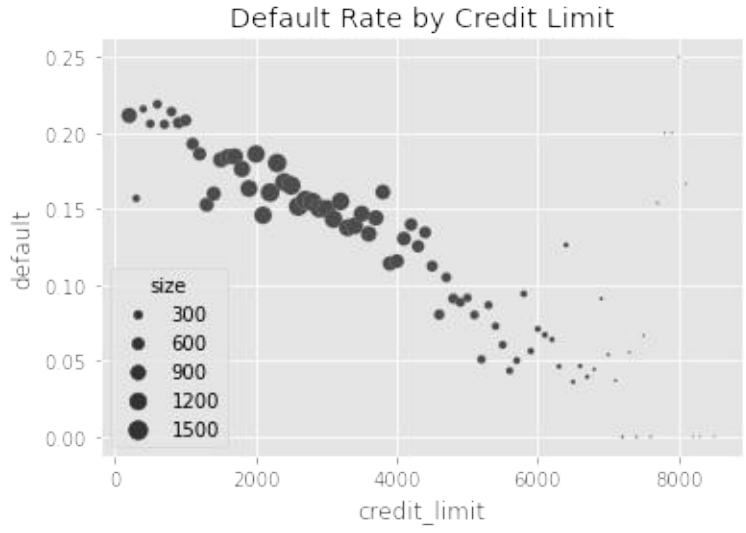
1. A debiasing step, where you regress the treatment T on confounders X and obtain the treatment residuals $\widetilde{T} = T - \widehat{T}$.
2. A denoising step, where you regress the outcome Y on the confounder variables X and obtain the outcome residuals $\widetilde{Y} = Y - \widehat{Y}$
3. An outcome model where you regress the outcome residual $\widetilde{Y}$ on the treatment residual $\widetilde{T}$ to obtain an estimate for the causal effect of T on Y.

Not surprisingly, this is just a restatement of the formula you just saw in the Multivariate Linear Regression section. The FWL theorem states an equivalence in estimation procedures with regression models. It also says that you can isolate the debiasing component of linear regression, which is the first step outlined above.

To get a better intuition on what is going on, let's break it down step by step.

Debiasing Step

Recall that, initially, due to confounding bias, your data looked something like this, with default trending downwards with credit line.



According to the FWL theorem, you can debias this data by fitting a regression model to predict the treatment, credit limit, from the confounders. Then, you can take the residual from this model: $\widetilde{line}_i = line_i - \widehat{line}_i$. This residual can be viewed as a version of the treatment which is uncorrelated with the variables used in the debiasing model. That's because, by definition, the residual is orthogonal to the variables that generated the predictions.

This process will make $\widetilde{line}$ centered around zero. Optionally, you can add back the average treatment, $\overline{line}$:

$$\widetilde{line}_i = line_i - \widehat{line}_i + \overline{line}$$

This is not necessary for debiasing, but it puts $\widetilde{line}$ in the same range as the original $line$, which is better for visualization purposes.

```
debiasing_model = smf.ols(
    'credit_limit ~ wage + credit_score1  + credit_score2',
    data=risk_data
).fit()

risk_data_deb = risk_data.assign(
    # for visualization, avg(T) is added to the residuals
    credit_limit_res=debiasing_model.resid + risk_data["credit_limit"].mean()
)
```

If you now run a simple linear regression, where you regress the outcome, $risk$, on the debiased or residualized version of the treatment, $\widetilde{line}$, you'll already get the effect of credit limit on risk while controlling for the confounders used in the debiasing model. Notice how the parameter estimate you get for β_1 here is exactly the same as the one you got earlier by running the complete model, where you've included both treatment and confounders.

```
model_w_deb_data = smf.ols('default ~ credit_limit_res', data=risk_data_deb).fit()
model_w_deb_data.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025
Intercept	0.1421	0.005	30.001	0.000	0.133
credit_limit_res	3.063e-06	1.56e-06	1.957	0.050	-4.29e-09

NOTE

The fact that the you only need to rezidualize the treatment sugest a simpler way of rewrtigin the regression coeficient formula. Istead of using the covariance of Y and the residual $\tilde{T}$ over the variance of $\tilde{T}$, you can use

$$\widehat{\beta_1} = \frac{\sum (T - \overline{T})\, y_i}{\sum (T - \overline{T})^2}$$

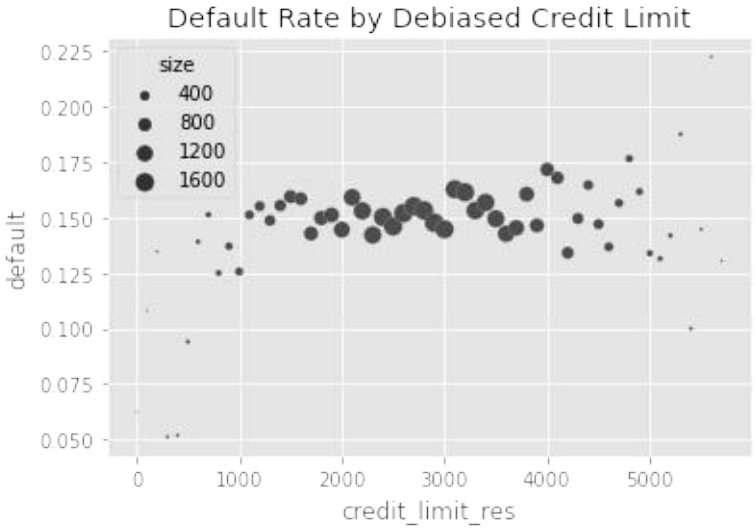
in the single variable regression case or

$$\widehat{\beta_1} = \frac{\sum \tilde{T} y_i}{\sum \tilde{T}^2}$$

in the more general case

There is a difference, though. Look at the p-value. It is a bit higher than what you got earlier. That's because you are not applying the denoising step, which is responsible for reducing variance. Still, with only the debiasing step, you can already get the unbiased estimate of the causal impact of credit limit on risk, given that all the confounders were included in the debiasing model.

You can also visualize what is going on by plotting the debiased version of credit limit against default rate. You'll see that the relationship is no longer downward sloping, like we had before when the data was biased.



Denoising Step

While the debiasing step is crucial to estimate the correct causal effect, the denoising step is also nice to have, although not as important. It won't change the value of your treatment effect estimate, but it will reduce its variance. In this stept, you'll regress the outcome on the covariates that are not the treatment. Then, you'll get the residual for the outcome $\widetilde{default}_i = default_i - \widehat{default}_i$.

Once again, for better visualization, you can add the average default rate to the denoised default for better visualization purposes:

$$\widetilde{default}_i = default_i - \widehat{default}_i + \overline{default}$$

```
denoising_model = smf.ols(
    'default ~ wage + credit_score1  + credit_score2',
    data=risk_data_deb
).fit()

risk_data_denoise = risk_data_deb.assign(
    default_res=denoising_model.resid + risk_data_deb["default"].mean()
)
```

Standard Error of the Regression Estimator

Since we are talking about noise, I think it is a good time to see how to compute the regression standard error. The SE of the regression parameter estimate is given by the following formula:

$$SE\left(\widehat{\beta}\right) = \frac{\sigma(\hat{\epsilon})}{\sigma\left(\widetilde{T}\right)\sqrt{n - DF}},$$

where $\hat{\epsilon}$ is the residual from the regression model and DF is the model's degree of freedom (number of parameters estimated by the model). If you prefer to see this in code, here it is:

```
model_se = smf.ols(
    'spend ~ wage + credit_score1  + credit_score2',
    data=spend_data
).fit()

print("SE regression:", model_se.bse["wage"])


model_wage_aux = smf.ols(
    'wage ~ credit_score1 + credit_score2',
    data=spend_data
).fit()

# subtract the degrees of freedom - 4 model parameters - from N.
se_formula = np.std(model_se.resid)/(np.std(model_wage_aux.resid) * np.sqrt(len(spend_data) - 4))
print("SE formula:   ", se_formula)


SE regression: 0.007587699648670252
SE formula:    0.007587699648670258
```

This formula is nice because it gives you further intuition about regression in general and the denoising step in particular. First, the numerator tells you that the better you can predict the outcome, the smaller the residuals will be and, hence the lower the variance of the estimator. This is very much what the denoising step is all about. It also tells you that if the treatment explains the outcome a lot, its parameter estimate will also have a smaller standard error.

Interestingly, the error is also inversely proportional to the variance of the (residualised) treatment. This is also intuitive. If the treatment varies a lot, it will be easier to measure its impact. You'll see more about this in the Good and Bad Control section, at the end of this chapter.

The standard error formula can also be useful if you plan to design an experiment where you care to measure the effect as the parameter estimate from a regression. This is a good idea if the treatment you

want to randomize is continuous. Notice that the standard error formula can be approximated by

$$SE \approx \frac{\sigma(y)}{\sigma(T)\sqrt{n-2}}$$

This approximation is conservative in the case of a single variable regression model, since $\sigma(y) \geq \sigma(\hat{e})$, because the treatment might explain a bit of the outcome. Then, you can take this and plug in the sample size calculation formula, from Chapter 2. But notice that designing this test has the additional complexity of choosing a sampling distribution from T, which can also affect the standard error via $\sigma(T)$.

Final Outcome Model

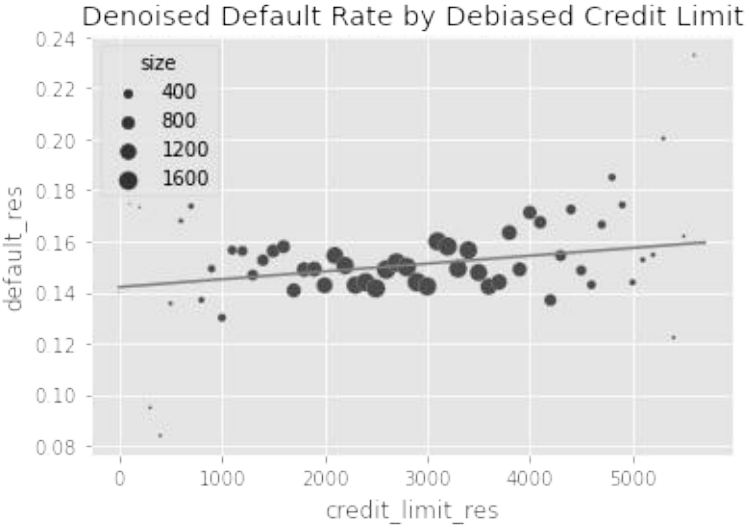
With both residuals, $\tilde{Y}$ and $\tilde{T}$, you can run the final step outlined by the FWL theorem: just regress $\tilde{Y}$ on $\tilde{T}$.

```
model_w_orthogonal = smf.ols('default_res ~ credit_limit_res', data=risk_data_denoise).fit()
model_w_orthogonal.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025
Intercept	0.1421	0.005	30.458	0.000	0.133
credit_limit_res	3.063e-06	1.54e-06	1.987	0.047	4.17e-08

Notice how the parameter estimate for the treatment is exactly the same as the one you got in both the debiasing step and when running the regression model with credit limit plus all the other covariates. Additionally, the standard error and p-value are now also just like when you first ran the model with all the variables included. This is the effect of the denoising step.

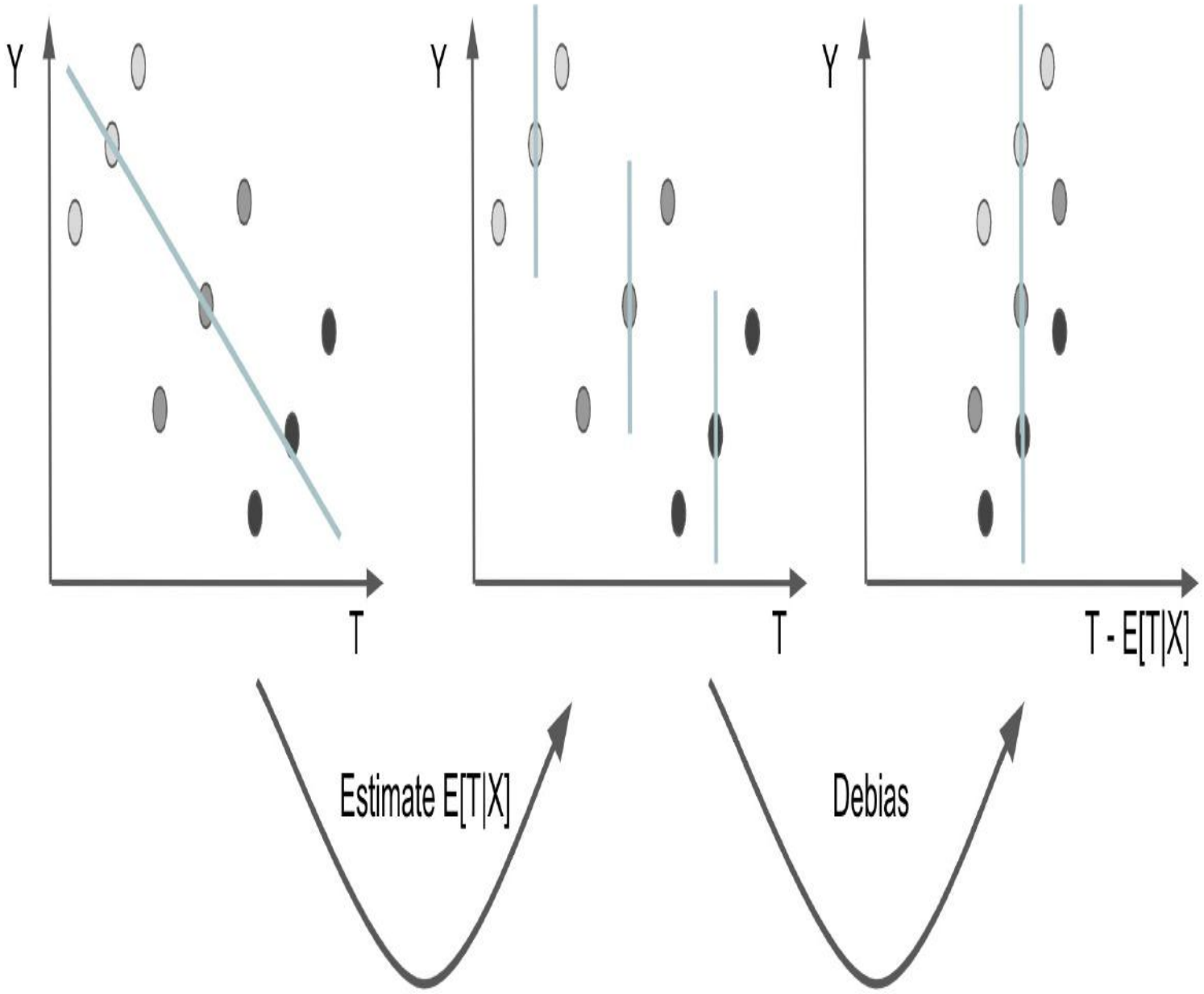
Of course, you can also plot the relationship between the debiased treatment with the denoised outcome, alongside the predictions from the final model to see what is going on.



FWL Summary

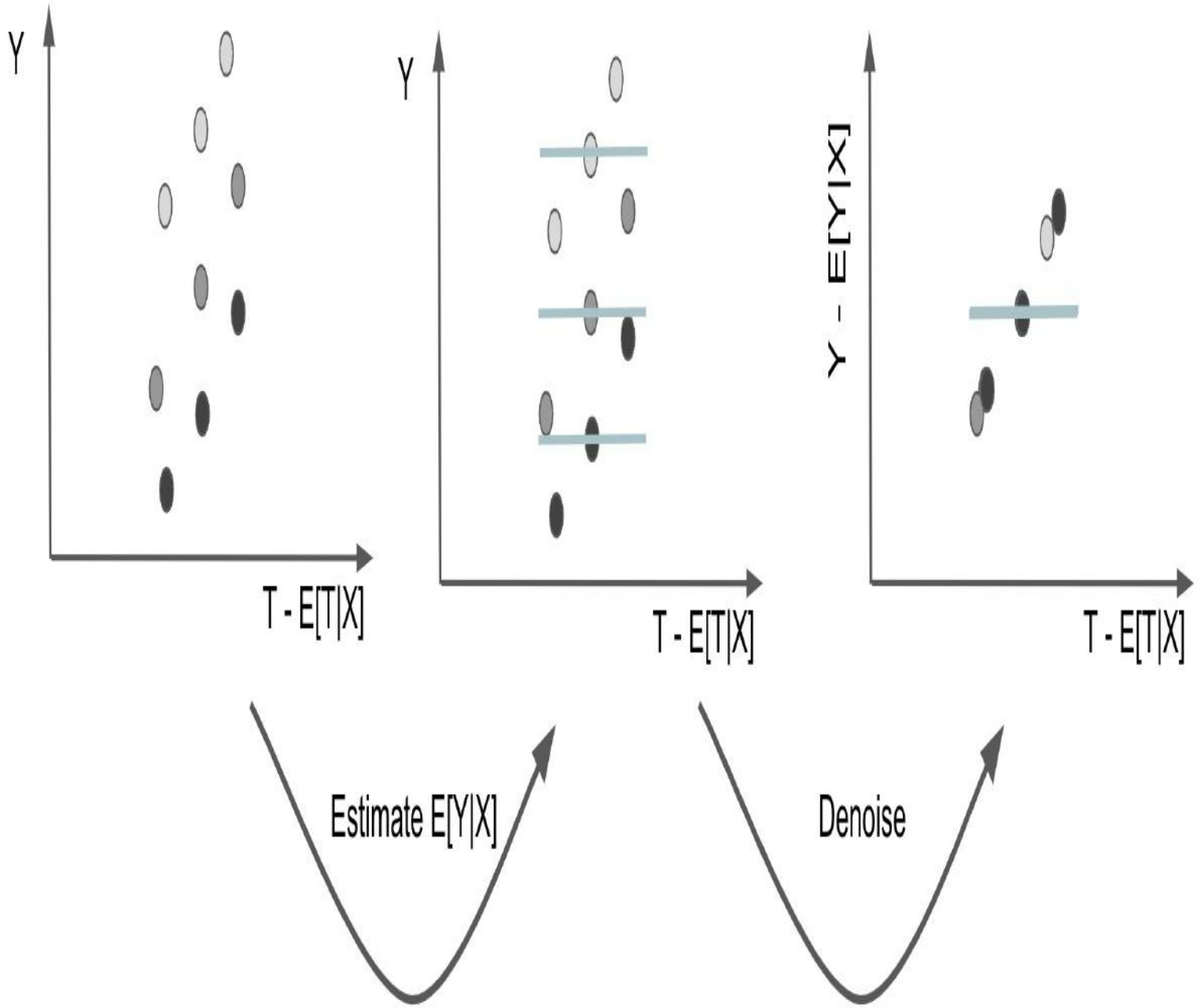
I don’t know if you can already tell, but I really like illustrative figures. Even if they don’t reflect any real data, they can be quite useful to visualize what is going on behind some fairly mathy technical concept. It wouldn’t be different with FWL. So to summarize, consider that you want to estimate the relationship between a treatment T and an outcome Y but you have some confounder X. You plot the treatment on the x-axis, the outcome on the y-axis and the confounder as the color dimension. You initially see a negative slope between treatment and outcome, but you have strong reasons (some domain knowledge) to believe that the relationship should be positive, so you decide to debias the data.

To do that, you first estimate $E[T|X]$ using linear regression. Then, you construct a debiased version of the treatment: $T - E[T|X]$. With this debiased treatment, you can already see the positive relationship you were hoping to find. But you still have a lot of noise.

