Welcome to DS4Humans

This website is both the course site for the Duke Interdisciplinary Data Science Course *Solving Problems with Data* (IDS 701), as well as the beginning of what I hope will eventually become a stand-alone textbook by Nick Eubank.

If you *aren't* a Duke IDS 701 student and wish to explore the content of the course, I'd suggest starting with the Introduction chapter here. You're also welcome to browse the class schedule of topics covered in the class in the class schedule link.

If you *are* a MIDS student, here's a little more about the course.

Solving Problems with Data (IDS 701)

Few fields have shown as much promise to address the world's problems as data science. At the same time, however, recent years have also made clear that today's global challenges will not be met by simply "throwing data science at the problem" and hoping things will work out. Even in business, where many assume that Artificial Intelligence is a sure ticket to profits, a major recent study found only > 11% of businesses that had piloted or employed Artificial Intelligence had reaped a sizeable return on their AI investments.

How, then, should a burgeoning data scientist approach this field full of such promise but also so many pitfalls? And why have so many data science endeavors failed to deliver on their promise?

The answer lies — at least in significant part — in a failure to provide students with a systematic approach to bringing the techniques learned in statistical modeling and machine learning courses to bear on real-world problems. Data science curricula usually begin with coding, statistics, and model evaluation techniques. All too often, however, that's where they stop. But while the hardest part of data science *classes* is often fitting a model well or getting a good AUC score, the hardest part of being an effective *professional* data scientist is ensuring that the models being fit and the results being interpreted actually solve the problem that motivated you (or your stakeholder) in the first place.

This class is designed to fill this gap. Through exercises, case studies, and projects, students will develop a *systematic* understanding of how to approach and manage data science projects from conception through delivery and adoption. It will provide a unified perspective on how the perspectives and tools learned in other courses complement one another, and *when* different approaches to data science are most appropriate.

In addition, this course will also provide an in-depth introduction into *causal inference* — the practice of answering causal questions. Given the interests of MIDS students, this introduction will focus heavily on experiments and A/B testing, but will also cover the use of observational data (data that did not come from an experiment that employed random assignment to treatment) for answering causal questions.

How to Succeed in this Class

In Duke course reviews, students are asked, "What would you like to say about this course to a student who is considering taking this course in the future?"

By far the most consistent thing past students say they would like to tell a student considering taking it in the future is to **do the readings and take them seriously.**

There is a tendency among data science students — especially those from a STEM background — to assume that readings that don't have a lot of math aren't "serious," and consequently don't require substantial focus. That's a mistake. This course is about the critical reasoning required to make the leap from the relatively clean math of statistics and machine learning to the messiness of real world problems. To help you learn how to do so, the readings are full of examples, ways to think about problems, and problem-solving frameworks to help you cross that wobbly bridge from the clean world of problem sets to the real, under- or mis-defined problems you will face when you enter the work force. But with this type of material, what you get out of it depends on what you put into it, and unlike with a theorem — which you either follow or you don't — thinking critically happens on many levels. So while it's easy to skim a reading and — because you weren't confused by any greek notation — assume you internalized it, succeeding in this class requires actively wrestling with the material, not just letting your eyes glide over it.

In other words, **take the readings for this course just as seriously as the exercises.** There is as much learning to be done by thinking deeply about the readings as there is to be gained from doing the exercises, a fact that is also reflected in how the course is graded (individual or two-person

exercises make up 20% of your grade, while reading comprehension quizzes and the midterm count for 20% each).

Big Ideas

This course is organized around three big ideas:

1. **Data science is about solving problems.** All too often, data scientists get lost in the technical details of models and lose sight of the bigger picture. Data science is not about maximizing accuracy or AUC scores — it's about using data and quantitative methods to solve problems, and at the end of the day the only "metric" that matters is whether your work has solved the problem you set out to address.
2. **Data scientists solve problems by answering questions, and the question you are asking determines what tool is appropriate.** At their core, all data science tools are tools for answering questions, whether you realize it or not. Learning to recognize how data scientists use questions to solve problems — and exactly what questions are being answered by the tools you use every day — is key to navigating the ambiguity of real-world problem solving.
3. **Reasoning rigorously about uncertainty and errors is what differentiates good data scientists from great data scientists.** Data science isn't just about minimizing classification errors and uncertainty — it's also about deciding how unavoidable errors should be distributed, and how to weigh the risks and trade-offs inherent in probabilistic decision-making rigorously and in a manner that takes into account the problem you are trying to solve.

Pre-Requisites for Non-MIDS Students

This course is primarily designed for students in the Duke Masters in Interdisciplinary Data Science (MIDS) program, but students from other programs are more than welcome if they have the appropriate pre-requisite training. Data Science is a fundamentally interdisciplinary field, so the more perspectives we have represented in the classroom the better!

This course will assume that enrolled students have a good grasp of inferential statistics and statistical modeling (e.g. a course in linear models), though no prior experience with causal inference is expected. In addition, MIDS students will be taking a concurrent course in applied machine learning, so incoming students will also be expected to have some basic experience with machine learning or be concurrently enrolled in an applied machine learning course.

This course will also assume students are comfortable manipulating real-world data in Python. The substantive content of this course is language-independent, but because students will be required to work on their projects in teams, comfort with Python will be required to facilitate collaboration (MIDS students are, generally, "bilingual" in R and Python, but have a strong preference for Python, and it's hard to write problem sets to accommodate multiple languages).

Finally, students will also be expected to be comfortable collaborating using git and github. If you meet the other requirements for this course but are not familiar with git and github, this is a skill you should be able to pick up on your own in advance of the course without too much difficulty. You can read more about git and github here. The Duke Center for Data and Visualization Science also hosts git and github workshops if you are a Duke student.

Course Schedule

You can find the course schedule here. Note that our schedule is subject to change, but this should give you a good sense of the material we will cover.

Syllabus

To learn more about the course, please read the course syllabus here..

Class Schedule

Class: Tuesday / Thursday, 1:25-2:40pm

Office Hours:

- Nick: Thursday, 9:30-10:30am, https://duke.zoom.us/my/nickeubank or Gross Hall 231

Texts Used

This course will make use of readings from a handful of different sources. The main four, in order of the amount they will be used, are:

- **DS4H**: Data Science for Humans (This website, free.)

- **Cunningham**: Causal Inference Mixtape ($30 new, $20 used. Online Access through Duke Library)
- **KTX**: Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing ($40 new, $30 used.)
- **GHV**: Regression and Other Stories ($40. Too new to have used versions.)