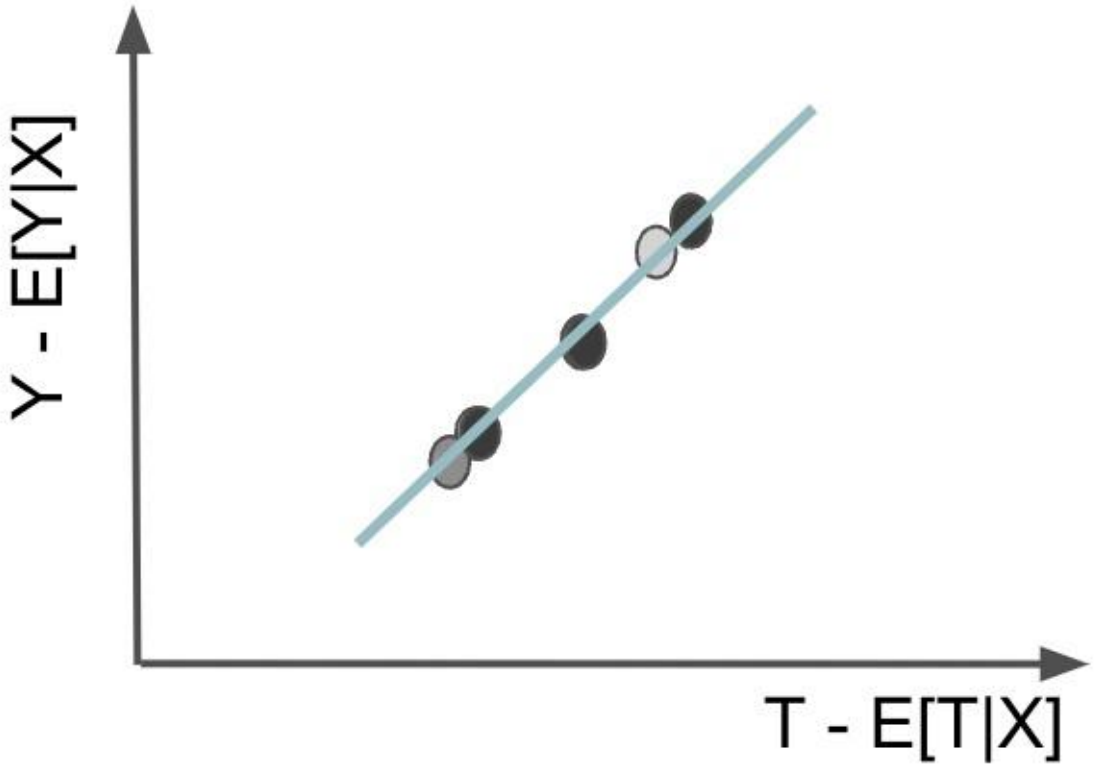


To deal with the noise, you estimate $E[Y|X]$, also using a regression model. Then, you construct a denoised version of the outcome: $Y - E[T|X]$. You can view this denoised outcome as the outcome

after you've accounted for all the variance in it that was explained by X. If X explains a lot of the variance in Y, the denoised outcome will be less noisy, making it easier to see the relationship you really care about: that between T and Y.



Finally, after both debiasing and denoising, you can clearly see a positive relationship between T and Y. All there is left to do is fit a final model to this data.



This final regression will have the exact same slope as one where you would regress Y on T and X at the same time.

> **DEBIASING AND THE INTERCEPT**
>
> One word of caution though. In causal inference, you are mostly concerned with the slope of this regression line, since the slope is a linear approximation to the effect of the continuous treatment, $\frac{\partial}{\partial t} E[y|t]$. But, if you also care about the intercept - for instance, if you are trying to do counterfactual predictions - you should know that debiasing and denoising makes the intercept equal to zero.

Regression as an Outcome Model

Through this section I emphasized how regression works mostly by orthogonalizing the treatment. However, you can also see regression as a potential outcome imputation technique. Suppose that the treatment is binary. If regression of Y on X in the control population ($T = 0$) yields good approximation to $E[Y_0|X]$, than, you can use that model to impute Y_0 and estimate the ATT:

$$ATT = \frac{1}{N_1} \sum 1\left(T_i = 1\right)\left(Y_i - \hat{\mu}_0\left(X_i\right)\right),$$

where N_1 is the number of treated units. A similar argument can be made to show that if regression on the treated units can model $E[Y_1|X]$, you can use it to estimate the average effect on the untreated. If you put these two arguments side by side, you can estimate the ATE as follows

$$ATE = \frac{1}{N} \sum \left(\hat{\mu}_1 \left(X_i \right) - \hat{\mu}_0 \left(X_i \right) \right)$$

This estimator will impute both potential outcomes for all units. It is equivalent to regressing Y on both X and T and reading the parameter estimate on T.

Alternatively, you can impute just the potential outcomes that are missing:

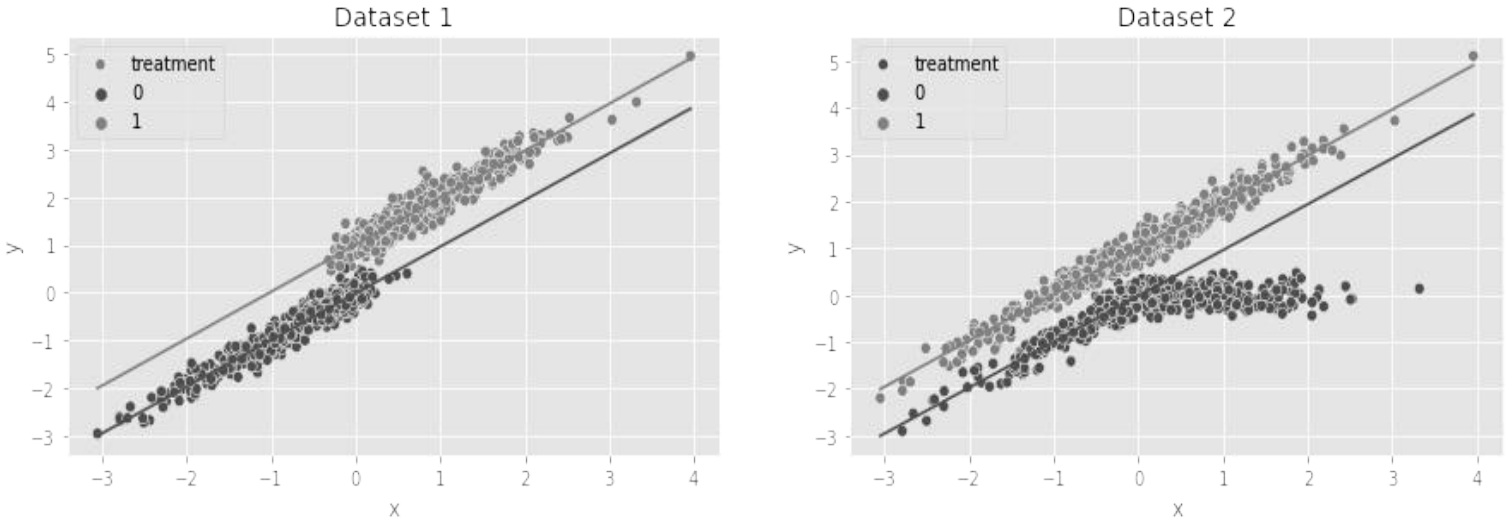
$$ATE = \frac{1}{N} \sum \left(1 \left(T_i = 1 \right) \left[Y_i - \hat{\mu}_0 \left(X_i \right) \right] + 1 \left(T_i = 0 \right) \left[\hat{\mu}_1 \left(X_i \right) - Y_i \right] \right)$$

When T is continuous, this is a bit harder to conceptualize, but you can understand regression as imputing the whole treatment response function, which involves inputting the potential outcomes $Y(t)$ as if it was a line.

The fact that regression can work if can either correctly estimate $E[T|X]$ for orthogonalization or correctly estimate the potential outcomes $E[Y_t|X]$ grants it doubly robust properties, something you'll explore further in Chapter 5. Seeing regression through this less will also be important when you learn about Difference in Differences in part IV.

Positivity and Extrapolation

Since regression models the potential outcome as a parametric function, it allows for extrapolation outside the region where you have data on all treatment levels. This can be a blessing or a curse. It all depends if the extrapolation is reasonable. For example, consider that you have Dataset 1 to estimate a treatment effect. Dataset 1 has low overlap, with high no controls for high values of a covariate x and no treated for low values of that same covariate. If you use regression to estimate the treatment effect on this data, it will input Y_0 and Y_1 as shown by the lines in the first plot below. This is fine, so long as the same relationship between Y_0 and x you've fitted in the control for low levels of x is also valid for high values of x and that the Y_1 you've fitted on the treated also extrapolates well to low levels of x. Generally, if the trends in the outcome where you do have overlap look similar across the covariate space, a small level of extrapolation becomes less of an issue.



However, too much extrapolation is alway dangerous. Let's suppose you've estimated the effect on

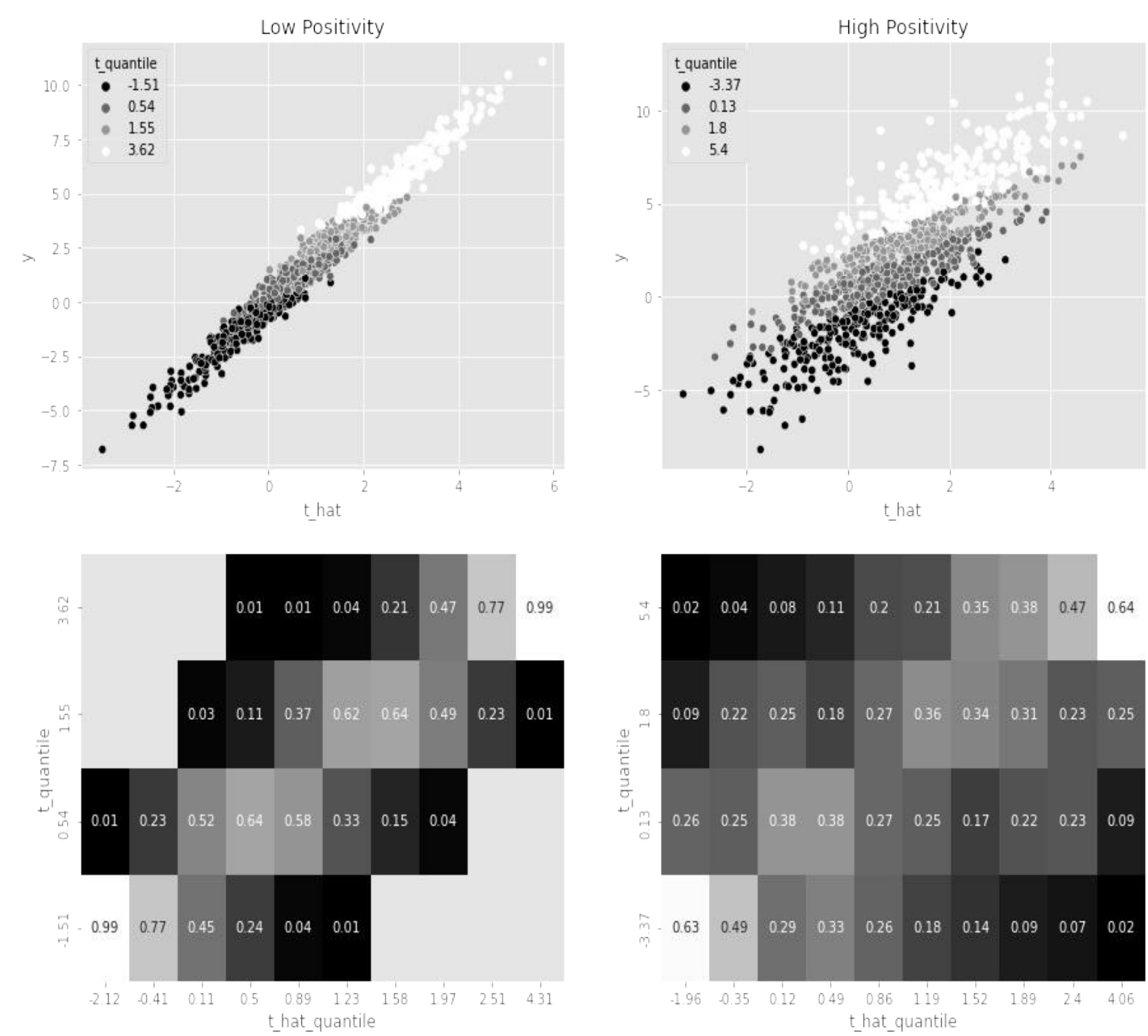
Dataset 1, but then you collect more data, now randomizing the treatment. On this new data, call it Dataset 2, you see that the effect gets larger and larger for positive values of x. Consequently, if you evaluate your previous fit on this new data, you'll realize that you grossly underestimated the true effect of the treatment.

You might be asking yourself how you can evaluate the level of extrapolation of your regression? In most cases, you'll have more than one covariate, so you won't be able to produce plots like the ones above. Fortunately, there is a pretty easy trick you can use.

First, fit a model to predict treatment from the covariates. That is exactly the debiased model you saw in the FWL context. You can think about the predictions of this model, $\hat{T}$, as performing dimensionality reduction on X. $\hat{T}$ contains all the information from X which is relevant to determine T. For this reason, if positivity is high, this model will have a relatively low predictive power. Moreover, you should be able to find many - and ideally all - values of the treatment for any segment of $\hat{T}$.

This suggests two visual checks of positivity based on $\hat{T}$. First, you can plot the outcome by $\hat{T}$ and color code the dots by the treatment level. Second, you can plot the density of $\hat{T}$ by treatment values. If the treatment is continuous, you can discretize it, also using quantiles.

Here is an example of those two visualizations on two datasets with different levels of overlap. Both dataset have the same data generating process, with two confounders x. The difference between them is that the first has low positivity, which is controlled by the variance in the treatment. The second dataset has a much higher degree of overlap. To get $\hat{T}$, I've regressed T on the two confounders, x_1 and x_2.



First, look at the scatter plot in the low positivity case. Notice how you can't find any data for low values of the treatment and high values of $\widehat{T}$ and vice versa. On the density plot right below, for the lowest quantile of $\widehat{T}$, you can see that 99% of the samples received the lowest quantile of the treatment. Again, this shows lack of overlap, as there is no data on the region of high treatment and low $\widehat{T}$.

Contrast this with the results on the high positivity data. Here, the treatment is more dispersed at all levels of $\widehat{T}$. Positivity is not perfect here, as there are still fewer points with high treatment for low values of $\widehat{T}$. But, on the density plot, you can see that all tiles have some data, even if just a few data points.

Now, these checks will only get you so far. The goal here is to be transparent about the degree to which your data complies with the positivity assumption. But they won't tell you how much extrapolation is too much. Unfortunately, I can't give you a clear rule for that. I just recommend that you make this call intentionally and for that you need to know how much overlap you have.

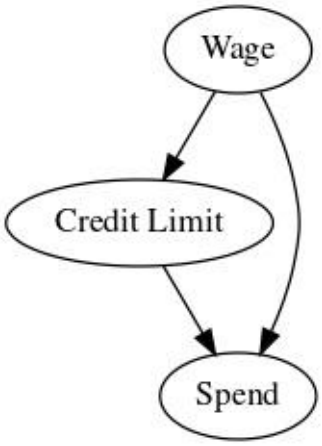
Non-Linearities in Linear Regression

Up until this point, the treatment response curve seemed pretty linear. It looked like an increase in credit line caused a constant increase in risk, regardless of the credit line level. Going from a line of 1000 to 2000 seemed to increase risk about the same as going from a line of 2000 to 3000. However, you are likely to encounter situations where this won't be the case. As an example, consider the same data as before, but now your task is to estimate the causal effect of credit limit on credit card spend.

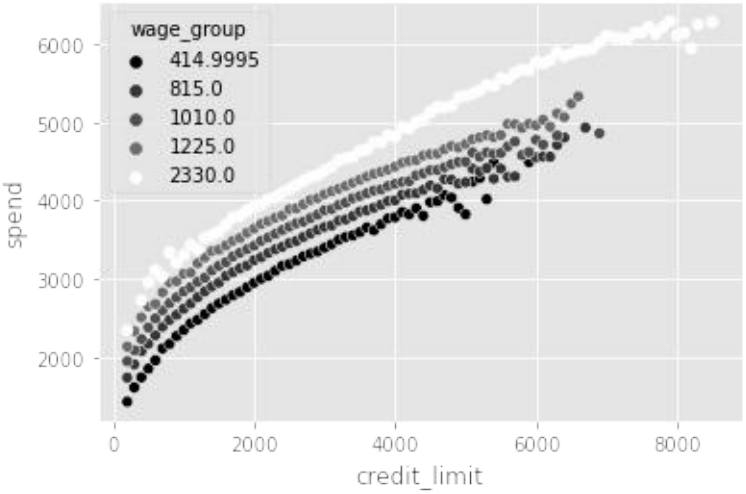
```
spend_data.head()
```

	wage	educ	exper	married	credit_score1
0	950.0	11	16	1	500.0
1	780.0	11	7	1	414.0
2	1230.0	14	9	1	586.0
3	1040.0	15	8	1	379.0
4	1000.0	16	1	1	379.0

And for the sake of simplicity, let's consider that the only confounder you have here is wage (assume that is the only information the bank considers when deciding the credit limit). So, the causal graph of this process looks something like this:



This means that you have to condition on wage to identify the effect of credit lines on spending. If you want to use orthogonalization to estimate this effect, you can say that you need to debias credit lines by regressing it on wage and getting the residuals. Nothing new so far. But there is a catch. If you plot spend by credit lines for multiple wage levels, you can clearly see that the relationship is not linear.

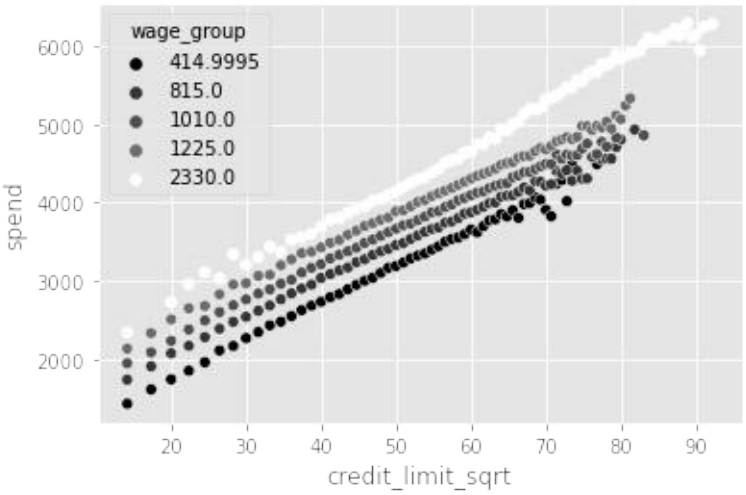


Rather, the treatment response curve seems to have some sort of concavity to it: the higher the credit limit, the lower the slope of this curve. Or, in causal inference language, since slopes and causal effects are intimately related, you can also say that the effect of lines on spend diminishes as you increase lines: going from a line of 1000 to 2000 increases spend more than going from 2000 to 3000.

Linearizing the Treatment

To deal with that, you first need to transform the treatment into something that does have a linear relationship with the outcome. For instance, you know that the relationship seems concave, so you can try to apply some concave function to lines. Some good candidates to try out are the Log function, the square root function or any function that takes credit lines to the power of a fraction.

In this case, let's try the square root:



Now we are getting somewhere! Notice how the square root of the credit line seems to have a linear relationship with spend. It's definitely not perfect. If you look very closely, you can still see some curvature. But it might just do for now.

I'm sad to say that this process is fairly manual. You have to try a bunch of stuff and see what linearizes the treatment the better. Once you find something that you are happy with, you can apply it when running a linear regression model. In this example, it means that you will be estimating the model

$$spend_i = \beta_0 + \beta_1 \sqrt{line}_i + e_i$$

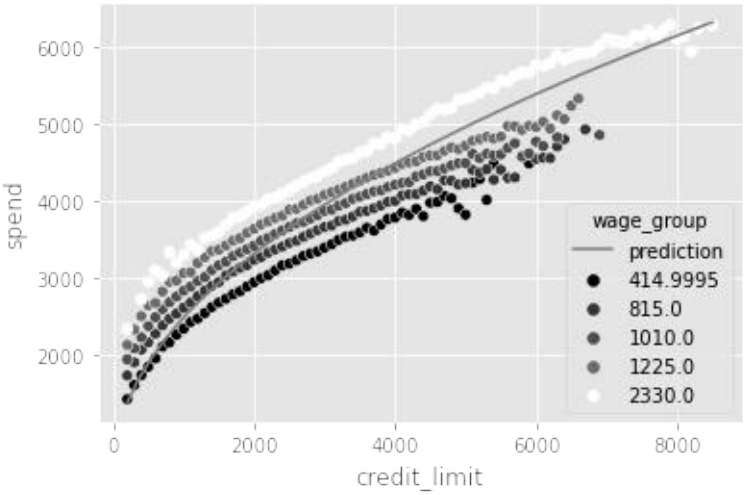
and your causal parameter is β_1.

Here, you can see how to estimate the model above using `statsmodels`. Is as simple as transforming the treatment with the function that you want, directly on the formula string:

```
model_spend = smf.ols('spend ~ np.sqrt(credit_limit)', data=spend_data).fit()
model_spend.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025
Intercept	493.0044	6.501	75.832	0.000	480.262
np.sqrt(credit_limit)	63.2525	0.122	519.268	0.000	63.014

But you are not done yet. Recall that wage is confounding the relationship between credit lines and spend. You can see this by plotting the predictions from the above model against the original data. Notice how its slope is probably upward biased. That's because more wage cause both spend and credit lines to increase.



If you include wage in the model,

$$spend_i = \beta_0 + \beta_1 \sqrt{line}_i + \beta_2 wage_i + e_i$$

and estimate β_1 again, you get an unbiased estimate of the effect of lines on spend (assuming wage is the only confounder, of course). Notice how this estimate is smaller than the one you got earlier. That is because including wage in the model fixed the upward bias.

```
model_spend = smf.ols('spend ~ np.sqrt(credit_limit)+wage', data=spend_data).fit()
model_spend.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025
Intercept	383.5002	2.746	139.662	0.000	378.118
np.sqrt(credit_limit)	43.8504	0.065	672.633	0.000	43.723
wage	1.0459	0.002	481.875	0.000	1.042

Non-Linear FWL and Debiasing

As to how the FWL theorem works with non linear data, it is exactly like before, but now you have to apply the non-linearity first. That is, you can decompose the process of estimating a non-linear model with linear regression as follows:

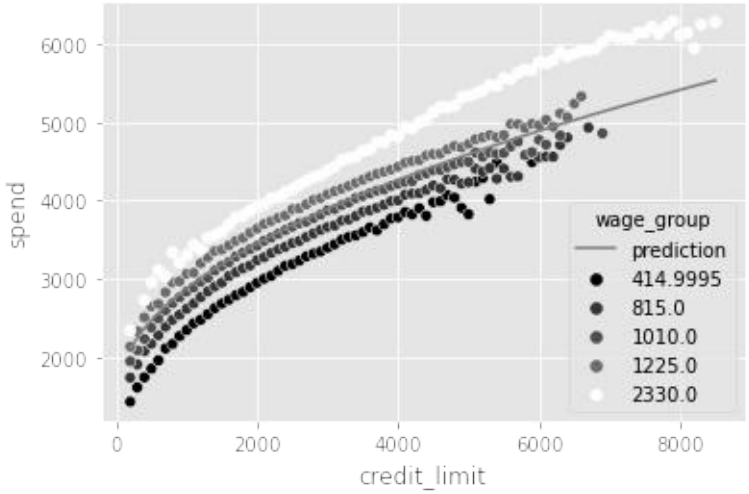
1. Find a function F which linearizes the relationship between T and Y
2. A debiasing step, where you regress the the treatment $F(T)$ on confounder variables X and obtain the treatment residuals $\widetilde{F(T)} = F(T) - \widehat{F(T)}$.
3. A denoising step, where you regress the outcome Y on the confounder variables X and obtain the outcome residuals $\widetilde{Y} = Y - \widehat{Y}$
4. An outcome model where you regress the outcome residual $\widetilde{Y}$ on the treatment residual $\widetilde{F(T)}$ to obtain an estimate for the causal effect of $F(T)$ on Y.

In our example, F is the square root, so here is how you can apply the FWL theorem considering the non-linearity. (Notice that I'm also adding $\overline{F(lines)}$ and $\overline{spend}$ to the treatment and outcome residuals, respectively. This is optional, but it makes for better visualization).

```
debias_spend_model = smf.ols(f'np.sqrt(credit_limit) ~ wage', data=spend_data).fit()
denoise_spend_model = smf.ols(f'spend ~ wage', data=spend_data).fit()

credit_limit_sqrt_deb = debias_spend_model.resid + np.sqrt(spend_data["credit_limit"]).mean()
spend_den = denoise_spend_model.resid + spend_data["spend"].mean()

spend_data_deb = (spend_data
                  .assign(credit_limit_sqrt_deb=credit_limit_sqrt_deb,
                          spend_den = spend_den))

final_model = smf.ols(f'spend_den ~ credit_limit_sqrt_deb', data=spend_data_deb).fit()

final_model.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025
Intercept	1493.6990	3.435	434.818	0.000	1486.966
credit_limit_sqrt_deb	43.8504	0.065	672.640	0.000	43.723

Not surprisingly, the estimate you get here for β_1 is the exact same as the one you got earlier, by running the full model including both the wage confounder and the treatment. Also, if you plot the prediction from this model against the original data, you can see that it is not upward biased like before. Instead, it goes right through the middle of the wage groups.



Regression for Dummies

Regression and orthogonalization are great and all, but ultimately you have to make an independence assumption. You have to assume that treatment looks as good as randomly assigned, when some covariates are accounted for. This can be quite a stretch. It's very hard to know if all confounders have been included in the model. For this reason, it makes a lot of sense for you to push for randomized experiments as much as you can. For instance, in our example, it would be great if the credit limit was randomized, as that would make it pretty straightforward to estimate its effect on default rate and customer spend. The thing is that this experiment would be incredibly expensive. You would be giving random credit lines to very risky customers, which would probably default and cause a huge loss.

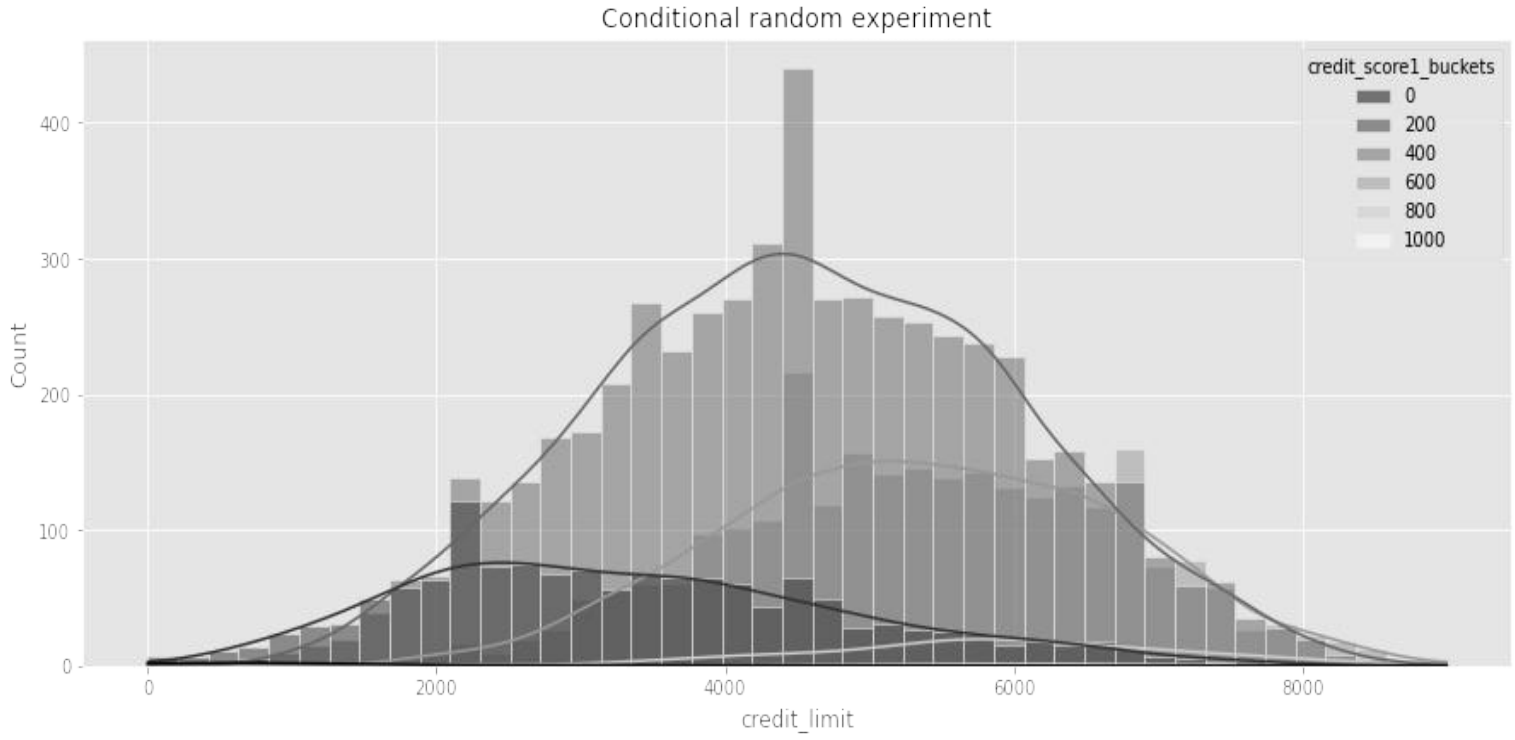
Conditionally Random Experiments

The way around this conundrum is not the ideal randomized controlled trial, but it is the next best thing: **Stratified or Conditionally Random Experiments**. Instead of crafting an experiment where lines are completely random and drawn from the same probability distribution, you instead create multiple local experiments, where you draw samples from different distributions, depending on customer covariates. For instance, you know that the variable `credit_score1` is a proxy for customer risk. So you can use it to create groups of customers that are more or less riskier, dividing them into buckets of similar `credit_score1`. Then, for the high risk bucket - with low `credit_score1` - you randomize credit lines by sampling from a distribution with a lower average; for low risk customers - with high `credit_score1` - you randomize credit lines by sampling from a distribution with a higher average.

```
risk_data_rnd.head()
```

	wage	educ	exper	married	credit_score1
0	890.0	11	16	1	490.0
1	670.0	11	7	1	196.0
2	1220.0	14	9	1	392.0
3	1210.0	15	8	1	627.0
4	900.0	16	1	1	275.0

Plotting the histogram of credit limit by `credit_score1_buckets`, you can see lines were sampled from different distributions. The buckets with higher score - low risk customers - have a histogram skewed to the left, with higher lines. The groups with risker customers - low score - have lines drawn from a distribution that is skewed to the right, with lower lines. This sort of experiment explores credit lines that are not too far from what is probably the optimal line, which lowers the cost of the test to a more manageable amount.



Now, this doesn't mean that conditionally random experiments are better than completely random experiments. They sure are cheaper, but they add a tone of extra complexity. For this reason, if you opt for a conditionally random experiment, for whatever reason, try to keep it as close to a completely random experiment as possible. This means that

1. The lower the number of groups, the easier it will be to deal with the conditional random test. In this example we only have 5 groups, since we divided `credit_score1` in buckets of 200 and the score goes from 0 to 1000. Combining different groups with different treatment distribution adds a tone of complexity, so sticking to fewer groups is a good idea

2. The bigger the overlap in the treatment distributions across groups, the easier your life will be. This has to do with the positivity assumption. In this example, if the high risk group had zero probability of receiving high lines, you would have to rely on dangerous extrapolations to know what would happen if they were to receive those high lines.

Notice that if you crank this two rules of thumb to their maximum, you get back a completely random experiment, which means both of them carry a trade-off: the lower the number of groups and the higher the overlap, the easier it will be to read the experiment, but it will also be more expensive, and vice versa.

> **NOTE**
>
> Stratified experiment can also be used as a tool to minimize variance and to ensure balance between treatment and control on the stratified variables. But in those applications, the treatment distribution is designed to be the same across all groups or strata.

Dummy Variables

The neat thing about conditionally random experiments is that the conditional independence assumption is much more plausible, since you know lines were randomly assigned given a categorical variable of your choice. The downside is that a simple regression of the outcome on the treated will yield a biased estimate. For example, here is what happens when you estimate the model, without the confounder included

$$default_i = \beta_0 + \beta_1 lines_i + e_i$$

```
model = smf.ols("default ~ credit_limit", data=risk_data_rnd).fit()
model.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025
Intercept	0.1369	0.009	15.081	0.000	0.119
credit_limit	-9.344e-06	1.85e-06	-5.048	0.000	-1.3e-05

As you can see, the causal parameter estimate, $\widehat{\beta_1}$, is negative, which makes no sense here. Higher credit lines probably do not decrease a customer's risk. What happened is that, in this data, due to the way the experiment was designed, lower risk customers - those with high `credit_score1` - got, on average, higher lines.

To adjust for that, you need to include in the model the group within which the treatment is randomly assigned. In this case, you need to control for `credit_score1_buckets`. Even though this group is represented as a number, it is actually a categorical variable: it represents a group. So, the way to control for the group itself is to create **dummy variables**. A dummy is a binary column for a group. It is 1 if the

customer belongs to that group and 0 otherwise. As a customer can only be from one group, at most one dummy column will be one, with all the others being zero. If you come from a machine learning background, you might know this as one-hot encoding. They are exactly the same thing.

In Pandas, you can use the `pd.get_dummies` function to create dummies. Here, I'm passing the column which represents the groups, `credit_score1_buckets` and saying that I want to create dummy columns with the suffix `sb` (for score bucket). Also, notice how I'm dropping the first dummy, that of the bucket 0 to 200. That's because one of the dummy columns is redundant. If I know that all the other columns are zero, the one that I dropped must be 1.

```
risk_data_dummies = (risk_data_rnd
                     .join(pd.get_dummies(risk_data_rnd["credit_score1_buckets"],
                                          prefix="sb",
                                          drop_first=True)))
risk_data_dummies.head()
```

	wage	educ	exper	married	...
0	890.0	11	16	1	...
1	670.0	11	7	1	...
2	1220.0	14	9	1	...
3	1210.0	15	8	1	...
4	900.0	16	1	1	...

5 rows × 14 columns

Once you have the dummy columns, you can add them to your model and estimate β_1 again.

$$default_i = \beta_0 + \beta_1 lines_i + \boldsymbol{\theta} \boldsymbol{G}_i + e_i$$

Now, you'll get a much more reasonable estimate, which is at least positive, indicating that more credit lines increase risk of default.

```
model = smf.ols("default ~ credit_limit + sb_200+sb_400+sb_600+sb_800+sb_1000",
                data=risk_data_dummies).fit()
model.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025
Intercept	0.2253	0.056	4.000	0.000	0.115
credit_limit	4.652e-06	2.02e-06	2.305	0.021	6.97e-07
sb_200	-0.0559	0.057	-0.981	0.327	-0.168
sb_400	-0.1442	0.057	-2.538	0.011	-0.256
sb_600	-0.2148	0.057	-3.756	0.000	-0.327
sb_800	-0.2489	0.060	-4.181	0.000	-0.366
sb_1000	-0.2541	0.094	-2.715	0.007	-0.438

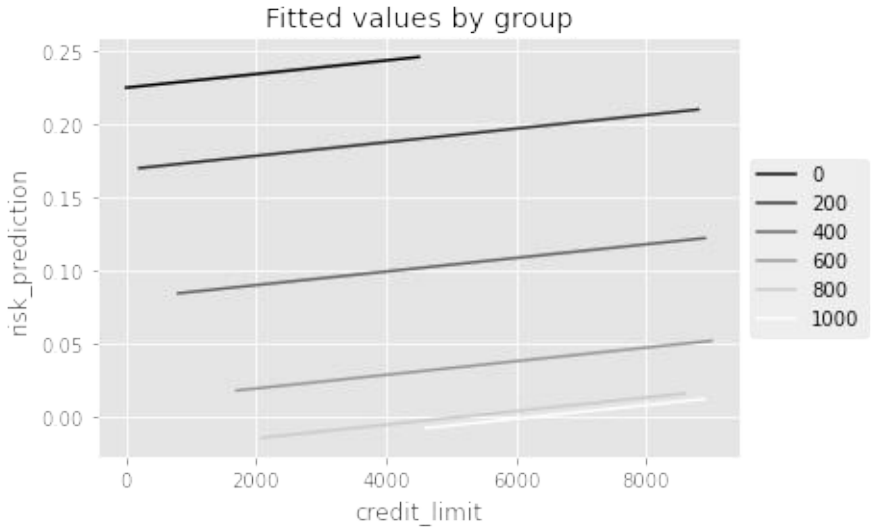
Additionally, intercept itself can be interpreted as the intercept for the group which had its dummy variable omitted, `sb_100`. Here, the regression lines cross zero at 0.22 for that group. As for the dummy parameters, they are group level intercept. For example, the parameter associated with `sb_200` tells you that the intercept for that group is $\beta_0 + \theta_{200}$, that is, $0.2253 - 0.0559 = 0.1694$.

I'm only showing you how to create dummies so you know what happens under the hood. This will be very useful if you have to implement that sort of regression in some other framework that is not in Python. In Python, if you are using `statsmodels`, the `C()` function in the formula can do all of that for you.

```
model = smf.ols("default ~ credit_limit + C(credit_score1_buckets)",
                data=risk_data_rnd).fit()
model.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.02
Intercept	0.2253	0.056	4.000	0.000	0.115
C(credit_score1_buckets)[T.200]	-0.0559	0.057	-0.981	0.327	-0.16
C(credit_score1_buckets)[T.400]	-0.1442	0.057	-2.538	0.011	-0.25
C(credit_score1_buckets)[T.600]	-0.2148	0.057	-3.756	0.000	-0.32
C(credit_score1_buckets)[T.800]	-0.2489	0.060	-4.181	0.000	-0.36
C(credit_score1_buckets)[T.1000]	-0.2541	0.094	-2.715	0.007	-0.43
credit_limit	4.652e-06	2.02e-06	2.305	0.021	6.97e

Finally, notice how you only have one slope parameter. Adding dummies to control for confounding gives one intercept per group, but the same slope for all groups. We'll discuss this shortly, but this slope will be a variance weighted average of the regression in each group. If you plot the model's predictions for each group, you can clearly see that you have one line per group, but all of them have the same slope.



Saturated Regression Model

Remember how I've started the chapter highlighting the similarities between regression and a conditional average? I showed you how running a regression with a binary treatment is exactly the same as comparing the average between treated and control group. Now, since dummies are binary columns, the parallel also applies here. If you take our conditionally random experiment data and give it to

someone that is not as versed in regression as you are, their first instinct will probably be to simply segment the data by `credit_score1_buckets` and estimate the effect in each group separately.

```
def regress(df, t, y):
    return smf.ols(f"{y}~{t}", data=df).fit().params[t]

effect_by_group = (risk_data_rnd
                   .groupby("credit_score1_buckets")
                   .apply(regress, y="default", t="credit_limit"))
effect_by_group
```

```
credit_score1_buckets
0      -0.000071
200     0.000007
400     0.000005
600     0.000003
800     0.000002
1000    0.000000
dtype: float64
```

This would give an effect by group, which means you also have to decide how to average them out. A natural choice would be to do a weighted average, where the weights are the size of each group.

```
group_size = risk_data_rnd.groupby("credit_score1_buckets").size()
ate = (effect_by_group * group_size).sum() / group_size.sum()
ate
```

```
4.490445628748722e-06
```

Of course, you can do the exact same thing with regression, by running what is called a saturated model. You can interact the dummies with the treatment to get an effect for each dummy defined group. In this case, because the first dummy is removed, the parameter associated with `credit_limit` actually represents the effect in the omitted dummy group, `sb_100`. Notice how it is the exact same number as the one estimated above for the `credit_score1_buckets` group 0 to 200: -0.000071.

```
model = smf.ols("default ~ credit_limit * C(credit_score1_buckets)",
                data=risk_data_rnd).fit()
model.summary().tables[1]
```

	coef	std err	t	P>|t|
Intercept	0.3137	0.077	4.086	0.000
C(credit_score1_buckets)[T.200]	-0.1521	0.079	-1.926	0.054
C(credit_score1_buckets)[T.400]	-0.2339	0.078	-3.005	0.003
C(credit_score1_buckets)[T.600]	-0.2957	0.080	-3.690	0.000
C(credit_score1_buckets)[T.800]	-0.3227	0.111	-2.919	0.004
C(credit_score1_buckets)[T.1000]	-0.3137	0.428	-0.733	0.464
credit_limit	-7.072e-05	4.45e-05	-1.588	0.112
credit_limit:C(credit_score1_buckets)[T.200]	7.769e-05	4.48e-05	1.734	0.083
credit_limit:C(credit_score1_buckets)[T.400]	7.565e-05	4.46e-05	1.696	0.090
credit_limit:C(credit_score1_buckets)[T.600]	7.398e-05	4.47e-05	1.655	0.098
credit_limit:C(credit_score1_buckets)[T.800]	7.286e-05	4.65e-05	1.567	0.117
credit_limit:C(credit_score1_buckets)[T.1000]	7.072e-05	8.05e-05	0.878	0.380

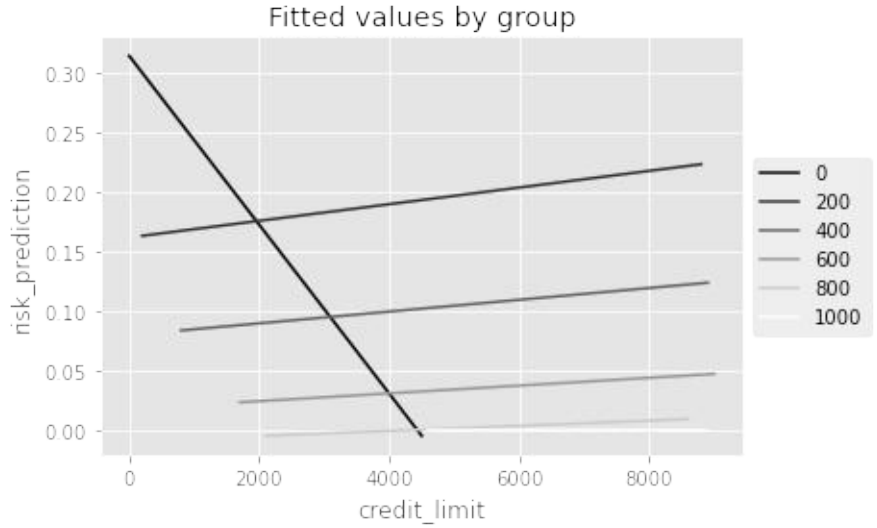
The interaction parameters are interpreted in relation to the effect in the first (omitted) group. So, if you sum the parameter associated with `credit_limit` with other interaction terms, you can see the effects for each group estimated with regression. They are exactly the same as estimating one effect per group.

```
(model.params[model.params.index.str.contains("credit_limit:")]
 + model.params["credit_limit"]).round(9)
```

```
credit_limit:C(credit_score1_buckets)[T.200]     0.000007
credit_limit:C(credit_score1_buckets)[T.400]     0.000005
credit_limit:C(credit_score1_buckets)[T.600]     0.000003
credit_limit:C(credit_score1_buckets)[T.800]     0.000002
credit_limit:C(credit_score1_buckets)[T.1000]    0.000000
dtype: float64
```

Plotting this model's prediction by group will also show that, now, it is as if you are fitting a separate regression for each group. Each line will have not only a different intercept, but also a different slope.