Date	Topic	Do Before Class	In Class
Th Jan 9	Course Overview		acme
Tu Jan 14	Solving Problems by Answering Questions	• Read, sign, submit syllabus on gradescope. • How to Read • Solving Problems	generating
Th Jan 16	Stakeholder Management	• Solving the Right Problem • History of Data Science • Why It's Important To Get Your Question Right (30 min video)	stake
Tu Jan 21	Proscriptive v. Descriptive Questions	• Descriptive versus Prescriptive Questions	
Th Jan 23	Exploratory Questions: Purpose, Internal and External Validity	• EDA • Using Exploratory Questions • Answering Exploratory Questions • Internal v. External Validity	exp
Tu Jan 28	Capstone With Training Wheels Assignment		• Explor Assign • How to write to stakeholders
Th Jan 30	Teams 1	• What Project Aristotle Learned • Edmondson, The Fearless Organization, Chpt 1 (on Canvas)	
Tu Feb	Teams 2	• Review Team Charter Assignment	

Date	Topic	Do Before Class	In Class
4		• Edmondson, Teaming, Chpt 2 from "Cognitive Barriers to Teaming" (p. 82) to end of Chapter (on Canvas) • Cross Cultural Intelligence • When and How to Escalate > Optional: Fostering Psychological Safety Tips	
Th Feb 6	Passive Prediction Questions: Purpose and Alignment	• Using Passive Prediction • Answering Passive Prediction	• Pass
Tu Feb 11	Passive Prediction Questions: Internal and External Validity	• Right Way To Be Wrong	
Th Feb 13	Giving Feedback	Feedback Exercise?	Feedback Exercise?
Tu Feb 18	Passive Prediction Questions: Loss Functions		• admissions • interp
Th Feb 20	Passive Prediction Questions: Interpretability	• Stop Using Black Boxes • Black box models require trusting data too > **Optional:** > > • Combining explicit models and black boxes in Scientific ML	

Date	Topic	Do Before Class	In Class
Tu Feb 25	Causal Questions: Purpose	• Using Causal Questions • Answering Causal Questions	
Th Feb 27	Causal Questions: Potential Outcomes	• Potential Outcomes Framework	po
Tu Mar 4	Causal Questions: Potential Outcomes	• Cunningham, Chpt 4, pp 135 (“Independence Assumption”) - 148 (Stop at “Randomization Inference”) • Indicator Variables (Note *SDO* (Simple Difference in Outcomes) in Cunningham same as “*widehat{ATE}* from class) > **Optional:** > • Read Chpt 4 in Cunningham from start (different presentation of potential outcomes)	resume
Th Mar 6	Causal Questions: Experiments in Practice, Internal Validity	Experiments: Internal Validity (In Practice): • KTX: Chpt 2 (End to End Example) • KTX: Chpt 3, “Threats to Internal Validity” (p. 42-47) • KTX: Chpt 19 (A/A Testing)	
Tu Mar 11	Causal Questions: Experiments in Practice, External Validity	Overall Evaluation Criteria: KTX Chpt 7. Finishing Internal Validity:	abtest

Date	Topic	Do Before Class	In Class
		• Different Randomizations: KTX Chpt 22 Experiments: External Validity (In Practice): • KTX, Chpt 3, "Threats to External Validity" to end (p. 47-54) • Kohvai, Tang and Xu, Chpt 23 (Primacy Effects / Long Term Decay)	
Th Mar 13	Causal Questions: Experiments in Practice, Design	Designing Experiments • Statistical Power and Sample Sizes: GHV Ch 16 • Don't stop experiments early!	power p-value preview
Tu Mar 18	**NO CLASS**		
Th Mar 20	**NO CLASS**		
Tu Mar 25	Causal Questions: Experiments, Bayesian Analysis	AB Testing Review	
Th Mar 27	Causal Questions: Experiments, Bayesian Analysis & Decision Theory	• Statistical Decision Theory (on Canvas). 550-556 (This is same as IDS 705 Lecture 8 Reading)	
Tu Apr	**MIDTERM**	**MIDTERM**	

Date	Topic	Do Before Class	In Class
16			
Th Apr 3	Causal Questions: Regression	• Causal Beyond Experiments	incomeineq
Tu Apr 8	Causal Questions: Matching	• The Why of Matching • The How of Matching Summary • Methods for Matching *Watch the video above from about 15 minutes in (where link starts) till at least 45 minutes in, keep going if you want to learn about propensity score matching problems.*	matching
Th Apr 9	Causal Questions: Differences and Differences / Panels	• Cunningham, Chpt 9 pp 406 (Difference in Differences) - pp 433 (Stop at "Importance of Placebos in Diff-in-Diff") > -(Book page numbers at bottom of PDF on canvas. Full chapter is in PDF, more than you need to read.) > Optional but encouraged: (dont need to follow everything, but here's a real diff-in-diff) > • Enfranchisement and Incarceration > • Diff-in-Diffs at Netflix	
Tu Apr	Interpretation of Results	• **Revised Exploratory Due along with Causal Proposal**	ev

Date	Topic	Do Before Class	In Class
Wed, April 23		• **Optional Due Date for Causal Rough Draft (11pm)**	
Wed, Apr 30		• **Final Causal Report Due**	

Solving Problems with Data

Few fields have shown as much promise to address the world's problems as data science. Today, data science is improving our understanding of and adaptation to climate change. It is being used in medicine to speed drug discovery, improve the quality of X-rays and MRIs, and ensure that patients receive appropriate medical care. It is used in courtrooms to fight for fair elections and electoral maps and by data journalists to document and communicate the injustices prevalent in our criminal justice system and issues in policing.

Data science also enables new technologies that may improve our lives. Autonomous drones are delivering blood and medical supplies to rural health clinics from Rwanda to North Carolina, and driver-aid features continue to make progress in reducing the over 30,000 traffic deaths and millions of injuries that occur in the US alone every year. And nearly every facet of business — from the way businesses source materials and manage inventory to the way product offerings respond to customer behavior — has been reshaped by data science.

At the same time, it is also increasingly clear that today's challenges will not be met by "throwing data science at the problem" or "just adding AI." According to a 2020 MIT/BCG survey, only 11% of businesses that had piloted or employed Artificial Intelligence (AI) had reaped a sizeable return on their AI investments. Businesses and regulators are also coming to appreciate the potential of data science tools to reinforce racial and gender inequities. Algorithms at Amazon have been found to discriminate against female job applicants. Medical algorithms have been found to prioritize White patients over Black patients for kidney transplants and preventative care. In the criminal justice system, algorithms have been found to incorrectly identify Black defendants than White defendants as being a "danger to society" when providing risk assessments to judges deciding on pre-trial

release, bail and sentencing. And even Meta's own research has shown its algorithms drive political polarization and division among users, and push users into extremist groups.[1]

How, then, should a burgeoning data scientist approach this discipline, full of such promise and peril? And why have so many data science endeavors failed to deliver on their promise?