Not only that, the saturated model has more parameters (degrees of freedom), which will also mean more variance. If you look at the following plot, you'll see a line with negative slope, which doesn't make sense in this context. However, that slope is not statistically significant. It is probably just noise due to a small sample in that group.



Regression as Variance Weighted Average

But if both the saturated regression and calculating the effect by group give you the exact same thing, there is a very important question you might be asking yourself. When you run the model "default ~ credit_limit + C(credit_score1_buckets)", without the interaction term, you get a single effect. Only one slope parameter. Importantly, if you look back, that effect estimate is different from the one you got by estimating an effect per group and averaging the results using the group size as weights. So, somehow, regression is combining the effects from different groups. And the way it does it is not a sample size weighted average. So what is it then?

Again, the best way to answer this question is by using some very illustrative simulated data. Here, let's simulate data from two different groups. Group 1 has a size of 1000 and an average treatment effect of 1. Group 2 has a size of 500 and an average treatment effect of 2. Additionally, the standard deviation of the treatment in group 1 is 1 and 2 in group 2.

```
np.random.seed(123)

# std(t)=1
t1 = np.random.normal(0, 1, size=1000)
df1 = pd.DataFrame(dict(
    t=t1,
    y=1*t1, # ATE of 1
    g=1,
))

# std(t)=2
t2 = np.random.normal(0, 2, size=500)
df2 = pd.DataFrame(dict(
    t=t2,
    y=2*t2, # ATE of 2
    g=2,
))
```

```
df = pd.concat([df1, df2])
df.head()
```

	t	y	g
0	-1.085631	-1.085631	1
1	0.997345	0.997345	1
2	0.282978	0.282978	1
3	-1.506295	-1.506295	1
4	-0.578600	-0.578600	1

If you estimate the effects for each group separately and average the results with the group size as weights, you should get an ATE of around 1.33, $(1 * 1000 + 2 * 500)/1500$.

```
effect_by_group = df.groupby("g").apply(regress, y="y", t="t")
ate = (effect_by_group * df.groupby("g").size()).sum() / df.groupby("g").size().sum()
ate
```

```
1.333333333333333
```

But if you run a regression of `y` on `t` while controlling for the group, you get a very different result. Now, the combined effect is closer to the effect of group 2, even though group 2 has half the sample of group 1.

```
model = smf.ols("y ~ t + C(g)", data=df).fit()
model.params
```

```
Intercept    0.024758
C(g)[T.2]    0.019860
t            1.625775
dtype: float64
```

The reason for this is that regression doesn't combine the group effects by using the sample size as weights. Instead, it uses weights that are proportional to the variance of the treatment in each group. Regression prefers groups where the treatment varies a lot. This might seem odd at first, but if you think about it, it makes a lot of sense. If the treatment doesn't change much within a group, how can you be sure of its effect? If it changes a lot, its impact on the outcome will be more evident.

To summarize, if you have multiple groups where the treatment is randomized inside each group, the conditionality principle states that the effect is a weighted average of the effect inside each group.

$$ATE = E\left\{ \left(\frac{\partial}{\partial t} E\left[Y_i | T = t, Group_i \right] \right) w\left(Group_i \right) \right\}$$

Depending on the method, you'll have different weights. With regression, $w\left(Group_i\right) \propto \sigma^2\left(T\right)|Group$, but you can also choose to manually weight the group effects using the sample size as the weight: $w\left(Group_i\right) = N_{Group}$.

> **SEE ALSO**
>
> Knowing this difference is key to understanding what is going on behind the curtains with regression. The fact that regression weights the group effects by variance is something that even the best researchers need to be constantly reminded of. In 2020, the econometric field went through a renaissance regarding the Diff-in-Diff method (you'll see more about it in Part IV). At the center of the issue was regression not weighting effects by sample size. If you want to learn more about it, I recommend checking it out the paper *Difference-in-Differences with Variation in Treatment Timing*, by Andrew Goodman-Bacon. Or just weight until get to part IV

De-Meaning and Fixed Effects

You just saw how to include dummy variables in your model to account for different treatment assignments across groups. But it is with dummies where the FWL theorem really shines. If you have a tone of groups, adding one dummy for each is not only tedious, but also computationally expensive. You would be creating lots and lots columns which are mostly zero. You can solve this pretty easily by applying FWL and understanding how regression orthogonalizes the treatment when it comes to dummies.

You already know that the debiasing step in FWL involves predicting the treatment from the covariates, in this case, the dummies.

```
model_deb = smf.ols("credit_limit ~ C(credit_score1_buckets)",
                    data=risk_data_rnd).fit()
model_deb.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.02
Intercept	1173.0769	278.994	4.205	0.000	626.1
C(credit_score1_buckets)[T.200]	2195.4337	281.554	7.798	0.000	1643
C(credit_score1_buckets)[T.400]	3402.3796	279.642	12.167	0.000	2854
C(credit_score1_buckets)[T.600]	4191.3235	280.345	14.951	0.000	3641
C(credit_score1_buckets)[T.800]	4639.5105	291.400	15.921	0.000	4068
C(credit_score1_buckets)[T.1000]	5006.9231	461.255	10.855	0.000	4102

And since dummies work basically as group averages, you can pretty much see that, with this model, you are predicting exactly that: if `credit_score1_buckets=0`, you are predicting the average line for the group `credit_score1_buckets=0`; if `credit_score1_buckets=1`, you are predicting the average line for the group `credit_score1_buckets=1` (which is given by summing the intercept to the coefficient for that group $\mathbf{1173.0769 + 2195.4337 = 3368.510638}$) and so on and so forth. Those are exactly the group averages:

```
risk_data_rnd.groupby("credit_score1_buckets")["credit_limit"].mean()
```

```
credit_score1_buckets
0       1173.076923
200     3368.510638
400     4575.456498
600     5364.400448
800     5812.587413
1000    6180.000000
Name: credit_limit, dtype: float64
```

Which means that if you want to residualize the treatment, you can do that in a much simpler and effective way. First, calculate the average treatment for each group.

```
risk_data_fe = (risk_data_rnd
                .assign(credit_limit_avg = lambda d: (d
                                                      .groupby("credit_score1_buckets")
                                                      ["credit_limit"]
                                                      .transform("mean"))))
```

Then, to get the residuals, subtract that group average from the treatment. Since this approach subtracts

the average treatment, it is sometimes referred to as de-meaning the treatment. If you want to do that inside the regression formula, you must wrap the mathematical operation around `I(...)`.

```
model = smf.ols("default ~ I(credit_limit-credit_limit_avg)",
                data=risk_data_fe).fit()
model.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025
Intercept	0.0935	0.003	32.121	0.000	0.088
I(credit_limit - credit_limit_avg)	4.652e-06	2.05e-06	2.273	0.023	6.4e-07

If you go back a bit in the book, you will see that the parameter estimate you got here is exactly the same as the one you got when adding dummies to your model. That's because, mathematically speaking, they are equivalent. This idea goes by the name of **fixed effects**, since you are controlling for anything that is fixed within a group. It comes from the literature of causal inference with temporal structures (panel data), which you'll explore more in Part IV.

Another idea from the same literature is to include the average treatment by group in the regression model (from Mundlak's, 1978). Regression, will rezidualise the treatment from the additional variables included, so the effect here is pretty much the same

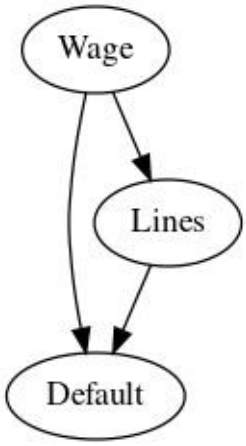
```
model = smf.ols("default ~ credit_limit + credit_limit_avg",
                data=risk_data_fe).fit()
model.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025
Intercept	0.4325	0.020	21.418	0.000	0.393
credit_limit	4.652e-06	2.02e-06	2.305	0.021	6.96e-07
credit_limit_avg	-7.763e-05	4.75e-06	-16.334	0.000	-8.69e-05

Omitted Variable Bias: Confounding Through the Lens of Regression

I hope I've made myself very clear in Chapter 3 when I said that common causes - confounder - will bias the estimated relationship between the treated and the outcome. That is why you need to account for them by, for example, including them in a regression model. However, regression has its own particular take on confounding bias. Sure, everything said up until now still holds. But regression allows you to be more precise about the confounding bias. For example, let's say you want to estimate the

effect of credit lines on default and that wage is the only confounder



You know you should be estimating the model which includes the confounder,

$$default_i = \beta_0 + \beta_1 lines_i + \beta_2 wage_i + e_i,$$

but instead you estimate a shorter model, where the confounder is omitted

$$default_i = \beta_0 + \beta_1 lines_i + e_i,$$

```
short_model = smf.ols("default ~ credit_limit", data=risk_data).fit()
short_model.params["credit_limit"]
```

```
-2.401961992596885e-05
```

If you do so, the parameter estimate will be biased. As you can see, it looks like higher credit lines causes default to go down, which is nonsense. But you know that already. What you don't know is that you can be precise about the size of that bias. With regression, you can say that the bias due to an omitted variable is **equal the effect in the model where it is included plus the effect of the omitted variable on the outcome times the regression of omitted on included**. Don't worry. I know this is a mouthfull, so let's digest it little by little. First, it means that simple regression of Y on T will be the true causal parameter τ, plus a bias term:

$$\frac{Cov(T,Y)}{Var(T)} = \tau + \beta'_{omitted}\delta_{omitted}$$

This bias term is the coefficient of the omitted confounder on the outcome, $\beta_{omitted}$, times the coefficient of regressing the omitted variable on the treatment, $\delta_{omitted}$. To check that, you obtain the biased parameter estimate you got earlier with the following code, which reproduces the omitted variable bias formula

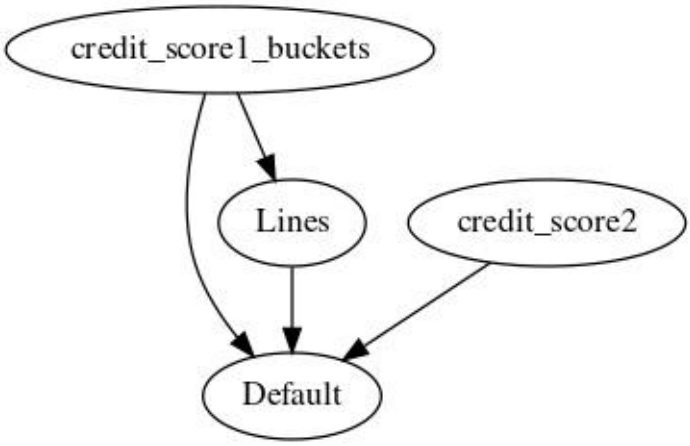
```
long_model = smf.ols("default ~ credit_limit + wage", data=risk_data).fit()
omitted_model = smf.ols("wage ~ credit_limit", data=risk_data).fit()

long_model.params["credit_limit"] + long_model.params["wage"]*omitted_model.params["credit_limit"]
```

```
-2.4019619925968762e-05
```

Neutral Controls

By now, you probably have a good idea about how regression adjusts for confounder variables. If you want to know the effect of the treatment T on Y while adjusting for confounders X, all you have to do is include X in the model. Alternatively, to get the exact same result, you can predict T from X, get the residuals and use that as a debiased version of the treatment. Regressing Y on those residuals will give you the relationship of T and Y while holding X fixed.

But what kind of variables should you include in X? Again, it's not because adding variables adjusts for them that you want to include everything you have in your regression model. As seen in the previous chapters, you don't want to include common effects (colliders) nor mediators, as those would induce selection bias. But in the context of regression, there are more types of controls you should know about. Controls that, at first, seem like they are innocuous, but are actually quite harmful. These controls are named neutral because they don't influence the bias in your regression estimate. But they can have severe implications in terms of variance. As you'll see, there is a bias-variance trade-off when it comes to including certain variables in your regression.



Should you include `credit_score2` in your model? If you don't include it, you'll get the same result you've been seeing all along. That result is unbiased, as you are adjusting for `credit_score1_buckets`. But, although you don't need to, look at what happens when you do include `credit_score2`. Compare the following results to the one you got earlier, which didn't include `credit_score2`. What changed?

```
model = smf.ols("default ~ credit_limit + C(credit_score1_buckets) + credit_score2",
                data=risk_data_rnd).fit()
model.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.02
Intercept	0.5576	0.055	10.132	0.000	0.450
C(credit_score1_buckets)[T.200]	-0.0387	0.055	-0.710	0.478	-0.14
C(credit_score1_buckets)[T.400]	-0.1032	0.054	-1.898	0.058	-0.21
C(credit_score1_buckets)[T.600]	-0.1410	0.055	-2.574	0.010	-0.24
C(credit_score1_buckets)[T.800]	-0.1161	0.057	-2.031	0.042	-0.22
C(credit_score1_buckets)[T.1000]	-0.0430	0.090	-0.479	0.632	-0.21
credit_limit	4.928e-06	1.93e-06	2.551	0.011	1.14e
credit_score2	-0.0007	2.34e-05	-30.225	0.000	-0.00

First, the parameter estimate on `credit_limit` became a bit higher. But, more importantly, the standard error decreases. That's because `credit_score2` is a good predictor of the outcome Y and it will contribute to the denoising step of linear regression. In the final step of FWL, because `credit_score2` was included, the variance in $\widetilde{Y}$ will be reduced, and regressing it on $\widetilde{T}$ will yield more precise results.

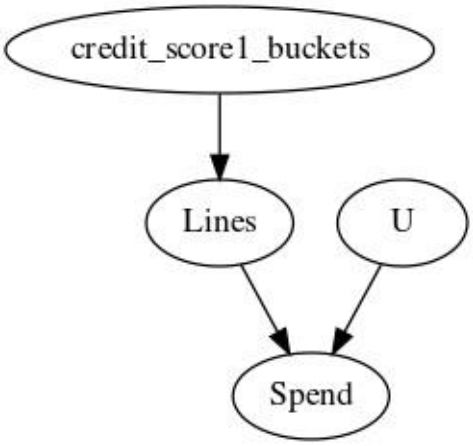
This is a very interesting property of linear regression. It shows that it can be used not only to control for confounders, but also to reduce noise. For example, if you have data from a properly randomized A/B test, you don't need to worry about bias. But you can still use regression as a noise reduction tool. Just include variables that are highly predictive of the outcome (and that don't induce selection bias).

> **NOTE**
>
> There are other noise reduction techniques out there. The most famous one being CUPED, which was developed by Microsoft researchers and it is widely used in tech companies. CUPED is very similar to just doing the denoising part of the FWL theorem.

Noise Inducing Control

Just like controls can reduce noise, they can also increase it. For example, consider again the case of a conditionally random experiment. But this time, you are interested in the effect of credit limit on spend, rather than on risk. Just like in the previous example, credit limit was randomly assigned, given `credit_score1`. But this time, let's say that `credit_score1` is not a confounder. It causes the

treatment, but not the outcome. The causal graph for this data generating process looks like this:



This means that you don't need to adjust for `credit_score1` to get the causal effect of credit limit on spend. A single variable regression model would do. Here, I'm keeping the square root function to account for the concavity in the treatment response function.

```
model = smf.ols("spend ~ np.sqrt(credit_limit)", data=spend_data_rnd).fit()
model.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025
Intercept	2153.2154	218.600	9.850	0.000	1723.723
np.sqrt(credit_limit)	16.2915	2.988	5.452	0.000	10.420

But, what happens if you do include `credit_score1_buckets`?

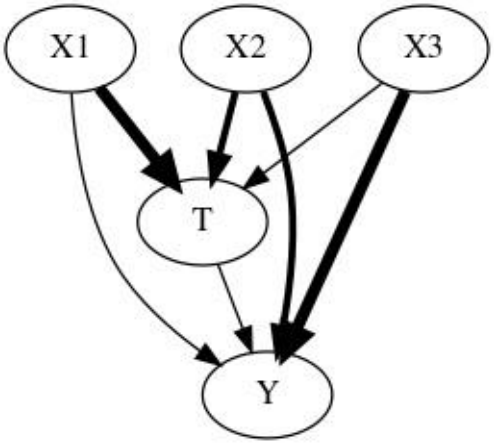
```
model = smf.ols("spend ~ np.sqrt(credit_limit) + C(credit_score1_buckets)",
                data=spend_data_rnd).fit()
model.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.02
Intercept	2367.4867	556.273	4.256	0.000	1274
C(credit_score1_buckets)[T.200]	-144.7921	591.613	-0.245	0.807	-130
C(credit_score1_buckets)[T.400]	-118.3923	565.364	-0.209	0.834	-122
C(credit_score1_buckets)[T.600]	-111.5738	570.471	-0.196	0.845	-123
C(credit_score1_buckets)[T.800]	-89.7366	574.645	-0.156	0.876	-121
C(credit_score1_buckets)[T.1000]	363.8990	608.014	0.599	0.550	-830.
np.sqrt(credit_limit)	14.5953	3.523	4.142	0.000	7.673

You can see that it increases the standard error, widening the confidence interval of the causal parameter. That is because, like you saw in the *Regression as Variance Weighted Average* section, OLS likes when the treatment has a high variance **in the treatment**. But if you control for a covariate which explains the treatment, you are effectively reducing its variance.

Feature Selection: A Bias-Variance Trade-Off

In reality, it's really hard to have a situation where a covariate causes the treatment but not the outcome. Most likely, you will have a bunch of confounders that cause both T and Y, but to different degrees. In the following example, X_1 is a strong cause of T but a weak cause of Y, X_3 is a strong cause of Y but a weak cause of T and X_2 is somewhere in the middle, as denoted by the thickness of each arrow.



In these situations, you can quickly be caught between a rock and a hard place. On one hand, if you want to get rid of all the biases, you must include all the covariates, after all, they are confounders

which need to be adjusted.

To see that, let's simulate data according to the causal graph above. Here, the true ATE is 0.5. If you try to estimate this effect while controlling for the confounders, the standard error of your estimate will be too high to conclude anything.

```
np.random.seed(123)

n = 100
(x1, x2, x3) = (np.random.normal(0, 1, n) for _ in range(3))
t = np.random.normal(10*x1 + 5*x2 + x3)

# ate = 0.05
y = np.random.normal(0.05*t + x1 + 5*x2 + 10*x3, 5)
df = pd.DataFrame(dict(y=y, t=t, x1=x1, x2=x2, x3=x3))

smf.ols("y~t+x1+x2+x3", data=df).fit().summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025
Intercept	0.2707	0.527	0.514	0.608	-0.775
t	0.8664	0.607	1.427	0.157	-0.339
x1	-7.0628	6.038	-1.170	0.245	-19.049
x2	0.0143	3.128	0.005	0.996	-6.195
x3	9.6292	0.887	10.861	0.000	7.869

If you know that one of the confounders is a strong predictor of the treatment and a weak predictor of the outcome, you can choose to drop it from the model. In this example, that would be X_1. Now, be warned! This will bias your estimate. But maybe this is a price worth paying if it also decreases variance significantly.