The Three Big Ideas

This book is my answer to these questions, and it is organized around three big ideas:

1. **Data science is about solving problems.** All too often, data scientists get lost in the technical details of models and lose sight of the bigger picture. Data science is not about maximizing accuracy or AUC scores — it's about using data and quantitative methods to solve problems, and at the end of the day the only "metric" that matters is whether your work has solved the problem you set out to address.
2. **Data scientists solve problems by answering questions, and the question you are asking determines what tool is appropriate.** At their core, all data science tools are tools for answering questions, whether you realize it or not. Learning to recognize how data scientists use questions to solve problems — and exactly what questions are being answered by the tools you use every day — is key to navigating the ambiguity of real-world problem solving.
3. **Reasoning rigorously about uncertainty and errors is what differentiates good data scientists from great data scientists.** Data science isn't just about minimizing classification errors and uncertainty — it's also about deciding how unavoidable errors should be distributed, how to weigh the risks and trade-offs inherent in probabilistic decision-making rigorously and in a manner that takes into account the problem you are trying to solve, and to take uncertainty into account when acting on data.

From the summary of these three ideas, it should be immediately obvious that this book is different from many other data science books you may have read. Where most data science books are designed to teach specific data science techniques or methods, the aim of this book is to provide readers with a framework for thinking about their goals and how to achieve them using data science methods. It is, in a sense, about everything you need to know *beyond* the technicalities of model fitting. It is about what comes *before* and *after* you fit your model: it will help you decide what questions to answer, what data to collect, and what models to consider using *before* you actually fit your model, and it will help you learn to evaluate whether a result is likely to generalize,

whether a model is safe to deploy, and how to communicate those facts to a stakeholder *after* model fitting.

The importance of these skills is often underestimated by data science students, and for understandable reasons. Data science curricula usually begin with coding, statistics, and model evaluation techniques. As a result, the hardest part of data science classes is often mastering the technical details of model implementation. Moreover, the limited time available to instructors and the need to support full classes of students means data science exercises almost always have to come with clear directions and problem scaffolding to ensure students meet their learning goals.

But real-world problems don't come with directions. Indeed, a problem that is clearly defined and for which a solution is obvious isn't a problem anyone will pay you very much to solve. No, classroom exercises are carefully structured to foster learning and to make it possible for instructors to grade and provide feedback at scale. But real problems — the kind you will encounter in industry, government, or research — are hard to even articulate clearly, never mind solve. And that is why, as we will see, what really sets exceptional professional data scientists apart is not their ability to get a high AUC — **it's their ability to navigate and thrive in the face of ambiguous problems and goals.**

This Is A Book About The Forest, Not The Trees

The goal of this book, to frame things a little differently, is to help young data scientists maintain perspective. Students spend so much time learning individual techniques that they are unable to see the forest for trees. But this is not a book about individual techniques — it's about learning to think strategically about how to use those techniques to solve real problems.

To maintain its focus on the "forest," this book *takes as given* that you have already been introduced to statistical inference and machine learning and know how to faithfully fit a model in a robust manner. That means that topics like hypothesis testing, cross-validation, how to use train-test splits, and how to evaluate a model's AUC will be treated as assumed knowledge.[2] This is in no way meant to suggest these topics aren't important — we will reference them constantly — just that I will not attempt to teach them here, both to maintain focus on the goals of this book, and also because there already exist many other resources that introduce these topics better than I could.

Introduction Structure

The remainder of this introductory chapter contains an overview of the Big 3 ideas of the book. All concepts discussed here will also be covered in greater detail in future readings, but before we dive into them in detail, it's helpful to get a sense of the overall approach we will be taking!

---

[1] Recent reporting by the Wall Street Journal has shown that Facebook's own research has confirmed what many outside experts have long argued: the way its recommendation engines prioritize content that results in "user engagement" (clicks, shares, comments) ends up promoting partisan, polarizing, sensationalist, or extreme content. In addition, their own research has also shown that group recommendations are contributing to extremism. According to one internal presentation, "64% of all extremist group joins are due to our recommendation tools" like *Groups You Should Join* and other discovery tools.

[2] If you are a Duke student reading this, it's ok if you are not yet familiar with all of these topics so long as you are taking a good machine learning course — like IDS 705 — concurrently.

Idea 1: Solving Problems

Given the focus of data science coursework on the math, statistics, and programming that is required to use data science tools, one would be forgiven for thinking that understanding these tools is the ultimate goal of data science.

At the end of the day, however, your success as a data scientist will not be evaluated by whether you write good code or whether your classification model's AUC score is high. No, your success as a data scientist will always be evaluated based on a much simpler criterion: **did you make your stakeholder's life better by solving a problem they faced?**

On the last page, I said that that one of the key goals of this book is to help young data scientist's understand the broader context of their work. There is perhaps no better way to do that then to always keep this one fact front of mind. The things you practice in class exercises — writing good code and finding a classification model with a high AUC — are things that may *help* you achieve the goal of

Your success as a data scientist will *always* be evaluated by one simple criterion: **did you make your stakeholder's life better by solving a problem they faced?**

solving your stakeholder's problem, but they are never sufficient in and of themselves. Indeed, it is hard to overstate how common it is to see talented young data scientists write brilliant, performant code and fit extremely accurate models that predict the wrong property, solve a problem that isn't actually core to the stakeholder's needs, require data not available in deployment, or work in training but won't generalize to the stakeholder's deployment context. The data scientists on these projects often deployed the skills they learned in technical classes magnificently, but this execution was not well-directed in service of the stakeholder — the classic "not seeing the forest for the trees" mistake.

> **Note**
>
> Throughout this book, I will frequently use the term "stakeholder" to refer to the person whose problem that you, the data scientist, is seeking to address. I use this term because, as a young data scientist, you will often be in the position of having to use your data science skills to help someone else. Thus your stakeholder may be your manager, your CEO, or someone at another company you are advising.
>
> However, if you're lucky enough to *not* be directly answerable to someone else, either because you work for yourself or because you're in a field that gives you substantial autonomy like academia, you can simply think of your "stakeholder" as yourself.
>
> If you're interested in developing a consumer-facing product (e.g., you're an independing developer whose thinking of creating a new data-science-based web app), you may also find it useful to think of your customer of the stakeholder, since very few products are successful if they don't solve a problem customers face.

Specifying The Problem

How, then, can we be sure our hard work as data scientists serves the interests of our stakeholder?

The first step in solving any problem is *always* to carefully specify the problem. While this may seem obvious, properly articulating the core problem one seeks to address can be remarkably difficult. Moreover, because everything you will do *after* articulating your problem is premised on having correctly specified your objective, it is *the* greatest determinant of the success of your project. The most sophisticated, efficiently executed, high precision, high recall model in the world

isn't worth a lick of good if the results it generates don't solve the problem you or your stakeholder need solved.

Specifying your problem not only ensures that your subsequent efforts are properly directed, but it can also radically simplify your task. Many times problems only *appear* difficult because of how they are presented. As Charles Kettering, Head of Research at General Motors from 1920 to 1947 once said, "A problem well stated is a problem half solved."

How do you know if you've "clearly articulated the problem," and how should you go about refining your problem statement with your stakeholder? Those are topics we will discuss in detail in the coming chapters, as well as strategies for using data to help inform this process through iterative refinement of your understanding of the contours of the problem space.

Stakeholder Management and Domain Expertise

Application is, in many ways, what sets data science apart from the disciplines of mathematics, computer science, and statistics. And to apply the tools of data science tools effectively, by definition, requires an understanding of the domain to which your tools are being deployed. This fact makes many data scientists uncomfortable — after all, many young data scientists do not even know the industry in which they will be employed when the graduate, never mind feel they can lay claim to being a "domain expert" in any specific substantive field.

A consequence of this discomfort is that "domain expertise" is that the term "domain expertise" is often invoked as a way of abdicating responsibility for part of an analysis as "somebody else's problem" (a dangerously powerful construction, as explained by the incomparable Douglas Adams).

But it would be a mistake to view domain expertise as beyond the responsibility of the data scientist for two reasons. First, engaging with the details of a substantive domain to tailor the application of data science methods to solve a specific problem isn't ancillary to the job of a data scientists — it's a data scientist's comparative advantage. Data scientists (generally) are not the most technically skilled mathematicians, statisticians, or programmers — we are professionals who specialize in taking the best insights from all three of these skills sets and adapting them to fit the specific needs of a specific problem. So embrace the role of "domain knowledge!"

The second reason is that while you are unlikely to be a "domain expert" yourself, learning to solicit important information about a domain from true domain experts is a skill unto itself. As we will detail in our readings on "stakeholder management," stakeholders may be experts in their particular

substantive domain, but because they (usually) won't understand data science, they are unlikely to ever provide you with the domain knowledge you need to do your job successfully (and you can't get away with just saying "yes, this project failed, but it was because you didn't tell me X!" See the discussion above of the single criterion used to evaluate data scientists). So learning to *partner* with your stakeholder and domain experts to understand a problem is a key part of understanding it properly, and thus eventually solving it.

Idea 2: Solving Problems by Answering Questions

Once you've clearly articulated the problem you wish to solve, the next step is... to solve it! But how best to do so?

As data scientists, we are somewhat restricted in the types of solutions to which we have access. Nobody expects a data scientist to discover a new semiconductor manufacturing technique or solicit donors for cancer research funds.

But what data scientists can do is **answer questions about the world.** And while that may not seem inspiring at first, it's a remarkably powerful ability. That's because whether it's trying to answer the question "does this patient have disease X?", "what would happen to profits if we changed our pricing strategy?", or "do universal income programs reduce long-term poverty?", it turns out that a great many problems faced by both businesses and society as a whole become much easier to solve if we can better understand the consequences of our actions or reduce our uncertainty about the world.

In light of that fact, we can reframe the challenge of a data scientist from the more amorphous task of "figuring out how to solve the problem" to the more concrete "what question, if answered, would make it easier to solve this problem?" Once we've articulated a question to answer we can turn to choosing the best tool for generating an answer. But it is worth emphasizing this point — it is only at this stage of our project—not at the beginning!—that we start thinking about what statistical method, algorithm, or model is appropriate for the task.

But what if my stakeholder wants me to do something other than answer a question about the world? What if they want me to write a model to automatically read x-rays, or detect fraudulent transactions?

Answering Questions

The challenge of a data scientist is to figure out "what question, if

While it may not be immediately obvious how these fit in the "data scientists solve problems by answering questions about the world" framework, I can assure you they do. Indeed, anything you can think of that a data scientist might do ends up fitting this model because — as we'll discuss in detail below and later in the book — *all* data science models and algorithms can be understood as instruments designed to answer very specific, very concrete questions about data. And once you come to appreciate that fact, not only will this frame make more sense, but you will probably also find you understand many of the models you work with regularly more intuitively than you did before.

answered, would make it easier to solve this problem?"

Types of Questions

There are three types of questions we will explore in this book, each of which helps to solve problems in a different way:[1]

- Exploratory Questions: Questions about patterns and structure in the world.
  - Help us to understand the problem space better and prioritizing subsequent efforts.
- Passive Prediction Questions: Questions about likely outcomes for individual observations or entities.
  - Useful for targeting individuals for additional attention or automating certain tasks.
- Causal Questions: Questions about the consequences of actions or interventions being considered.
  - Useful for deciding on appropriate courses of action.

Each of these can play a different but important role in solving problems, and any effort to answer a question of each type will raise similar issues that need to be considered. As summarized next, by recognizing the *class* of questions we are seeking to answer, we can significantly narrow both the set of data science tools that are appropriate to consider and provide a short list of common considerations to think through.

Exploratory Questions

Once you have settled on a problem you wish to address, the next step is often to use data science to better understand the contours of the problem in order to better prioritize and strategize your

efforts. As data scientists, our best strategy for this type of investigation is to ask questions about general patterns related to your problem — what I call *Exploratory Questions*.

Why is this necessary? Well, as we'll discuss in a future reading on "stakeholder management," you would be *shocked* at how often stakeholders have only a vague sense of the patterns surrounding their problem. This makes refinement of your problem statement (and thus prioritization of your subsequent efforts) impossible. So before you get too far into any data science project, it's important to ask Exploratory Questions to improve your understanding of how best to get at your problem.

To illustrate, suppose a company hired you because they were having trouble recruiting enough high-quality employees. You *could* ask for their HR data and immediately try to train a neural network to... well, I'm not even sure what you'd want to train it to do right off that bat! And that's a big part of the problem. Getting more high-quality employees is a very general problem, and you could imagine addressing it in any number of ways — you could try and get more people to apply for the job in the first place, you could try and get a *different type* of candidate to apply then is currently applying, you could try and get more high-quality people who are given job offers to accept those offers, or you could help try to increase the number of people who are hired who turn out to be successful hires! But which should you do first?

To help answer this question, we can start by asking a series of Exploratory Questions that, when answered, will aid in your efforts to solve your stakeholder's problem:

- How many job applications are you receiving when you post a job?
- What share of your current job applicants are of high quality?
- If your current applicants come from different sources (online ads, services like Indeed, outreach to colleagues for recommendations, etc.), what share of job applicants from each of these sources are of high quality?
- What share of employees you try to hire accept your offer?
- What share of employees you do hire turn out to be successful employees?

Suppose, for example, only 10% of applicants who receive job offers accept. Then clearly that would seem a place where intervention would be likely to substantially increase the number of high-quality employees being hired. If, by contrast, 95% of applicants accept offers, then that is clearly not a place where you would want to focus.