```
smf.ols("y~t+x2+x3", data=df).fit().summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025
Intercept	0.1889	0.523	0.361	0.719	-0.849
t	0.1585	0.046	3.410	0.001	0.066
x2	3.6095	0.582	6.197	0.000	2.453
x3	10.4549	0.537	19.453	0.000	9.388

The bottom line is that the more confounders you include (adjust for) in your model, the lower the bias in your causal estimate. However, if you include variables that are weak predictors of the outcome but strong predictors of the treatment, this bias reduction will come at a steep cost in terms of variance increase. Saying the same thing but differently, sometimes it is worth accepting a bit of bias in order to reduce variance. Also, you should be very aware that not all confounders are equal. Sure, all of them are common because of both T and Y. But if they explain the treatment too much and the outcome almost nothing, you should really consider dropping it from your adjustment. This is valid for regression, but it will also be true for other adjustment strategies, like propensity score weighting.

Unfortunately, how weak should the confounder be in terms of explaining the treatment to justify removing it is still an open question in causal inference. Still, it is worth knowing that this bias-variance trade-off exists, as it will help you understand and explain what is going on with your linear regression.

Key Ideas

This Chapter was about regression, but from a very different perspective than the one you usually see in machine learning books. Regression here is not a prediction tool. Notice how I didn't talk about R^2 not even once! Rather, regression is used here as a way to, primarily adjust for confounders and, sometimes, as a variance reduction technique.

The core of this chapter was orthogonalization as a means to make treatment look as good as randomly assigned if conditional independence holds. Formally, if $Y_t \perp T|X$, you can adjust for the confounding bias due to X by regressing T on X and obtaining the residuals. Those residuals can be seen as a debiased version of the treatment.

This approach was further developed using the Frisch-Waugh-Lovell Theorem, which states that a multivariate regression can be decomposed into the following steps:

1. A debiasing step, where you regress the treatment T on confounders X and obtain the treatment residuals $\widetilde{T} = T - \widehat{T}$;
2. A denoising step, where you regress the outcome Y on the confounder variables X and obtain the outcome residuals $\widetilde{Y} = Y - \widehat{Y}$;
3. An outcome model where you regress the outcome residual $\widetilde{Y}$ on the treatment residual $\widetilde{T}$ to obtain an estimate for the causal effect of T on Y.

Everything else in the chapter follows from this theorem. Be it nonlinear treatment response functions, understanding how regression with categorical variables implements a weighted average or the role of good and bad controls in regression.

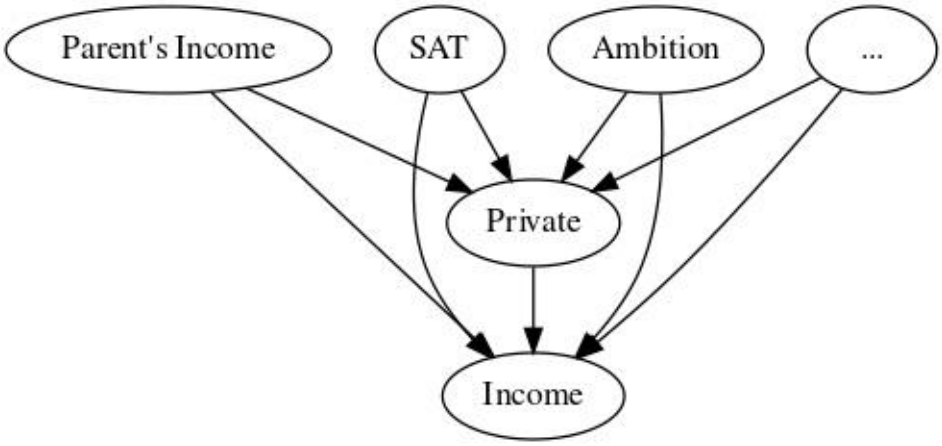
Other Examples

These examples are used to solidify what you learned in this chapter. They use concepts and techniques you just learned and are presented in a fast paced manner. As you read through them, try to link what they tell you with what you learned in this chapter. Also, use them to try to generalize the ideas outlined

here to applications in your own work.

Public or Private Schools?

In the book Mastering Metrics, Angrist and Pischke show how regression can be used to adjust for the bias when estimating the impact of going to private schools on one's income. Graduate's of private school often make more money than those of public school, but it's hard to say how much of this relationship is causal. For instance, your parents' income might confound the relationship, as kids of richer families are both more likely to go to private schools and to earn more. Similarly, since private schools are very selective, it could be that they take in only the students who would already be better off anyway.



So much so that a naive regression of income on a private school dummy is almost sure to return a positive effect. In other words, estimating the following model would give you a positive and significant $\widehat{\beta_1}$

$$income_i = \delta_0 + \beta_1 private + e_i$$

What Angrist and Pischke show, however, is that if you adjust for SAT score and parent's income, the measured impact decreases. That is, if you augment the model with these two variables, your $\widehat{\beta_1}$ will be smaller, compared to the one you would get with the short model above.

$$income_i = \delta_0 + \beta_1 private + \delta_1 SAT_i + \delta_2 ParentInc_i + e_i$$

Still, after running a regression like the one above, the effect of private schools remained positive and significant, at least in the dataset used by the authors. However, one final set of controls managed to make the relationship insignificant. The authors included the average SAT of the school the student's applied to (regardless of them being accepted). This can be interpreted as a proxy for ambition.

$$income_i = \delta_0 + \beta_1 private + \delta_1 SAT_i + \delta_2 ParentInc_i + \delta_3 AvgSATSchool_i + e_i$$

Once they added the ambition proxy controls, the estimated $\widehat{\beta_1}$ became insignificant. Interestingly, keeping only those controls and dropping the SAT and parents income controls still resulted in a non-significant estimate. This indicates that, given your ambition level, it doesn't matter if you go to a public or private school, at least in terms of your earnings.

Marketing Mix Modeling

Measuring the impact of advertising on sales is very hard, since you usually can't randomize who gets to see your ads. One popular alternative to randomization in the advertising industry is the technique called Marketing Mix Modeling (MMM). Despite the fancy name, MMMs are just regressions of sales on marketing strategies indicators and some confounders. For example, let's say you want to know the effect of your budget on TV, social media and search advertising on your product's sales. You can run a regression model where each unit i is a day.

$$Sales_i = \delta_0 + \beta_1 TV_i + \beta_2 Social_i + \beta_3 Search_i + \delta_1 CompetitorSales_i + \delta_2 Mont + \delta_3 Trend_i + e_i$$

To account for the fact that you might have increased your marketing budget on a good month, you can adjust for this confounder by including additional controls in your regression. For example, you can include your competitor's sales, a dummy for each months and a trend variable.

Chapter 5. Propensity Score

In Chapter 4, you saw how to adjust for confounders using linear regression. More broadly, you learned the principle of debiasing through orthogonalization. That is one of the most useful bias adjusting techniques you have at your disposal. But there is a second one. It's now time to learn: propensity weighting. Much like with orthogonalization, you'll model the treatment assignment mechanism. But instead of building residuals, you'll use that model's for reweighting the data. Moreover, you'll also see how to combine what you learned in Chapter 4 with propensity weighting to achieve what is called doubly-robustness.

The content of this chapter is better suited for when you have binary or discrete treatments. Still, I'll show an extension that allows you to use propensity weighting for continuous treatment.

The Impact of Management Training

A common phenomena in tech companies is for talented individual contributors (ICs) to branch out to a management track. But because management often requires a very different skill set than the ones which made them talented ICs, this transition is often far from easy. It comes at a high personal cost, not only for the new managers, but for those they manage.

Hoping to make this transition less painful, a big multinational company decided to invest in manager training for its new managers. Also, to measure the effectiveness of the training, the company tried to randomly select managers into this program. The idea was to compare an engagement score for the employees whose managers got enrolled in the program with the engagement of those whose managers didn't. With proper randomization, this simple comparison would give the average treatment effect of the training.

Unfortunately, things are not that simple. Some managers didn't want to go to the training, so they simply didn't show up. Others managed to get the training even without being assigned to receive it. The result is that what was to be a randomized study ended up looking a lot like an observational one.

> **NON-COMPLIANCE**
>
> People not getting the treatment they are intended to is called **non-compliance**. You will see more about this when we talk about Instrumental Variables.

Now, as an analyst who has to read this data, you'll have to make treated and untreated comparable by adjusting for confounders. To do that, you are given data on the company managers along with some covariates that describe them. The treatment variable is `intervention` and your outcome of interest is `engagement_score`, which is the average standardized engagement score for that manager's employees. You hope that, by controlling for the other variables, you manage to reduce or even eliminate the bias when estimating the causal relationship between management training and employee engagement.

```
import pandas as pd

df.head()
```

	departament_id	intervention	engagement_score	...	last
0	76	1	0.277359	...	0.64
1	76	1	-0.449646	...	0.64
2	76	1	0.769703	...	0.64
3	76	1	-0.121763	...	0.64
4	76	1	1.526147	...	0.64

5 rows × 10 columns

> **NOTE**
>
> This dataset was adapted from the one used in the study *Estimating Treatment Effects with Causal Forests: An Application*, by Susan Athey and Stefan Wager.

Adjusting with Regression

Before moving on to the propensity weighting, let's use regression to adjust for the confounders. This will serve as a benchmark for you to know if the results you'll get later make sense - that is, if they are in line with what you get with linear regression.

For starters, if you simply compare treated and control groups, this is what you'll get:

```
import statsmodels.formula.api as smf
```

```
smf.ols("engagement_score ~ intervention", data=df).fit().summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025
Intercept	-0.2347	0.014	-16.619	0.000	-0.262
intervention	0.4346	0.019	22.616	0.000	0.397

But then again, this result is probably biased, since the treatment was not entirely random. To reduce this bias, you can adjust for the covariates you have in your data, estimating the following model

$$engagement_i = \tau T_i + \theta X_i + e_i,$$

where X are all the confounders you have plus a constant column for the intercept. Additionally, gender and role are all categorical variables, so you have to wrap them in `C()` inside your OLS formula.

```
model = smf.ols("""engagement_score ~ intervention
+ tenure + last_engagement_score + deparament_score
+ n_of_reports + C(gender) + C(role)""", data=df).fit()

print("ATE:", model.params["intervention"])
print("95% CI:", model.conf_int().loc["intervention", :].values.T)
```

```
ATE: 0.26933207925982916
95% CI: [0.23498424 0.30367992]
```

Notice that the effect estimate here is considerably smaller than the one you got earlier. This is some indication of positive bias, which means that managers whose employees were already more engaged are more likely to have participated in the manager training program. OK, enough with the preamble. Let's see what propensity weighting is all about.

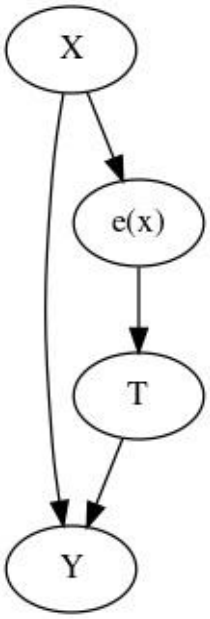
Propensity Score

Propensity weighting revolves around the concept of a propensity score, which itself comes from the realization that you don't need to directly control for confounders X to achieve conditional independence $(Y_1, Y_0) \perp T|X$. Instead, it is sufficient to control for a balancing score which estimates $E[T|X]$. This balancing score is often the conditional probability of the treatment, $P(T|X)$, also called the propensity score, $e(x)$.

The propensity score can be viewed as a dimensionality reduction technique. Instead of conditioning on X, which can be very high dimensional, you can simply condition on the propensity score in order to block the backdoor paths that flows through X: $(Y_1, Y_0) \perp T|P(x)$

There is a formal proof for why this is. It's not complicated, but a bit beyond the scope of this book. Here, you can approach the matter in a more intuitive way. The propensity score is the conditional

probability of receiving the treatment, right? So you can think of it as some sort of function that converts X into the treatment T. The propensity score makes this middle ground between the variable X and the treatment T. If we show this in a causal graph, this is what it would look like.



If I know what $e(x)$ is, X alone gives you no further information about T. Which means that controlling for $e(x)$ works the same way as controlling for X directly.

Think of it in terms of the manager training program. Treated and non-treated are initially not comparable because the managers with more engaged direct reports are probably more likely to participate in the training. However, if I take 2 managers, one from the treated and one from the control group, but with the same probability of receiving the treatment, they are comparable. Think about it. If they have the exact same probability of receiving the treatment, the only reason one of them did it and the other didn't is by pure chance. Given the same propensity score, treatment is as good as random.

Propensity Score Estimation

In an ideal world, you would have the true propensity score $e(x)$. You might have this in the case of conditionally random experiment, where the assignment mechanism is non-deterministic, but known. However, in most cases, the mechanism that assigns the treatment is unknown and you'll need to replace the true propensity score by an estimation of it $\hat{e}\ (x)$.

Since you have a binary treatment, a good candidate for estimating $e(x)$ is using logistic regression. To fit a logistic regression with `statsmodels`, you can simply change the method `ols` to `logit`.

```
ps_model = smf.logit("""intervention ~
tenure + last_engagement_score + deparament_score
+ C(n_of_reports) + C(gender) + C(role)""", data=df).fit(disp=0)
```

Now, let's save your estimated propensity score in a data frame. It will be used a lot in the following sections, where I'll show you how to use the propensity score and what it is doing.

```
data_ps = df.assign(
    propensity_score = ps_model.predict(df),
)
```

```
data_ps[["intervention", "engagement_score", "propensity_score"]].head()
```

	intervention	engagement_score	propensity_score
0	1	0.277359	0.629402
1	1	-0.449646	0.428037
2	1	0.769703	0.629402
3	1	-0.121763	0.629402
4	1	1.526147	0.652934

PROPENSITY SCORE AND ML

Alternatively, you can use machine learning models to estimate the propensity score. But you have to be more careful. First, you must ensure that your ML model outputs a calibrated probability prediction. Second, you need to use out-of-fold predictions to avoid bias due to overfitting. You can use `sklearn`'s calibration module for the first task and the `cross_val_predict` function, from the model selection module, for the latter. However, I want to warn you that the use of ML for propensity score is often underwhelming if not detrimental. Oftentimes a simple logistic regression will do just fine.

Propensity Score and Orthogonalization

If you recall from the previous chapter, according to the FLW theorem, linear regression also does something very similar to estimating a propensity score. In the debiasing step, it estimates $E[T|X]$. So, very much like the propensity score estimation, OLS is also modeling the treatment assignment mechanism. This means you can also use propensity score $\hat{e}(X)$ inside a linear regression in order to adjust for the confounders X:

```
model = smf.ols("engagement_score ~ intervention + propensity_score",
                data=data_ps).fit()
model.params["intervention"]
```

```
0.26383730706157343
```

The estimated ATE you get with this approach is remarkably similar to the one you got earlier, fitting a linear model with the treatment and confounder X. This is not at all surprising, as both approaches are simply orthogonalizing the treatment. The only difference is that OLS uses linear regression to model T, while this propensity score estimate was obtained from a logistic regression.

Inverse Propensity Weighting

A more common way to use the propensity score is to reweight the data by the inverse probability of the treatment (Inverse Propensity Weighting or IPW). This will make the treatment look as good as

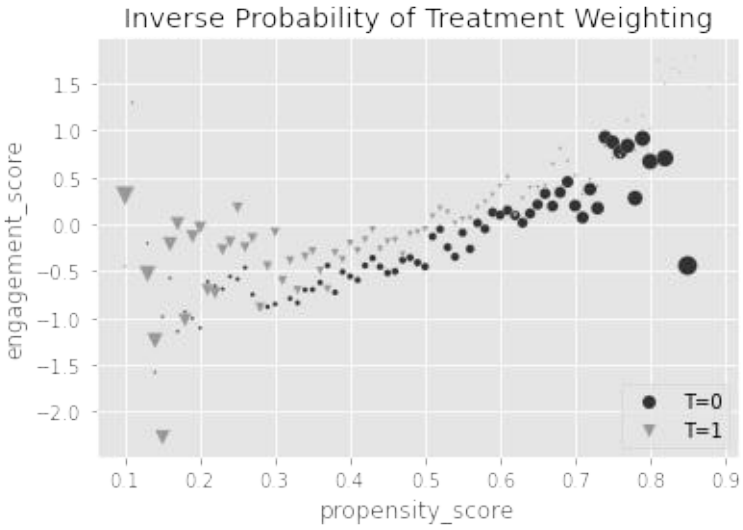
randomly assigned in the reweighted data. The idea is to reweight your sample by $1/P(T=t|X)$ to construct a pseudo-population that would resemble what would happen had everyone received the treatment t:

$$E\left[Y_t\right] = E\left[\frac{I(T=t)Y}{P(T=t|X)}\right]$$

Once again, the proof for this is not complicated, but besides the point here. So let's stick with the intuition. Suppose you want to know the expectation of Y_1, that is, of what would be the average engagement had all managers taken the training. To get that, you take all those that are treated and scale them by the inverse probability of getting the treatment. This puts a high weight on those with very low probability of treatment, but that nonetheless got it. You are essentially up-weighting rare treatment examples.

This makes sense, right? If a treated individual has a low probability of treatment, that individual looks a lot like the untreated. This must be interesting! This treated unit that look like the untreated will probably be very informative of what would happen to the untreated, had they been treated, $Y_1|T=0$. That is why you give a high weight to that unit. Same thing for the control. If a control unit looks a lot like the treated group, it is probably a good estimate for $Y_0|T=1$, so you give it more weight.

Here is what this process looks like with the management training data, with weights depicted as the size of each dot:



Notice how those that got the training, $T=1$, have a high weight when $\hat{e}\left(X\right)$, the propensity score, is low. You are giving high importance to the treated that look like the untreated. Conversely, the untreated have a high weight when $\hat{e}\left(X\right)$ is high, or when $\widehat{P}\left(T=0|X\right)$ is low. Here you are giving high importance to the untreated that look like the treated.

If you can use the propensity score to recover the average potential outcome, it also means you can use it to identify the ATE

$$ATE = E\left[\frac{I(T=1)Y}{P(T=1|X)}\right] - E\left[\frac{I(T=0)Y}{P(T=0|X)}\right]$$

Both expectations above can be estimated from data with very simple code:

```
weight_t = 1/data_ps.query("intervention==1")["propensity_score"]
weight_nt = 1/(1-data_ps.query("intervention==0")["propensity_score"])

y1 = sum(data_ps.query("intervention==1")["engagement_score"]*weight_t)/len(data_ps)
y0 = sum(data_ps.query("intervention==0")["engagement_score"]*weight_nt)/len(data_ps)

print("E[Y1]:", y1)
print("E[Y0]:", y0)
print("ATE", y1 - y0)
```

```
E[Y1]: 0.1196797239967515
E[Y0]: -0.14622629574638177
ATE 0.26590601974313327
```

Using this approach, the ATE is once again smaller than the one you got naively, not adjusting for X. Moreover, this result looks pretty similar to the one you got by using OLS, which is a good check to make sure you didn't do anything wrong. It is also worth noticing that the ATE expression from above can be simplified to the following

$$ATE = E\left[Y\frac{T - e(x)}{e(x)(1 - e(x))}\right]$$

Notice how it produces the exact same result as before

```
np.mean(data_ps["engagement_score"]
        * (data_ps["intervention"] - data_ps["propensity_score"])
        / (data_ps["propensity_score"]* (1-data_ps["propensity_score"])))
```

```
0.2659060197431334
```

The formula above is very neat because it also gives you some insight into how IPW compares to regression. With regression, you are estimating the treatment effect with

$$ATE = E\left[Y(T - E[T|X])\right]/Var\left(T\right)$$

Now, recall that the variance of a bernoulli variable with probability p is simply $p(1-p)$. This means that IPW is just estimating

$$ATE = E\left[Y(T - E[T|X])\right]/Var\left(T|X\right)$$

Variance of IPW

Unfortunately, computing the standard error for the IPW isn't as easy as with linear regression. The simplest way to place a confidence interval around your IPW estimate is by using bootstrap. To do that, you'll re-sample the data with replacement multiple times and get the IPW estimator for each sample. Then, you can get the 2.5 and 97.5 percentile of these multiple estimates to get a 95% Confidence

Interval.

To code that, let's first wrap your IPW estimation into a reusable function. Notice how I'm replacing `statsmodels` with `sklearn`. The `logit` function in `statsmodels` is slower than the logistic regression model from `sklearn`, so this change will save you some time. Also, since you probably don't want to lose the convenience of formulas you get from `statsmodels`, I'm using `patsy`'s `dmatrix` function. This function engineers a feature matrix based on an R-style formula, like the ones you've been using so far.

> **NOTE**
>
> By default, `sklearn`'s classifiers output 0 or 1 predictions following the logic $\widehat{P}(Y|X) > 0.5$. Since you want your model to output a probability, you'll have to use the `predict_proba` method. This method outputs a two column matrix, where the first column is $\widehat{P}(Y = 0|X)$ and the second column, $\widehat{P}(Y = 1|X)$. You only want the second one - which in this case is $\widehat{P}(T = 1|X)$ -, hence, the indexing `[:, 1]`.