Similarly, if most applicants are high quality and there just aren't enough of them, then you would probably want to focus your efforts on increasing the number of people who apply in the first place. But if only 2% of applicants seem appropriate to the company, then maybe focus should be put on changing *who* is applying for positions with an eye towards increasing the average quality of applicants.

Answering these questions will likely not, on its own, make it clear exactly where to focus your efforts. Your stakeholder may look at the fact that only 2% of applicants are appropriate and say "That's fine — we have so many applications that the *absolute number* of quality applicants is actually high enough, and it's easy to filter out the bad applicants." But these are numbers you can bring back to your stakeholder to discuss and use to zero in on the specific facet of their problem that is most amenable to an impactful solution.

Generating answers to these types of Exploratory Questions doesn't have the same "coolness factor" as using expensive GPUs to train deep learning models. But it is precisely this type of analysis that will help ensure that when if you *do* later run up a giant bill renting GPUs, at least that money will have been spent addressing a part of your stakeholder's problem that matters.

So how does one go about answering Exploratory Questions? As we'll discuss in later chapters, at times Exploratory Questions can be answered with simple tools, like scatter plots, histograms, and the calculation of summary statistics like means and medians. Other times, however, it may require more sophisticated methods, like clustering or other unsupervised machine learning algorithms that can, say, identify "customer-types" in a large dataset of customer behavior.

Regardless of the tool used, however, the goal is always to identify patterns in the world that are salient to understanding your stakeholder's problem.

Note

Answering Exploratory Questions is *not* synonymous with "Exploratory Data Analysis" (EDA). As it is commonly understood and practiced by students, EDA refers to the process of poking around in a new data set before fitting a more complicated statistical model. It entails learning what variables are present, how they are coded, and *sometimes* looking at general patterns in the data prior to model fitting. Crucially, it is generally defined by what it is *not* — it is what you do *before* you fit a complicated model — and by the tools used — EDA consists of plotting distributions or cross-tabulations, not fitting models or doing more sophisticated analyses.

Answering Exploratory Questions, by contrast, is about achieving a substantive goal — learning about patterns *in the world* — not the tools used to do it, or what it comes before. Answering an important Exploratory Question may require you to actively seek out new data, merge data from different sources together, and potentially do novel data collection. It may also entail model fitting or use of unsupervised machine learning algorithms to uncover latent patterns.

We will discuss the conceptual issues surrounding EDA in more detail in a later reading.

Passive Prediction Questions

Answering Exploratory Questions helps you to prioritize your efforts and improve your understanding of your stakeholder's problem. Often you will bring the answers you generate to Exploratory Questions back to your stakeholder and use them to refine your problem statement in an iterative loop. "Yes, your broader problem is your profit margins are too thin, but after looking at your cost structure, it seems that the biggest driver of costs is the labor that goes into product packaging. So let's focus on minimizing that." But what then? Simple! We return to asking "What question, if answered, would help solve my problem?"

The second class of questions which, if answered, often help solve stakeholder problems are *Passive Prediction Questions*. Passive Prediction Questions are questions about the future outcomes or behaviors of *individual entities* (people, stocks, stores, etc.). This type of question may take the form "Given this new customer's behavior on my website, are they likely to spend a lot over the next year?" or "Given the symptoms of this patient and their test results, how likely are they to develop complications after surgery?" (Don't worry about the "passive" part of the name —

we'll circle back to why that's there below. For the moment, you may find it helpful to just think of these as "Prediction Questions.").

Because Passive Prediction Questions are questions about individual entities, they don't necessarily have one "big" answer. Rather, Passive Prediction Questions are answered by fitting or training a model that can take the characteristics of an individual entity as inputs (e.g., this patient is age 67, has blood pressure of 160/90, and no history of heart disease) and spitting out an answer *for that individual* (given that, her probability of surgical complications is 82%). This differentiates Passive Prediction Questions from Exploratory Questions, which are about global patterns, not individual level predictions.

With Passive Prediction Questions, the first question to ask is often one of feasibility: "Given data on new customer behavior on my website, **can I** predict how much they are likely to spend a lot over the next year?" But you then answer that question by training a model that can answer the question you really care about for any given customer: "Given this new customer's behavior on my website, are they likely to spend a lot over the next year?"

This ability to make predictions about future outcomes is obviously of tremendous use to stakeholders as it allows them to tailor their approach at the individual level. A hospital that can predict which patients are most likely to experience complications after surgery can allocate their follow-up care resources accordingly. A business that knows which customers are more likely to be big spenders can be sure that those customers are given priority by customer care specialists.

But the meaning of the term "Prediction" in Passive Prediction Questions extends beyond "predicting the future". Passive Prediction Questions also encompass efforts to predict how a third party *would* behave or interpret something about an individual if given the chance.

For example, suppose our hospital stakeholder wanted to automate the reading of mammograms so that rural hospitals without full-time radiologists could give patients diagnoses more quickly (or, more cynically, pay fewer radiologists).[2] How does a model that reads mammograms "answer a question"? Well, in a very meaningful way, if you train your model by feeding it a dataset of mammograms that have been labelled as "normal" or "abnormal" by radiologists, then what that model is learning to do is answer the question: "if a radiologist looked at this particular scan, would they conclude it is abnormal?"

The value of this type of prediction to stakeholders is likely also self-evident, as it opens the door for automation and scaling of tasks that would otherwise be too costly or difficult for humans. Indeed, answering this question is the type of task for which machine learning has become most

famous. Spam filtering amounts to answering the question "If the user saw this email, would they tag it as spam?" Automated content moderation amounts to answering "Would a Meta contractor conclude the content of this photo violates Facebook's Community Guidelines?" Indeed, even Large Language Models (LLMs) like chatGPT, Bard, and LLaMA can be understood in this way, as we will discuss later.

> **Note**
>
> Thinking of training an algorithm to read a mammogram as a model that answers the question "if a radiologist looked at this particular scan, would they conclude it is abnormal?" may seem a little strange at first. But this framing is both more accurate and more conceptually useful than the more common frame of "this is a a model that detects abnormal mammograms." That's because when a data science problem is solved by training a model on data on past human behavior, it isn't actually learning to "detect cancer" — it's learning to emulate the behavior of the humans in the training data.
>
> This distinction is subtle, but it is important because it helps us to understand why any model we train in this way will inherit all of the biases and limitations of the radiologists who created the data used to train the algorithm. If, for example, our radiologists were less likely to see cancer in denser breast tissue, that bias would also be inherited by the algorithm.
>
> (We call the inevitable existance of some difference between between what we *want* the algorithm to do — in this case, detect cancer — and *what it is actually being trained to do* — predict how a radiologist would interpret the scan — an "alignment problem.")