```
from sklearn.linear_model import LogisticRegression
from patsy import dmatrix

# define function that computes the IPW estimator
def est_ate_with_ps(df, formula, T, Y):

    X = dmatrix(formula, df)
    ps_model = LogisticRegression(penalty="none", max_iter=1000).fit(X, df[T])
    ps = ps_model.predict_proba(X)[:, 1]

    # compute the ATE
    return np.mean((df[T]-ps) / (ps*(1-ps)) * df[Y])
```

Here is how you would use this function:

```
formula = """tenure + last_engagement_score + deparament_score
+ C(n_of_reports) + C(gender) + C(role)"""
T = "intervention"
Y = "engagement_score"

est_ate_with_ps(df, formula, T, Y)
```

```
0.26590302870250465
```

Now that you have the code to compute ATE inside a neat function, you can apply it inside a bootstrap procedure. To speed things up a little, I'm also going to run the resampling in parallel. All you have to do is call the data frame method `.sample(frac=1, replace=True)` to get a bootstrap sample. Then, pass this sample to the function you've created earlier. To make the bootstrap code more generic, one of its arguments is an estimator function, `est_fn`, which takes a dataframe and returns a single number as an estimate. I'm using 4 jobs, but feel free to set this to the number of cores in your computer.

Run this estimator multiple times, one in each bootstrap sample, and you'll end up with an array of estimates. Finally, to get the 95% CI, just take the 2.5 and 97.5 percentiles of that array.

```
from joblib import Parallel, delayed # for parallel processing

def bootstrap(data, est_fn, rounds=200, seed=123, pcts=[2.5, 97.5]):
    np.random.seed(88)

    stats = Parallel(n_jobs=4)(delayed(est_fn)(data.sample(frac=1, replace=True))
                               for _ in range(rounds))

    return np.percentile(stats, pcts)
```

Notice that I tend to lean towards functional programming in my code, which might not be familiar for some. For this reason, I'll add notes explaining some of the functional patterns that I'm using, starting with the `partial` function.

`partial` takes in a function and some of its arguments and returns another function just like the input one, but with the arguments that you passed already applied.

```
def addNumber(x, number):
    return x + number

add2 = partial(addNumber, number=2)
add4 = partial(addNumber, number=4)

add2(3)
>>> 5

add2(3)
>>> 7
```

I'll use `partial` to take the `est_ate_with_ps` function and partially apply the formula, the treatment and the outcome arguments. This will give me a function that has a dataframe as its only input and that outputs the ATE estimate. I can then pass this function as the `est_fn` argument to the bootstrap function I've created earlier.

```
from toolz import partial

print(f"ATE: {est_ate_with_ps(df, formula, T, Y)}")
print(f"95% C.I.: ", bootstrap(df, partial(est_ate_with_ps, formula=formula, T=T, Y=Y)))


ATE: 0.26590302870250465
95% C.I.:  [0.24057215 0.30039199]
```

This 95% is about as wide as the one you got earlier, with linear regression. It's important to realize that the variance in the propensity score estimator will be large if you have big weights. Big weights means that some units have a big impact in the final estimate. A few units having a big impact in the final estimate is precisely what causes the variance.

Also notice that you'll have big weights if you have few control units in the region with high propensity score or a few treated units in the region with low propensity score. This will cause you to have few units to estimate the counterfactuals $Y_0|T=1$ and $Y_1|T=0$, which might give you a very noisy result.

Stabilized Propensity Weights

Weighting the treated samples by $1/P(T=1|X)$ creates a pseudo-population the same size as the original one, but as though everyone was treated. Likewise, weighting the control by $1/P(T=0|X)$ creates a pseudo-population that behaves as though everyone had the control.

If you come from a machine learning background, you might recognize IPW as an application of importance sampling. With importance sampling, you have data from an origin distribution $q(x)$ but want to sample from a target distribution $p(x)$. To do that, you can re-weight the data from $q(x)$ by $p(x)/q(x)$. Bringing this to an IPW context, weighting the treated by $1/P(T=1|X)$ essentially means you are taking data that came from $P(T=1|X)$ - which is biased if X also causes Y - and reconstructing $P(T=1)=1$, where the treatment probability does not depend on X, since it is just 1. This also explains why the resulting re-weighted sample behaves as if everyone in the original sample was treated.

Another way to see that is to notice how the sum of the weights for both the treatment and the untreated are pretty close to the original sample size.

```
print("Original Sample Size", data_ps.shape[0])
print("Treated Pseudo-Population Sample Size", sum(weight_t))
print("Untreated Pseudo-Population Sample Size", sum(weight_nt))
```

```
Original Sample Size 10391
Treated Pseudo-Population Sample Size 10448.707665603628
Untreated Pseudo-Population Sample Size 10340.117306133456
```

This is fine, as long as you don't have weights that are too large. But if a treatment is very unlikely, $P(T|X)$ can be tiny, which might cause you some computational issues. A simple solution is to stabilize the weights using the marginal probability of treatment, $P(T=t)$:

$$w = \frac{P(T=t)}{P(T=t|X)}$$

With these weights, a treatment with low probability won't have massive weights because the small denominator will be balanced by the also small numerator. This won't change the results you got earlier, but it is more computationally stable.

Moreover, the stabilized weights reconstruct a pseudo-population where the effective size (sum of the weights) of both treatment and control matches that of the original data. Making again a parallel with importance sampling, with stabilized weights, you are coming from a distribution where the treatment depends on X, $P(T=t|X)$, but reconstructing the marginal $P(T=t)$.

```
p_of_t = data_ps["intervention"].mean()

weight_t_stable = p_of_t/data_ps.query("intervention==1")["propensity_score"]
weight_nt_stable = (1-p_of_t)/(1-data_ps.query("intervention==0")["propensity_score"])

print("Treat size:", len(data_ps.query("intervention==1")))
print("W treat", sum(weight_t_stable))
```

```
print("Control size:", len(data_ps.query("intervention==0")))
print("W treat", sum(weight_nt_stable))
```

```
Treat size: 5611
W treat 5642.16136191911
Control size: 4780
W treat 4756.593275268779
```

Again, this stabilization keeps the same balancing properties of the original propensity score. You can verify that it yields the exact same ATE estimate as you had before.

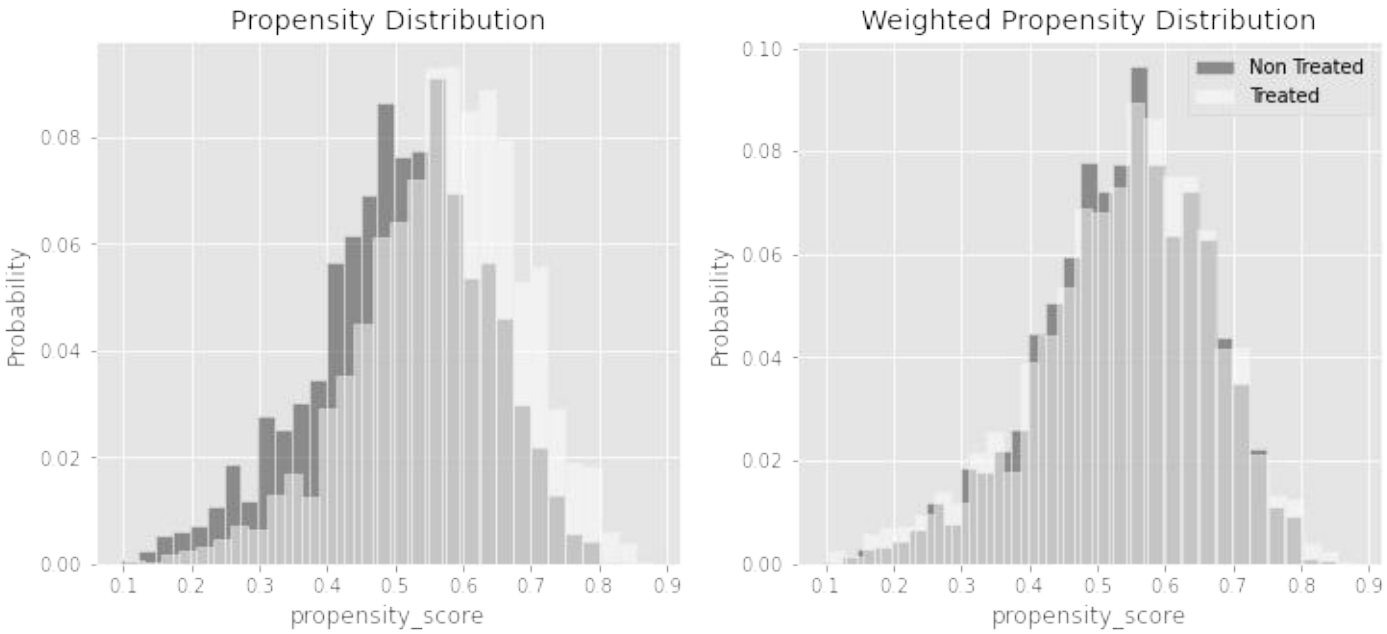
```
nt = len(data_ps.query("intervention==1"))
nc = len(data_ps.query("intervention==0"))

y1 = sum(data_ps.query("intervention==1")["engagement_score"]*weight_t_stable)/nt
y0 = sum(data_ps.query("intervention==0")["engagement_score"]*weight_nt_stable)/nc

print("ATE: ", y1 - y0)
```

```
ATE:  0.26590601974313244
```

Pseudo-Populations

I've mentioned pseudo-populations already, but understanding them better will help you understand how IPW removes bias. Lets first think about what bias means from the perspective of $P(T|X)$. If the treatment was randomly assigned with probability, say, 10%, you know that the treatment does not depend on X, or that $P(T|X) = P(T) = 10\%$. So, if the treatment is independent from X, you would not have confounding bias flowing through X and no adjustment would be needed. If you do have this sort of bias, then some units have a higher chance of getting the treatment than other units. For example, it could be that very passionate managers, who already have a very engaged team, are more likely to take the training (have higher $e(T)$) than the managers whose team is not so engaged.

If you plot the distribution of $\hat{e}(x)$ by treatment status, since managers don't have the same chance of taking the training (treatment is not random), treated individuals will have higher $\hat{e}(x)$. You can see that in the first plot below, where the treated distribution for $\hat{e}(x)$ is a bit skewed to the right of the untreated distribution.



Contrast this with the second figure. Here, in the low $\hat{e}(X)$ region, treated are up-weighted and control, down-weighted. Similarly, when $\hat{e}(X)$ is high, treated units are down-weighted and control, up-weighted. These movements make the two distributions overlap. The fact that they do means that, on the weighted data, treated and control have the same chance of getting both the treatment or the control. In other words, treatment assignment looks as good as random (assuming no unobserved confounders, of course).

This also shed some light on what IPW is doing. By taking the treated's outcome, $Y|T=1$, and up-weighting those where $\hat{e}(X)$ is low and down-weighting those where $\hat{e}(X)$, you are trying to figure out what $Y_1|T=0$ would look like. A similar argument can be made to show how you are also trying to learn $Y_0|T=1$ by reweighting the control sample by $1/(1-P(T=1))$.

Selection Bias

The example used here is meant to show how propensity score weighting can be used to adjust for common causes, making the treatment similar to the control and vice versa. That is, you saw how to use propensity score weighting as a way to account and control for confounding bias. However, IPW can also be used to adjust for selection issues. In fact, the IPW estimator was initially used in this context, as presented by Horvitz and Thompson in 1952. For this reason, you might see the IPW estimator as the Horvitz-Thompson estimator.

To give an example, suppose you want to know the satisfaction of your customers with your App. So you send out a survey asking them to rate your product on a 1 to 5 scale. Naturally, some customers don't respond. But the issue with this is that it can bias your analysis. If the non-respondents are mostly unsatisfied customers, the result you'll get back from the survey will be an artificially inflated rate.

To adjust for that you can estimate the probability of responding R given customer's covariates (like age, income, App usage, etc…), $P(R=1|X)$. Then, you can reweight those that responded by $1/\widehat{P}(R=1)$. This will upweight the respondents that look like the non-respondents (have low $\widehat{P}(R=1)$). With this, an individual that answered the survey will not only account for himself, but for other individuals like him, creating a pseudo-population that should behave like the original one, but

as if everyone responded to the survey.

Sometimes (but hopefully not many), you'll have to face both confounding and selection bias together. In this case, you can use the product of the weights for both selection and confounding. Since this product can be quite small, I recommend stabilizing the confounding bias weights by marginal probability $P(T = t)$

$$W = \frac{\widehat{P}\left(T = t\right)}{\widehat{P}\left(R = 1|X\right)\widehat{P}\left(T = t|X\right)}$$

Bias-Variance Trade-Off

As a naive data scientist that I was, when I learned about propensity scores I thought "Oh boy! This is huge! I can transform a causal inference problem into a prediction problem. If I can just predict $e(x)$, I'm golden!". Unfortunately, it is not that simple. In my defense, that is an easy mistake to make. At first glance, it does seem that the better your estimate of the treatment assignment mechanism, the better your causal estimates will be. But that is simply not the case.

Recall from the Linear Regression when you learned about noise inducing controls? The same logic applies here. If you have a covariate X_k which is a very good predictor of T, this variable will give you a very accurate model of $e(x)$. But if that same variable does not cause Y, it is not a confounder and it will only increase the variance of your IPW estimate. To see this, think about what would happen if you have a very good model of T. This model would output a very high $\hat{e}\left(x\right)$ for the all treated units (as it correctly predicts that they are treated) and a very low $\hat{e}\left(x\right)$ for all the control units (as it correctly predicts that they are untreated). This would leave you no treated units with low $\hat{e}\left(x\right)$ to estimate $Y_1|T = 0$ and no control units with high $\hat{e}\left(x\right)$ to estimate $Y_0|T = 1$.

In contrast, think about what would happen if the treatment was randomized. In this case, the predictive power of you $\hat{e}\left(x\right)$ should be zero! Still, even with no predictive power, this is the best situation you'll get in terms of estimating the treatment effect.

As you can see, there is also a bias-variance trade-off when it comes to IPW. In general, the more precise the propensity score model, the lower the bias. However, a very precise model for $e(x)$ will generate a very imprecise effect estimate. This means you have to make your model precise enough to control for the bias, but not too much, or you'll run into variance issues.

TRIMMING

One way to lower the variance of the IPW estimator is to trim the propensity score to be always above a certain number - say, 1% - to avoid weights that are too big - say, above 100. Equivalently, you can directly clip the weights to never be too large. The IPW with clipped weights is no longer unbiased, but it might have lower mean square error if the variance reduction is expressive.

Positivity

The bias variance trade-off can also be viewed in the light of two causal inference assumptions:

Conditional Independence (unconfoundedness) and Positivity. The more precise you make your model for $e(x)$, say, by adding more variables to it, the more you go in the direction of making the CIA hold. However, you also make positivity less plausible for the same reasons you already saw: you'll concentrate the treatment in a low $\hat{e}(x)$ region, far away from the controls and vice versa.

The IPW reconstruction is only possible if you have samples to reweight. If there are no treated samples in the region with low propensity score (high chance of being the control), there is no amount of reweight you can do to reconstruct Y_1 in that region. This is what positivity violations look like in terms of IPW. Also, even if positivity is not entirely violated, but some units have very small or large propensity scores, IPW will suffer from high variance.

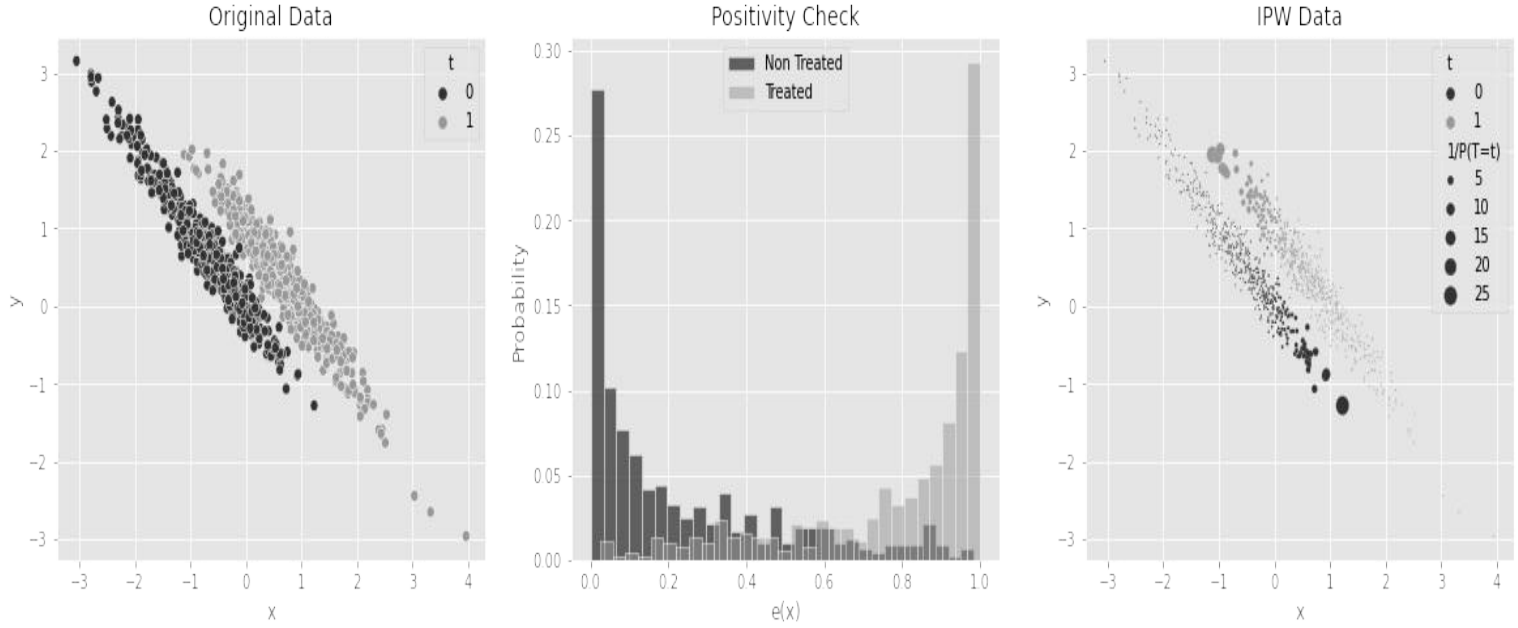
To see this on a more intuitive level, consider the following simulated data. Here, the true ATE is 1. However, x confounds the relationship between T and Y. The higher the X, the smaller the Y, but the higher the chance of receiving the treatment. Hence, a naive comparison in the average outcome between treated and control will be downward biased and can even be negative

```
np.random.seed(1)

n = 1000
x = np.random.normal(0, 1, n)
t = np.random.normal(x, 0.5, n) > 0

y0 = -x
y1 = y0 + t

y = np.random.normal((1-t)*y0 + t*y1, 0.2)

df_no_pos = pd.DataFrame(dict(x=x,t=t.astype(int),y=y))

df_no_pos.head()
```

	x	t	y
0	1.624345	1	-0.526442
1	-0.611756	0	0.659516
2	-0.528172	0	0.438549
3	-1.072969	0	0.950810
4	0.865408	1	-0.271397

If you estimate the propensity score in this data, some units (with high x) have a propensity score very close to 1, meaning it is almost impossible for them to get the control. In a similar manner, some units have almost zero chance of getting the treatment (those with low x). This can be seen in the second figure below. Notice the lack of overlap between the treated and untreated propensity score distribution. This is very troublesome, as it means that a huge chunk of the control distribution is condensed where

$e(x)$ is close to zero, but you have no treated unit to reconstruct that region. As a consequence, you end up lacking a good estimate for $Y_1|T=0$ for a good portion of the data.



Additionally, as you can see in the third plot, the few control units that are very to the right (have high $e(x)$) have massive weights. A similar thing can be said for the treated units with very small $e(x)$. As you know by now, these huge weights will generally increase the variance of the IPW estimator.

Combine these two problems - high variance and positivity violation - and you'll see how the IPW estimator fails to recover the ATE of 1 in this data:

```
est_fn = partial(est_ate_with_ps, formula="x", T="t", Y="y")
print("ATE:",  est_fn(df_no_pos))
print(f"95% C.I.: ", bootstrap(df_no_pos, est_fn))
```

```
ATE: 0.6478011810615735
95% C.I.:  [0.4795679  0.83514887]
```

Also notice how this is not simply a problem of high variance. Sure, the 95% CI of this estimator is large, but it is more than that. The upper end of the CI still seems pretty off the correct ATE of 1.

Lack of positivity is a problem not only for the IPW estimator. However, IPW can be more transparent about positivity issues. For instance, if you plot the distribution of the propensity score (the plot in the middle in the figure above) for the treatment variants you can visually check if you have decent levels of positivity. This analysis is very similar to the one from Chapter 4, where you plot the treatment distribution by $\hat{T}$.

In fact, let's contrast the IPW estimator with linear regression. You know that regression will not be very transparent about positivity violations. Instead, it will extrapolate to the regions where you have no data whatsoever. In some very lucky situations, this might even work. For instance, in this very simple simulated data, regression manages to recover the ATE of 1, but only because it correctly extrapolates both Y_0 and Y_1 to the treated and control region where there is no actual data.