It is worth emphasizing that the distinction between Exploratory Questions and Passive Prediction Questions is a distinction *in purpose*, not necessarily a distinction in the statistical tools that are most appropriate to the task. A linear regression, for example, may be used for answering either type of question, but in different ways. To answer an Exploratory Question, we might look at the coefficients in a linear regression to understand the partial correlations between variables in the data. To answer a Passive Prediction Question, we might only look at the predicted values from the regression model.

But even if the same *type* of model can be used for both purposes, how one *evaluates* the model depends entirely on the purpose to which it is being put. When answering an Exploratory Question through the interpretation of regression coefficients, the size of the standard errors on the

coefficients is critical. When making predictions, by contrast, one may not care about the coefficients of a model at all! So long as the R-squared is high enough (and other diagnostics seem good), one can simply use the predicted values the regression generates without ever looking "inside the box."

As such, there's no simple mapping between statistical or machine learning methods and the type of questions you aim to answer. With that said, the study of how to answer Passive Prediction Questions is often referred to as "supervised machine learning."

Causal Questions

The last category of question that data scientists are commonly called upon to answer are *Causal Questions*: questions about what the *effect* might be if a certain action is taken. For example, "What would be effect be of patients with disease X taking medication Y?" or "what would the effect of changing the interface of my app be on user retention?"

Causal Questions most often arise when a stakeholder wants to do something — buy a Superbowl ad, change how a recommendation engine works, authorize a new medical device — but they fear the action they are considering might be costly and not actually work. In these situations, stakeholders will often turn to a data scientist in the hope that the scientist can "de-risk" the stakeholder's decision by providing guidance on the likely effect of the action *before* the action is undertaken at full scale.

Causal Questions, therefore, take the form of "What is the effect of an action X on an outcome Y?"—or more usefully, "If I do X, how will Y change?". Nearly anything can take the place of X and Y in this formulation: X could be something small, like changing the design of a website, or something big, like giving a patient a new drug or changing a government regulation. Y, similarly, could be anything from "how long users stay on my website" or "how likely are users to buy something at my store" to "what is the probability that the patient survives".

In my view, Causal Questions are perhaps the hardest to answer for two reasons. The first is that when we ask a Causal Question, we are fundamentally interested in *comparing* what our outcome Y would be in two states of the world: the world where we do X, and the world where we don't do X. But as we only get to live in one universe, we can never perfectly know what the value of our outcome Y would be in *both* a world where we do X and one where we don't do X—a problem known as the **Fundamental Problem of Causal Inference** (causal inference is just what people call the study of how to answer Causal Questions).

But the second reason is Causal Questions land on the desk of data scientists when a stakeholder wants to know the likely consequences of an action *before they actually undertake the action at full scale*. This may seem obvious, but it bears repeating — not only is answering Causal Questions hard because we never get to measure outcomes in both a universe where our treatment occurs and also a universe where it does not (the Fundamental Problem of Causal Inference), but answering Causal Questions is *also* hard because stakeholders want to know about the likely consequences of an action they aren't ready to actually undertake!

As a result, the job of a data scientist who wants to answer a Causal Question is to design a study that not only measures the effect of a treatment but also does so in a setting that is enough like the context in which the stakeholder wants to act that any measured effect will generalize to the stakeholder's context.

Note

Now that we've discussed Causal Questions, we can discussion the term "passive" in "Passive Prediction Questions." The term "Passive" in "Passive Prediction Questions" is meant to differentiate situations where a stakeholder wants to predict things about individuals they do not yet know *in a world where the status quo prevails and our behavior doesn't change*.

For example, when answering the Passive Prediction Question "Given their case history, how likely is this patient to experience post-surgical complications?" we don't actually want to know how likely they are to experience complications — we want to know how likely they would be to experience complications *absent any intervention*. Our hope, after all, is that by learning that a certain patient is likely to experience complications we can act to prevent that outcome!

Obviously, we might then want to build on the insights provided by that model and ask a followup Causal Question: "What would the effect be of assigning an extra nurse to patients predicted to be more likely to have post-surgical complications?"

But because these questions are different in their goals, we would also go about answering them in very different ways. For example, we might answer our Passive Prediction Question using historical patient data and a logistic regression or decision tree. And to answer our Causal Question, we might run a randomized experiment in which half the patients we predict as likely to experience complications are assigned an extra nurse and half are not, and we later compare patient outcomes.

An Example

In this introductory chapter alone, we've already covered a substantial amount of material. We've discussed the importance of problem articulation, the idea that the way data scientists solve problems is by answering questions, and the three types of questions data scientists are likely to encounter.

It's easy to see how this framework might result in a sequential development of a project. First, a hospital comes to you concerned about the cost of surgical complications. So you:

1. Work with them to more clearly define the problem ("Surgical complications are extremely costly to the hospital and harm patients. We want to reduce these complications in the most cost-effective manner possible.")
2. You answer some Exploratory Questions ("Are all surgical complications equally costly, or are there some we should be most concerned about?").
3. You develop a model to answer a Passive Prediction Question ("Given data in patient charts, can we predict which patients are most likely to experience complications?") so the hospital can marshal its limited nursing resources more effectively.
4. The hospital then comes back to you to ask the Causal Question "Would a new program of post-discharge nurse home visits for patients identified as being at high risk of complications reduce complications?"

In reality, however, while it is important that some steps come before others (if you don't start by defining your problem, where do you even start?), real projects are never so linear. The reality is that you will constantly find yourself moving back and forth between different types of questions, using new insights gained from answering one question to refine your problem statement and articulate new questions.

Nevertheless, by using this framework as a starting point, and using this taxonomy to help you recognize (a) the type of question you are asking, and (b) the reason you are seeking to answer a given question even when iterating through a project, you will see tremendous gains in your ability to please your stakeholders by staying focused on the problems they need addressed.

---

[1] Careful readers may notice that these categories do not include *should questions*, which are sometimes referred to as "prescriptive" or "normative" questions. As we will discuss in detail in an upcoming reading, that is because while data science is an amazing tool for characterizing the world around us, it cannot, on its own, answer questions about how the world *should* be. Answering "should questions" requires evaluating the desirability of different possible states of the world, and that can only be done with reference to a system of values, making them inherently subjective. Data science can help us predict the *consequences* of different courses of action, but it cannot tell us whether those consequences make a given course of action *preferable*.

[2] Mammograms are x-rays of breast tissue used for the detection of breast cancer.

Idea 3: Being Thoughtful About Being Wrong

The third big idea of this book is that one of the biggest differentiators of good data scientists and great scientists is not their ability to maximize the accuracy of their models, but to think rigorously about their models limitations and the errors they will inevitably commit.