```
smf.ols("t ~ x + t", data=df_no_pos).fit().params["t"]
```

1.0

In a sense, regression can replace the positivity assumption for a parametric assumption on $E[Y|T,X]$, which is essentially an assumption about the smoothness in the potential outcomes. If the linear model has a good fit to the conditional expectation, it will manage to recover the ATE even in regions where positivity doesn't hold. In contrast, IPW makes no assumptions on the shape of the potential outcome. As a result, it fails when extrapolation is needed.

Doubly Robust Estimation

You've just learned how to use propensity score weighting to estimate the treatment effect. Along with regression, this already gives two - and the most important - methods to debias non-experimental data. But which one should you use and when? Regression or IPW? When in doubt, just use both! Doubly Robust (DR) Estimation is a way of combining propensity score and linear regression in a way you don't have to rely on either of them. Let me first show you a very popular DR estimator and then I'll tell you why it is awesome.

Quite generally, a doubly robust estimator for the counterfactual Y_t can be written as follows:

$$\widehat{\mu}_t^{DR}\left(\widehat{m},\hat{e}\right)=\frac{1}{N}\sum\widehat{m}\left(X\right)+\frac{1}{N}\sum\left[\frac{T}{\hat{e}\left(x\right)}\left(Y-\widehat{m}\left(X\right)\right)\right]$$

where $\widehat{m}\left(X\right)$ is a model for $E[Y|X]$ (linear regression, for example) and $\hat{e}\left(X\right)$ is a propensity score model for $P(T|X)$. Now, the reason why this is amazing - and why it is called doubly robust - is that it only requires one of the models, $\widehat{m}\left(X\right)$ or $\hat{e}\left(X\right)$, to be correctly specified.

For example, suppose that the propensity score model was wrong, but the outcome model $\widehat{m}\left(X\right)$ was correct. In this case, the second term would converge to zero, since $E[Y=\widehat{m}\left(X\right)]=0$. You would be left with the first term, which is just the outcome model, which is correct.

Now, what would happen if the outcome model was wrong, but the propensity score model $\hat{e}\left(X\right)$ was correct? To see it, let's do some algebraic manipulation to the formula above and rewrite it as follows:

$$\widehat{\mu}_t^{DR}\left(\widehat{m},\hat{e}\right)=\frac{1}{N}\sum\frac{TY}{\hat{e}\left(X\right)}+\frac{1}{N}\sum\left[\frac{T-\hat{e}\left(X\right)}{\hat{e}\left(X\right)}\widehat{m}\left(X\right)\right]$$

I hope this makes it more clear. If the propensity model is correct, $T-\hat{e}\left(X\right)$ would converge to zero. That would leave you only the first term, which is the IPW estimator. And since the propensity model is correct, this estimator would be too. That's beauty of this estimator: if one of the model is right, it converges to it.

The doubly robust estimator outlined above would estimate the average counterfactual outcome Y_t. If you want to estimate the average treatment effect, all you have to do is put two of those estimators together, one for $E[Y_0]$ and one for $E[Y_1]$, and take the difference.

$$ATE = \widehat{\mu}_1^{DR}\left(\widehat{m}, \hat{e}\right) - \widehat{\mu}_0^{DR}\left(\widehat{m}, \hat{e}\right)$$

Having understood the theory behind DR, it's time to code it up. The models $\hat{e}$ and $\widehat{m}$ don't have to be a logistic and linear regression, respectively, but I think those are very good candidates for a starter. Once again, I'll begin by using the R-style formula from `patsy`'s `dmatrix` to engineer my covariate matrix X. Next, I'm using logistic regression to fit the propensity model and get $\hat{e}\left(X\right)$. Then comes the output model part. I'm fitting one linear regression per treatment variant, giving me two of them - one for the treated and one for the control. Each model is fitted on the subset of the data of its treatment variant, but makes predictions for the entire data. For example, the control model fits only in the data where $T == 0$, but it predicts everything. This prediction is an outcome model estimate for Y_0.

Finally, I'm combining the two models to form the DR estimator for both $E[Y_0]$ and $E[Y_1]$. This is simply the conversion of the formula you just saw into code.

```
from sklearn.linear_model import LinearRegression

def doubly_robust(df, formula, T, Y):
    X = dmatrix(formula, df)

    ps_model = LogisticRegression(penalty="none", max_iter=1000).fit(X, df[T])
    ps = ps_model.predict_proba(X)[:, 1]

    m0 = LinearRegression().fit(X[df[T]==0, :], df.query(f"{T}==0")[Y]).predict(X)
    m1 = LinearRegression().fit(X[df[T]==1, :], df.query(f"{T}==1")[Y]).predict(X)

    return (
        np.mean(df[T]*(df[Y] - m1)/ps + m1) -
        np.mean((1-df[T])*(df[Y] - m0)/(1-ps) + m0)
    )
```

Let's see how this bad boy performs in our manager training data. You can also pass it to your bootstrap function to construct a confidence interval for the DR ATE estimate

```
formula = """tenure + last_engagement_score + deparament_score
+ C(n_of_reports) + C(gender) + C(role)"""
T = "intervention"
Y = "engagement_score"

print("DR ATE:", doubly_robust(df, formula, T, Y))
print("95% CI", bootstrap(df, partial(doubly_robust, formula=formula, T=T, Y=Y)))


DR ATE: 0.2719759652386582
95% CI [0.24465668 0.30696022]
```

As you can see, the result is pretty in line with both the IPW and the regression estimator you saw earlier. This is good news, as it means the DR estimator is not doing anything crazy. But honestly, it is kind of boring and it doesn't exactly show the power of DR. So, to better understand why DR is so interesting, let's craft two new examples. They will be fairly simple, but very illustrative.

Treatment is Easy to Model

The first example is one where the treatment assignment is fairly easy to model, but the outcome model is a bit more complicated. Specifically, the treatment follows a Bernoulli distribution with probability given by the following propensity score

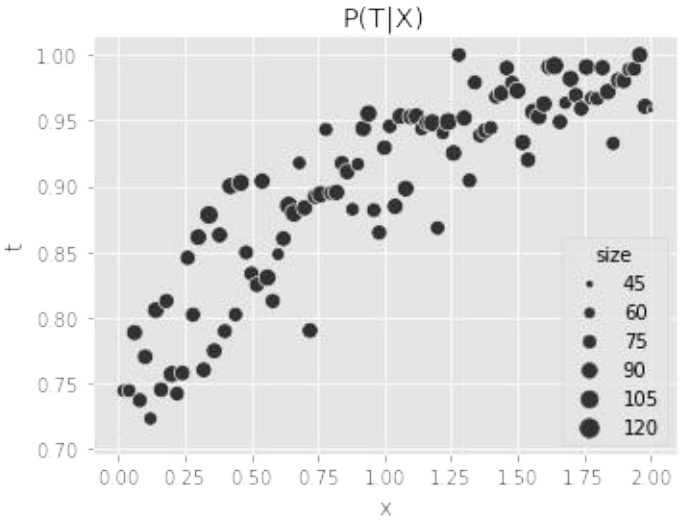
$$e\left(x\right) = \frac{1}{1 + e^{-(1+1.5x)}}$$

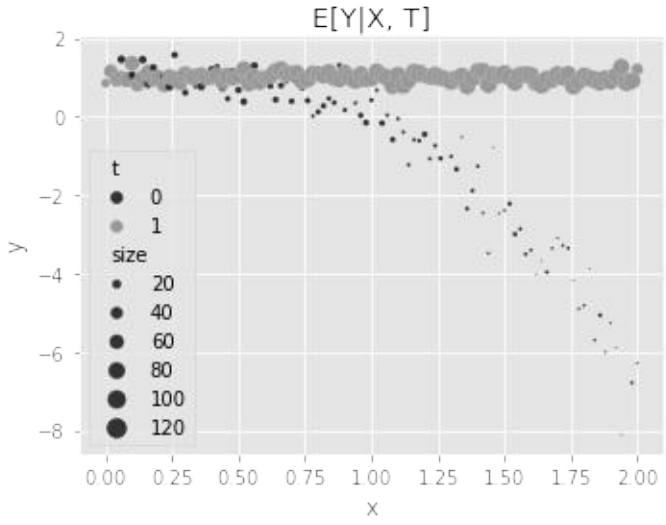
In case you didn't recognize, this is exactly the kind of form the logistic regression assumes, so it should be pretty easy to estimate it. Moreover, since $P(T|X)$ is easy to model, the IPW score should have no problem finding the true ATE here, which is close to 2. In contrast, since the outcome Y is a bit trickier, a regression model might run into some trouble.

```
np.random.seed(123)

n = 10000
x = np.random.beta(1,1, n).round(2)*2
e = 1/(1+np.exp(-(1+1.5*x)))
t = np.random.binomial(1, e)

y1 = 1
y0 = 1 - 1*x**3
y = t*(y1) + (1-t)*y0 + np.random.normal(0, 1, n)

df_easy_t = pd.DataFrame(dict(y=y, x=x, t=t))

print("True ATE:", np.mean(y1-y0))
```

```
True ATE: 2.0056243152
```

The following two plots show what this data looks like. It is interesting to notice the effect heterogeneity in the data, which is easy to see in the second plot. Notice how the effect is 0 for low values of x and it increases non-linearly as x increases. This sort of heterogeneity is oftentimes hard for regression to get it right.



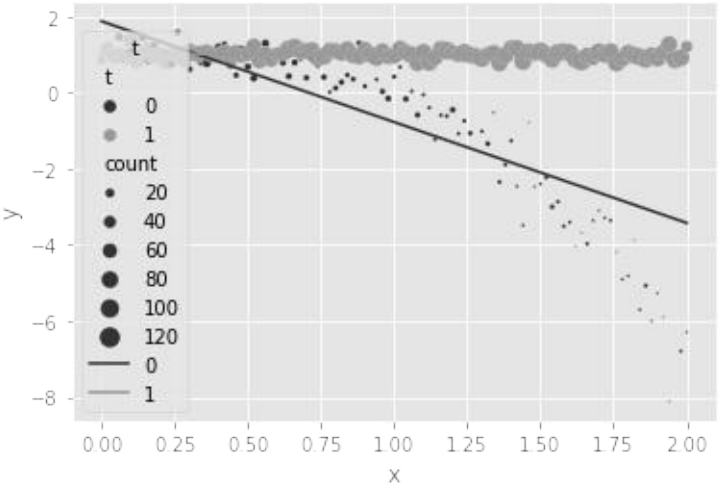


Now, let's see how regression does in this data. Here I'm once again fitting $\widehat{m}_1$ and $\widehat{m}_0$ separately and estimating the ATE as the average of the different predictions in the entire data, $N^-1 \sum (\widehat{m}_1\left(x\right) - \widehat{m}_0\left(X\right))$.

```
m0 = smf.ols("y~x", data=df_easy_t.query("t==0")).fit()
m1 = smf.ols("y~x", data=df_easy_t.query("t==1")).fit()
regr_ate = (m1.predict(df_easy_t) - m0.predict(df_easy_t)).mean()

print("Regression ATE:", regr_ate)
```

```
Regression ATE: 1.786678396833022
```

As expected, the regression model fails to recover the true ATE of 2. If you plot the predicted values against the original data you can see why. Regression fails to capture the curvature in the control group.



To be clear, this doesn't mean it is not possible to correctly estimate the ATE with regression. If you knew about the true curvature of the data, you could pretty much model it correctly:

```
m = smf.ols("y~t*(x + np.power(x, 3))", data=df_easy_t).fit()
regr_ate = (m.predict(df_easy_t.assign(t=1))
            - m.predict(df_easy_t.assign(t=0))).mean()

print("Regression ATE:", regr_ate)
```

```
Regression ATE: 1.9970999747190072
```

But, of course, in reality, you don't really know how the data was generated. So, more likely than not, regression would have failed you here. In contrast, let's see how IPW does. Again, since it is pretty easy to model the treatment assignment, you should expect IPW to perform quite well on this data.

```
est_fn = partial(est_ate_with_ps, formula="x", T="t", Y="y")
print("Propensity Score ATE:", est_fn(df_easy_t))
print("95% CI", bootstrap(df_easy_t, est_fn))
```

```
Propensity Score ATE: 2.002350388474011
95% CI [1.77820012 2.24147974]
```

Notice how IPW pretty much nails the correct ATE. Finally, the moment you've been waiting for, let's see how the DR estimate does. Remember, DR requires one of the models - $P(T|X)$ or $E[Y|X]$ - to be correct, but not necessarily both. In this data, the model for $P(T|X)$ will be correct, but the model for $E[Y|X]$ will be wrong.

```
est_fn = partial(doubly_robust, formula="x", T="t", Y="y")
print("DR ATE:", est_fn(df_easy_t))
print("95% CI", bootstrap(df_easy_t, est_fn))

DR ATE: 2.001617934263116
95% CI [1.85295505 2.15761209]
```

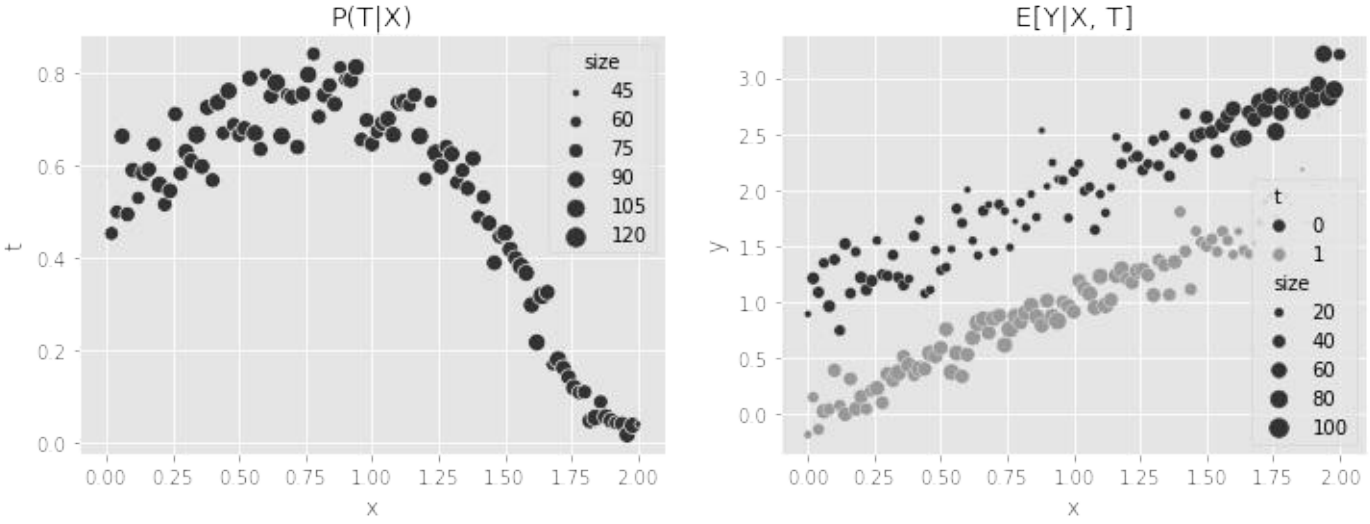
As expected, the DR performs quite well here, also recovering the true ATE. But there is more. Notice how the 95% CI is smaller than that of pure IPW estimate, meaning the DR estimator is more precise here. This simple example shows how DR can perform well when $P(T|X)$ is easy to model even if it gets $E[Y|X]$ wrong. But what about the other way around?

Outcome is Easy to Model

In this next simple yet illustrative example, the complexity is in $P(T|X)$ rather than $E[Y|X]$. Notice the non linearity in $P(T|X)$, while the outcome function is simply linear. Here, the true ATE is -1.

```
np.random.seed(123)

n = 10000
x = np.random.beta(1,1, n).round(2)*2
e = 1/(1+np.exp(-(2*x - x**3)))
t = np.random.binomial(1, e)

y1 = x
y0 = y1 + 1
y = t*(y1) + (1-t)*y0 + np.random.normal(0, 1, n)

df_easy_y = pd.DataFrame(dict(y=y, x=x, t=t))

print("True ATE:", np.mean(y1-y0))

True ATE: -1.0
```

The same kind of plot from before can be used to show the complex functional form for $P(T|X)$ and the simplicity of $E[Y|X]$.



With this data, since the propensity score is relatively complex to model, IPW does not manage to

recover the true ATE of -1

```
est_fn = partial(est_ate_with_ps, formula="x", T="t", Y="y")
print("Propensity Score ATE:", est_fn(df_easy_y))
print("95% CI", bootstrap(df_easy_y, est_fn))
```

```
Propensity Score ATE: -1.1042900278680896
95% CI [-1.1456649  -1.05376713]
```

But regression manages to get it precisely right.

```
m0 = smf.ols("y~x", data=df_easy_y.query("t==0")).fit()
m1 = smf.ols("y~x", data=df_easy_y.query("t==1")).fit()
regr_ate = (m1.predict(df_easy_y) - m0.predict(df_easy_y)).mean()

print("Regression ATE:", regr_ate)
```

```
Regression ATE: -1.0008783612504342
```

Once again, because DR only needs one of the models to be correctly specified, it also manages to recover the true ATE here

```
est_fn = partial(doubly_robust, formula="x", T="t", Y="y")
print("DR ATE:", est_fn(df_easy_y))
print("95% CI", bootstrap(df_easy_y, est_fn))
```

```
DR ATE: -1.0028459347805823
95% CI [-1.04879441 -0.95426987]
```

I hope these two examples made it more clear why Doubly Robust estimation can be very interesting. The bottom line is that it gives you two shots at being correct. In some cases, it's hard to model $P(T|X)$, but easy to model $E[Y|X]$. In others, the reverse might be true. Regardless, as long as you can model one of them correctly, you can combine a model for $P(T|X)$ and a model for $E[Y|X]$ in a way that only one of them needs to be correct. This is the true power of Doubly Robust estimator.

SEE ALSO

The DR estimator covered here is only one of the many out there. Just to give some examples, you could take the DR estimator covered in this chapter but fit the regression model with weights set to $\hat{e}(x)$. Or, you could add $\hat{e}(x)$ to the regression model. Interestingly, linear regression alone is a DR estimator that models the treatment as $e(x) = \beta X$. It is not a very good DR estimator, since βX is not bounded between 0 and 1, as a probability model should be, but it is nonetheless a DR estimator. To learn more about other DR estimators, check out the excellent discussion in *Comment: Performance of Double-Robust Estimators When "Inverse Probability" Weights Are Highly Variable*, 2008, by Robins et al.

Generalized Propensity Score for Continuous Treatment

Up until this point in the chapter, you only saw how to use propensity scores for discrete treatment. There is a pretty good reason for that. Continuous treatments are way more complicated to deal with. So

much so that I would say that causal inference as a science doesn't have a very good answer on how to deal with them.

In Chapter 4, you managed to get away with continuous treatment by assuming a functional form for the treatment response. Something like $y = a + bt$ (linear form) or $y = a + b\sqrt{t}$ (square root form), which you could then estimate with OLS. But when it comes to propensity weighting, there is no such thing as a parametric response function. The potential outcomes are estimated non-parametrically, by reweighting and taking averages. And, when T is continuous, there are infinitely many potential outcomes Y_t. Moreover, the probability of a continuous variable is always zero, so estimating $P(T = t|X)$ won't work here.

The simplest way out of these issues is to discretize the continuous treatment into a coarser version that can then be treated as discrete. But there is another way out, which is to use the Generalized Propensity Score. If you make some changes to the traditional propensity score, you'll be able to accommodate any type of treatment. To see how this would work, consider the following example.

A bank wants to know how a loan's interest rates affect the duration (in months) that the customer chooses to pay back that loan. Intuitively speaking, the effect of interest on the duration should be negative, since people like to pay back high rate loans as fast as possible to avoid paying too much on interest.

To answer this question, the bank could randomize the interest rate, but this would be costly, both in money and in time. Instead, it wants to use the data it already has. The bank knows that interest rates were assigned by two machine learning models: `ml_1` and `ml_2`. Additionally, since the bank's data scientists were very smart, they added a random Gaussian noise to the interest rate decision making process. This ensures that the policy is non-deterministic and that the positivity assumption is not violated. The observational (non-randomized) interest data, along with information on the confounders `ml_1` and `ml_2` and the outcome `duration` is stored in the `df_cont_t` data frame.