To illustrate the type of fallacy that young data scientists fall prey to in their quest to maximize model performance, let us consider that most intuitive of metrics, *accuracy* — the share of classifications a model makes that are correct. In my role as the Admissions Chair for the Duke Masters of Interdisciplinary Data Science (MIDS) program, I read hundreds of essays a year from aspiring data scientists, and I long ago lost count of the number of times I have seen some version of the following passage with little to no additional context:

> While working in [person]'s lab, I fit a [logit/XGBoost/Random Forest/Deep Learning] model to the data and achieved an accuracy of 96%.

Or, in the applicant's resume, they report attaining an accuracy score in the 90s with some model.

I assume that these authors believe that this type of declaration reflects well on them — after all, 96% accuracy means only 4% of observations were mis-classified! But the truth is that reporting a high accuracy absent addition information about model performance actually tells me more about the author's failure to understand this third Big Idea of the book than it tells me about their modelling prowess.

To illustrate why, suppose I told you I had developed a model that could predict breast cancer in mammograms with an accuracy of over 90%. That'd be amazing, right? Mammograms cost roughly $10 *billion* dollars a year in the US alone,[1] so any reduction in need for skilled radiologists to review mammograms would be a huge win for women's health and healthcare costs, right?

Now suppose I told you that the model I wrote was this:

```
def my_cancer_detection_model(mammogram):
    is_scan_abnormal_maybe_cancerous = False
    return is_scan_abnormal_maybe_cancerous
```

How does that have an accuracy of 90%? Simple — according to the Susan G. Komen Society, roughly 90% of routine mammograms are normal and require no followup. Thus a "model" that reports all routine mammograms are normal will immediately achieve an accuracy score of 90%. In fact, the true accuracy of the model is actually higher than 90%, since most mammograms flagged as "abnormal" are determined to not be indicative of cancer after followup (e.g., after biopsies).

So, are you still impressed by my model's 90% accuracy?

Of course not. And to be clear, the problem with this model is *not* that its accuracy is only 90% and not, say 95%. The problem with this model is that *it has a False Negative Rate (the share of cases that are positive — in this case, mammograms from women with cancer — that are classified as cancer free) of 100%.* And since the thing we care about most in cancer screenings is not telling a patient with cancer they're fine (a False Negative), that's a huge problem!

OK, if what we care about is the False Negative Rate, shouldn't we just minimize the False Negative Rate? Not so fast! Consider this model:

```
def my_no_false_negative_model(mammogram):
    is_scan_abnormal_maybe_cancerous = True
    return is_scan_abnormal_maybe_cancerous
```

Oh dear. Yes, that model has a 0% False Negative Rate, but it also has a 100% False *Positive* Rate, meaning anyone subject to this screening would be told they might have cancer and needs followup diagnostic procedures! That's no good either.

No, a *good* model for reading mammograms needs to maximize accuracy while also balancing the desire to not subject healthy women to unnecessary procedures (minimize False Positives) *and* the desire to not fail to flag potentially cancerous scans. And that's what I mean when I say that a *great* data scientist is one who is thoughtful about how their models make mistakes. Determining what model is "best" requires more than optimizing a simple objective function — it requires thinking critically about the problem one is trying to solve (Big Idea 1), the substantive impact of different types of model errors in the context of that problem, and using that to inform model selection and evaluation.

---

[1] "Aggregate Cost of Mammography Screening in the United States: Comparison of Current Practice and Advocated Guidelines", *Annals of Internal Medicine*, February 2014.

What *is* Data Science: An Historical Perspective

Given how often the term "data science" gets thrown around, you would be excused for thinking that the meaning of the term was clearly understood. The reality, however, is that if you were to ask ten people working in the field you will almost certainly get ten different descriptions of what it is and what they do.

Part of that is deliberate obfuscation—data science is *so* trendy that everyone wants to claim that what they're doing is data science in order to woo venture capitalists or to win research grants. Indeed, it has been said (half-joking, half-seriously): "Data science is *Artificial Intelligence* when you're raising money, *Machine Learning* when you're hiring, and it's *Logistic Regression* when you actually have to get the job done.

But the ambiguity that surrounds the term "data science" is also the result of the fact that data science is not a mature discipline in the way that computer science, economics, or mechanical engineering are mature disciplines. And, as a young data scientist, that immaturity is important for you to understand, as it is both the source of some of the most exciting opportunities and also some of the biggest challenges you will face.

The Organization of Academia, Data Science, and You

To explain what the term data science means in practice, we have to start by discussing a bit of the inside-baseball[1] of how academia operates. This may feel esoteric, but it's important to understand because the way academia is organized has shaped the professional training — and thus the language and thought patterns — of most people you will encounter in the data science space. Understanding academia better, as a result, will not only help you understand the material you are exposed to in data science classes better, but also help you relate to your future peers and colleagues.

The idea that academia is deeply fragmented often surprises students, and understandably so. Universities *love* to pay lip service to the importance of interdisciplinarity and are quick to highlight successful interdisciplinary collaborations. But successful interdisciplinary collaborations are so notable precisely because they are the exception, not the rule. The reality is that academic