```
df_cont_t.head()
```

	ml_1	ml_2	interest	duration
0	0.392938	0.326285	7.1	12.0
1	-0.427721	0.679573	5.6	17.0
2	-0.546297	0.647309	11.1	12.0
3	0.102630	-0.264776	7.2	18.0
4	0.438938	-0.648818	9.5	19.0

Your task is to unbias the relationship between interest rate and duration, adjusting for `ml_1` and `ml_2`. Notice that, if you estimate the treatment effect naively, not adjusting for anything, you'll find a positive treatment effect. As discussed already, this makes no business sense, so this result is probably biased.

```
m_naive = smf.ols("duration ~ interest", data=df_cont_t).fit()
m_naive.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025
Intercept	14.5033	0.226	64.283	0.000	14.061
interest	0.3393	0.029	11.697	0.000	0.282

To adjust for `ml_1` and `ml_2`, you could just include them in your model, but let's see how to manage the same thing with reweighting. The first thing you have to deal with is the fact that continuous variables have $P(T = t) = 0$ everywhere. That's because the probability is the area under the density and the area of a single point is always zero. A way around this is to use the conditional density $f(T|X)$, instead of the conditional probability $P(T = t|X)$. But this poses another problem, which is to specify the distribution of the treatment.

Here, for simplicity's sake, let's assume it is drawn from a normal distribution $T \sim N(\mu_i, \sigma^2)$. This is a fairly reasonable simplification, especially since the normal distribution can be used to approximate other distributions. Moreover, let's assume constant variance σ^2, instead of one that changes for each individual.

Recall that the density of the normal distribution is given by

$$f(t_i) = \frac{exp\left(-\frac{1}{t_i}\left(\frac{t_i - \mu_i}{\sigma_i}\right)^2\right)}{\sigma_i\sqrt{2\pi}}$$

Now, you need to estimate the parameters of this conditional gaussian, that is, the mean and standard deviation. The simplest way to do that is using OLS to fit the treatment variable.

```
model_t = smf.ols("interest~ml_1+ml_2", data=df_cont_t).fit()
```

Then, the fitted values will be used as μ_i and the standard deviation of the residual will be σ. With this, you have an estimate for the conditional density. Next, you'll need to evaluate that conditional density at the given treatment, which is why I'm passing T to the `x` argument in the density function in the following code

```
def conditional_density(x, mean, std):
    denom = std*np.sqrt(2*np.pi)
    num = np.exp(-((1/2)*((x-mean)/std)**2))
    return (num/denom).ravel()


gps = conditional_density(df_cont_t["interest"],
                          model_t.fittedvalues,
                          np.std(model_t.resid))
```

```
gps
```

```
array([0.1989118 , 0.14524168, 0.03338421, ..., 0.07339096, 0.19365006,
       0.15732008])
```

Alternatively, you can (and probably should) import the normal distribution from `scipy`:

```
from scipy.stats import norm

gps = norm(loc=model_t.fittedvalues,
           scale=np.std(model_t.resid)).pdf(df_cont_t["interest"])
gps
```

```
array([0.1989118 , 0.14524168, 0.03338421, ..., 0.07339096, 0.19365006,
       0.15732008])
```

NOTE

If the treatment follows another distribution other than the normal, you can use generalized linear models (`glm`) to fit it. For example, if T was assigned according to a Poisson distribution, you could build the GPS weights with something like the following code

```
import statsmodels.api as sm
from scipy.stats import poisson

mt = smf.glm("t~x1+x2", data=df, family=sm.families.Poisson()).fit()

gps = poisson(mu=m_pois.fittedvalues).pmf(df["t"])

w = 1/gps
```

Using the inverse of the GPS as weights in a regression model can adjust for the bias. You can see that now you'll find a negative effect of interest on duration, which makes more business sense.

```
final_model = smf.wls("duration ~ interest", data=df_cont_t, weights=1/gps).fit()
final_model.params["interest"]
```

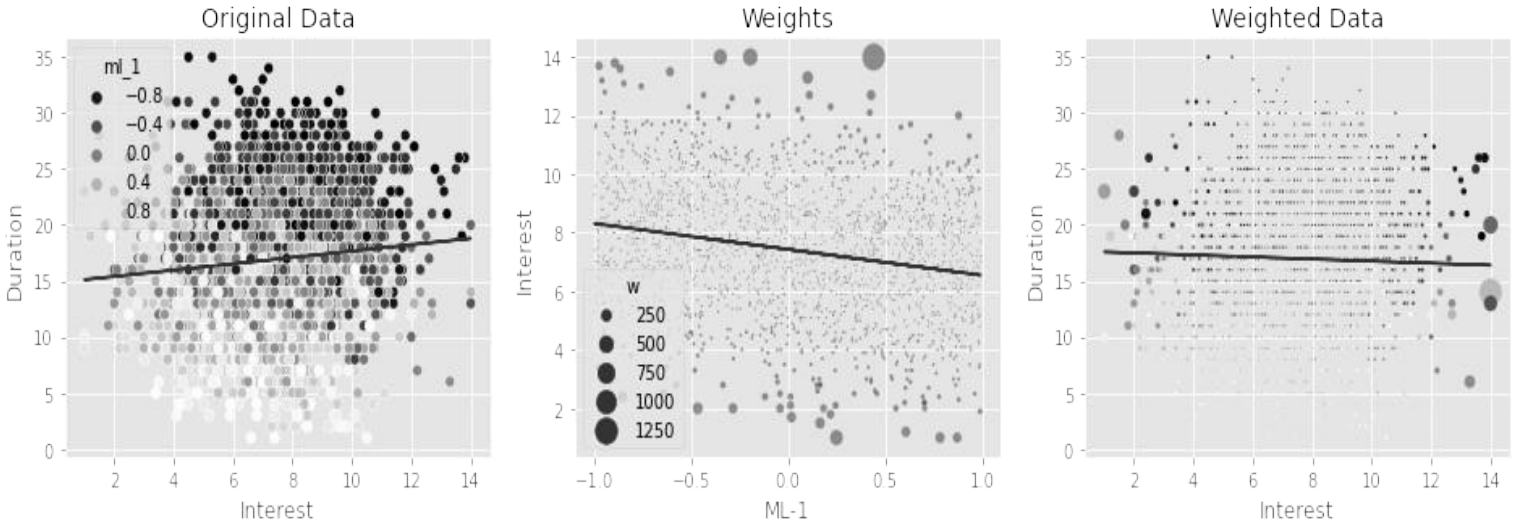
```
-0.6673977919925856
```

Still, there is one further improvement you can do. It is something that will also give you a more intuitive understanding of GPS. You see, using this score to construct weights will give more importance to points with unlikely treatments. You are essentially assigning high weights to units with high residuals in the treatment model you've just fitted. Also, due to the exponential nature of the normal density, the weights will increase exponentially with the size of the residual.

To see this, consider that instead of using both `ml_1` and `ml_2`, you fit the interest rate using only `ml_1`. This simplification is only so that I can show everything in a single plot. The next figure shows what these weights would look like. The first plot shows the original data, colored by the confounder `ml_1`. Customers with low scores on `ml_1` usually choose higher duration to pay back their loan. Also, higher

interest rates are assigned to customers with low `ml_1`. This causes upward bias in the relationship between interest rate and duration.

The second plot shows the fitted values of the treatment model and the weights constructed by the GPS obtained from that model. Notice how they are essentially larger the farther you go from the fitted line. This makes sense, as the GPS gives more weight to unlikely treatments. But look how big the weights can get. Some are bigger than 1000!



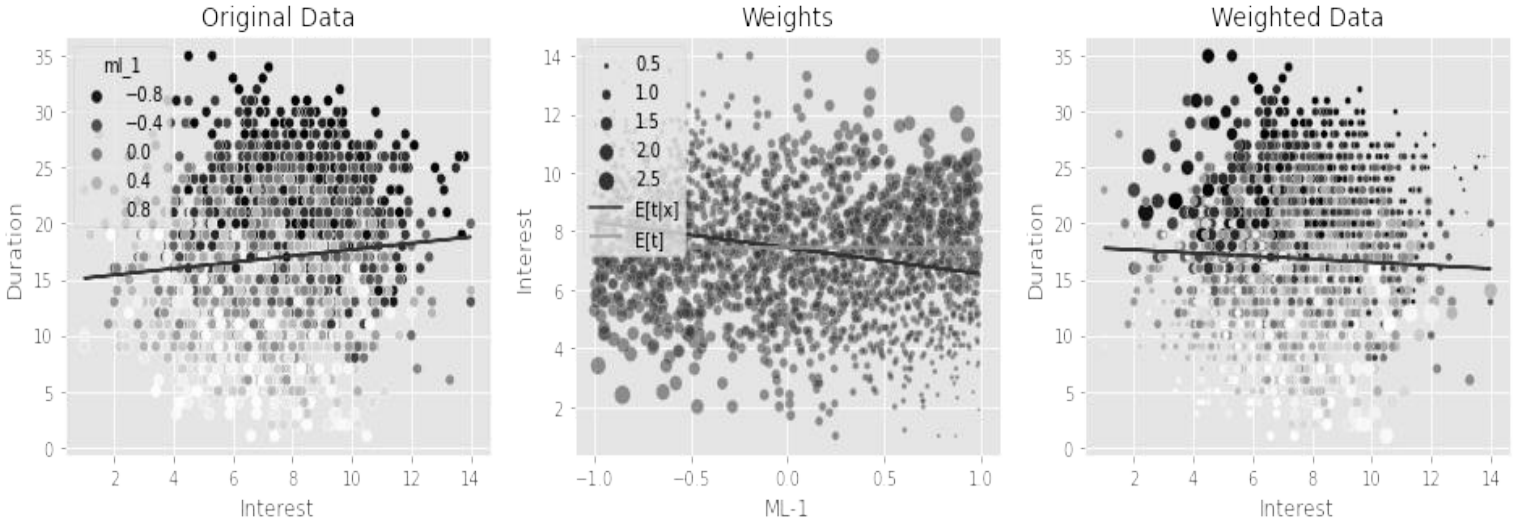
The last plot shows the same weights, but in the relationship between interest and duration. Since both low interest rates at low values of `ml_1` and high interest at high values of `ml_1` are unlikely, inverse GPS weight gives high importance to those points. This manages to reverse the positive (and biased) relationship between interest and duration. But this estimator will have insane variance, as it is practically just using a few data points - those with very high weights. Moreover, because this data was simulated, I know for a fact that the true ATE is -0.8, but the above estimate is only -0.66.

To improve upon it, you can stabilize the weights by the marginal density $f(t)$. Unlike with discrete treatment, where weight stabilization was just nice to have, when GPS, I would say it is a must. To estimate $f(t)$, you can simply use the average treatment value. Then, evaluate the resulting density at the given treatments.

Notice how this produces weights that sum to (almost) the original sample size. Thinking about this in the light of importance sampling, these weights take you from $f(t|x)$ to $f(t)$, a density where the treatment does not depend on x.

```
stabilizer = norm(
    loc=df_cont_t["interest"].mean(),
    scale=np.std(df_cont_t["interest"] - df_cont_t["interest"].mean())
).pdf(df_cont_t["interest"])

gipw =  stabilizer/gps

print("Original Sample Size:", len(df_cont_t))
print("Effective Stable Weights Sample Size:", sum(gipw))


Original Sample Size: 10000
Effective Stable Weights Sample Size: 9988.195951748608
```

Again, to understand what is going on, suppose you fit $f(t|x)$ using only `ml_1`. Once again, inverse propensity weighting gives high importance to points that are from from the fitted values of the treatment model, as they fall in a low density region of $f(t|x)$. But now, the stabilization gives low importance to points that are far away from $f(t)$, that is, points far from the mean. The result is twofold. First, the stabilized weights are much smaller, which gives you lower variance. Second, it becomes much more clear that you are now giving more importance to points with both low values of `ml_1` and low interest rate (and vice versa). You can see this by the change in the color pattern between the first and third plots.



Also, this stabilized weights gives you an estimate which is much closer to the true ATE of -0.8.

```
final_model = smf.wls("duration ~ interest", data=df_cont_t, weights=gipw).fit()
final_model.params["interest"]
```

```
-0.7787046278134085
```

As you can see, even though weight stabilization didn't have an impact in the case where T was discrete, it is very relevant for continuous treatments. It gets you closer to the true value of the parameter you are trying to estimate and it also significantly reduces the variance. Since it is a bit repetitive, I'm omitting the code to compute the 95% CI of the above estimates, but it is pretty much what you did before: just wrap the whole thing in a function and bootstrap it. But just so you can see it for yourself, here are the 95% CI with and without stabilization.

```
95% CI, non-stable:  [-0.82803619 -0.52822077]
95% CI, stable:  [-0.83806124 -0.70619965]
```

Notice how that both contain the true value of -0.8, but the one with stabilization is much narrower.

SEE ALSO

There are other ways to estimate the treatment effect with models that predict the treatment. One idea (by Hirano and Imbens) is to include the GPS as a covariate in a regression function. Another option (by Imai and van Dyk) is fit T, segment the data based on the predictions $\hat{T}$, regress the treatment on the outcome on each segment defined by $\hat{T}$ and combine the results using a weighted average, where the weights are the size of each group.

For a more comprehensive survey of the available options, I recommend checking out Douglas Galagate PhD thesis, *Causal Inference With a Continuous Treatment and Outcome*

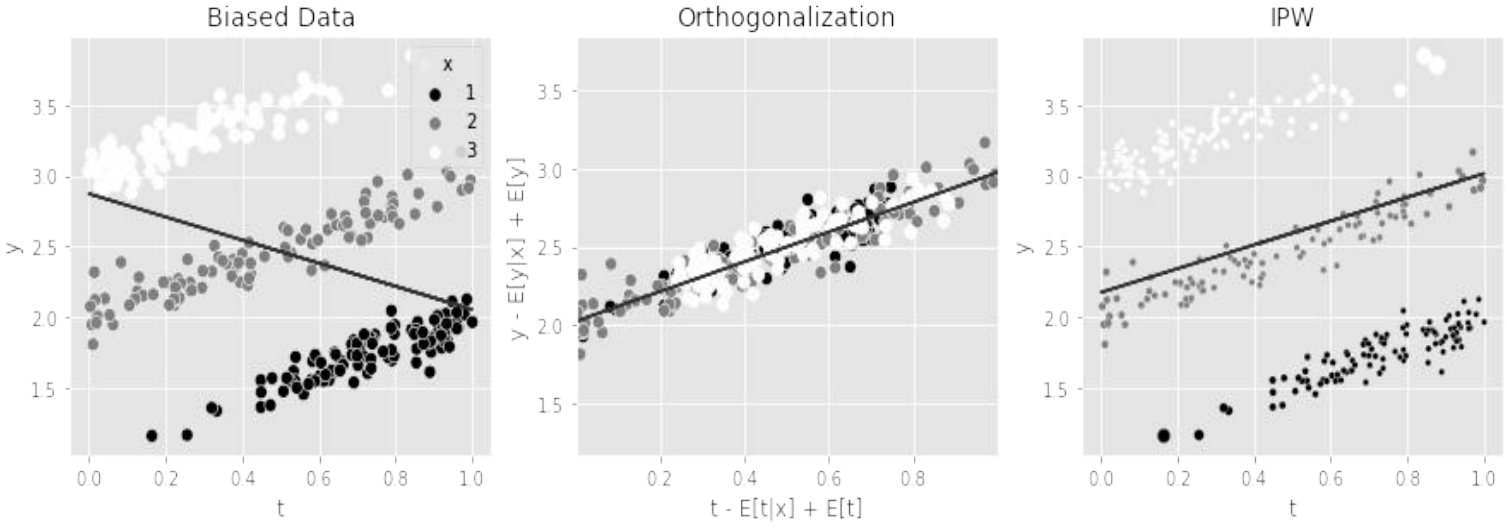
Keys Ideas

Along with regression - and orthogonalization in general - inverse propensity weighting is the second workhorse for bias reduction in your causal inference arsenal. Both techniques require you to model the treatment. This should serve as a reminder of how important it is to think about the treatment assignment mechanism in any causal inference problem. However, each technique makes use of that treatment model in a very unique way. Orthogonalization residualized the treatment, projecting it to a new space where it becomes linearly independent (orthogonal) to the covariates X that were used to model the treatment. IPW keeps the same treatment dimension, but reweights the data by the inverse of the treatment propensity:

$$w = \frac{P(T)}{P(T|X)}$$

This makes it look like the treatment was drawn from a distribution $P(T)$, which does not depend on the covariates X that were used to create the propensity model.

Here is a very simple data to showcase these two approaches. In this data, treatment is positive, but confounded by x, which is depicted in the color schema of the data points. The first plot contains the original data along a regression line of y on y. The negative slope is due to the bias that comes from x. The next two plots show how orthogonalization and IPW debias this data using very distinct ideas. Both manage to recover a positive causal effect of t on y, as shown by their respective regression lines.



If both procedures manage to debias the data, a natural question that arises is which one should you

choose. This is a bit personal, but here is my take on it. I really like IPW for when the treatment is discrete, especially if you pair it with outcome modeling in a doubly robust approach. However, when the treatment is continuous, I tend to gravitate towards regression modeling of the kind you saw in Chapter 4. With continuous treatment, you'll have very few data points around any specific treatment. As a result, a method like IPW, that doesn't pose a parametric assumption on the treatment response function becomes less appealing. For me, it is more productive to assume some smoothness in the treatment response function, allowing you to pool information from neighboring points around a particular treatment to infer its response.

Sure, I think it is very much worth understanding approaches like the Generalized Propensity Score, as it gives you further intuition into IPW in general. That is why I've included it in this chapter. Also, as the continuous treatment literature advances, I want you to be able to keep up with it, if you wish to. But, on your day to day, when T is continuous, I think you'll be better off with outcome models like linear regression.

Other Examples

These examples are used to solidify what you learned in this chapter. They use concepts and techniques you just learned and are presented in a fast paced manner. As you read through them, try to link what they tell you with what you learned in this chapter. Also, use them to try to generalize the ideas outlined here to applications in your own work.

Causal Contextual Bandits

Contextual bandits is a flavor of reinforcement learning where the goal is to learn an optional decision making policy. This can be easily framed as a causal inference problem, where you wish to learn the best treatment assignment mechanism, where best is defined in terms of the expected value of a desired outcome Y you wish to optimize. A very simple way to do this is to, at first, simply test a bunch of assignments of T and log the ones which yields better Y. Then, as time goes by, the algorithm starts to trade-off assigning treatments that already performed well and trying new treatments. Eventually, this algorithm will find the optimal treatment assignment, although it might take a long time and many sub-optimal decisions.

Taking a more causal approach to contextual bandits can yield significant improvements. If the decision making process is probabilistic, you can store the probability of assigning each treatment, which is exactly the propensity score $e(x)$. Then, you can use this propensity score to reweight the past data, where the treatment has already been selected and the outcome is already observed. This reweighted data should be unconfounded and hence much easier to learn what is the optimal treatment.

About the Author

Matheus Facure is an Economist and Senior Data Scientist at Nubank, the biggest FinTech company outside Asia. His has successfully applied causal inference in a wide range of business scenarios, from automated and real time interest and credit decision making, to cross sell emails and optimizing marketing budgets. He is also author of Causal Inference for the Brave and True, a popular book which aims at making causal inference mainstream in a light-hearted, yet rigorous way.