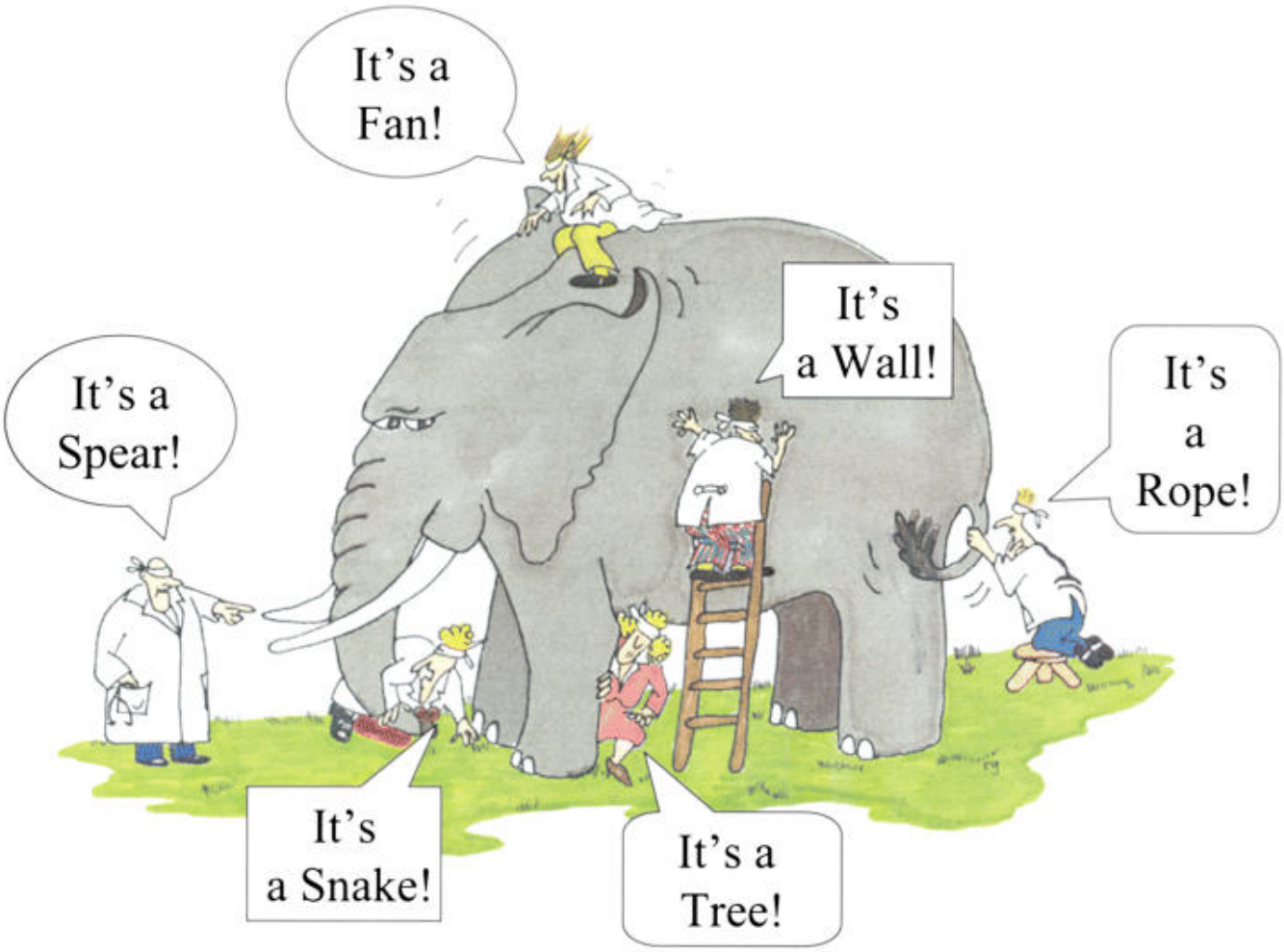
research is starkly divided into disciplinary silos (e.g., computer science, statistics, political science, economics, and engineering). This isn't because researchers aren't *interested* in interdisciplinary collaborations, but rather that their professional imperatives push them to focus their attention on the priorities and language of their own departments and disciplines.[2]

Thus, while the past several decades have seen an unprecedented emergence of new methods across all of academia, the lack of intellectual cross-pollination across academic silos has resulted in disciplines failing to take full advantage of discoveries from other disciplines. Over time, each discipline has developed a perspective on computational methods that emphasizes its own intellectual priorities.

To illustrate, suppose we were interested in using patient data to reduce heart attacks. A computer scientist looking at this problem might use their discipline's methods to *predict* which patients are most likely to experience a heart attack in the future using current patient data; a social scientist might focus on trying to understand the *effect* of giving patients a new drug on heart attack risk; and a statistician might focus on understanding *how confident* we should be in the conclusions reached by the computer scientist and social scientist.

This fragmentation has also resulted in a fragmentation of *language* around data science methodologies. Disciplines often come up with different terminology for the same phenomena, adding another layer of difficulty to efforts to work across departmental silos.

The result is a situation analogous to the Buddhist parable of the blind men and the elephant, wherein a group of blind people come upon an elephant, and upon laying hands on different parts of the elephant, they come to different conclusions about what lies before them. The person touching the tail declares "we have found a rope!", while the person touching the leg declares "we have found a tree!"



(*Note*: Not sure of original source of this image. Found it here, but need to figure out rights prior to anything about this becoming commercial! Lots of pics in public domain if needed, but not blindfolded scientists.)

And yet, as the poet John Godfrey Saxe wrote in his poem *The Blind Men and the Elephant* about this parable many centuries later:

> And so these men of Indostan, Disputed loud and long, Each in his own opinion Exceeding stiff and strong, Though each was partly in the right, And all were in the wrong!

In recent years, however, there has been a growing appreciation of what can be gained from pulling together the insights that have been developed in different fields, despite the challenges of language and professional imperatives to such collaborations. And, at least amongst those who are serious about the development of data science as a discipline and not just a buzzword to use when

raising money, is the promise of data science: to unify the different perspectives and methods for analyzing data. Or, to put it more succinctly: to finally see the whole elephant.

While the field is making progress towards "seeing the elephant as a whole," however, as a result of this fragmented origin story, *most* people you will encounter in the world doing data science were trained in one of these academic silos. That means that depending on who you are working with and how they were trained, you may find your future colleagues using terms you've never heard before. And when that happens, it's important to remember that while that *may* be because they're talking about a concept you've yet to encounter, it may also simply be because they're using different language for something you know. Similarly, you may also find senior colleagues unfamiliar with concepts that seem basic to you simply because you were exposed to perspectives that were alien to your colleague's academic silo at the time they were trained. Indeed, given that data science education *is* finally becoming more unified, you should probably expect to learn a lot of ideas that even your more senior colleagues (or rather, especially your more senior colleagues!) were never exposed to.

And therein also lies some of the greatest opportunities. Precisely because of this intellectual fragmentation, there are *lots* of opportunities for taking insights from one intellectual silo and using them to solve problems in another — a kind of "intellectual arbitrage," if you will.

The Data Analyst / Software Engineering Distinction

In addition to this broader intellectual fragmentation, the world of data science also often feels oddly fragmented around the way people use the tools of data science.

One model of data science is what we will call the "data analyst" approach. Data scientists doing this type of work often collect data to answer specific questions—what is the effect of expanded government health insurance subsidies on mortality? what type of customer should we target with our new advertising campaign?. As a result, when they write code, they write it to be run against a specific set of data to answer a specific question.

The other model is what we will call the "software engineering" approach. Data scientists doing this type of work write software they plan to *deploy* to thousands or millions of users. This is the type of work that gets embedded in the apps on your phone, or that generates your movie recommendations at Netflix. As a result, when these data scientists write code, they are writing more sophisticated and generalizable programs.

To be clear, most data scientists do at least a bit of both types of work—data analysts may often write small programs or packages to aid in types of analysis they do a lot, and software engineers have to prototype and test new programs before they write a version that can be deployed broadly. But most people will eventually choose to specialize in one direction or another, and when you see data science resources in the world—especially ones about programming for data science—bear in mind that depending on *your* proclivities towards on approach or another, not all resources will be well suited to your interests.

I also want to draw attention to this distinction because it's remarkable how dismissive most data scientists will be of the "other" type of data science, and I want to encourage you to both (a)not be so tribal yourself (both flavors of data science have their place, and help solve real world problems!), and (b) not be too surprised when you encounter people with irrationally strong opinions about which approach is the "right" approach to doing data science.

---

[1] "Inside baseball" refers to the discussion of the idiosyncracies and details of how an institution or system operates internally, something that is often not of interest to people who aren't part of the system.

[2] Nearly all university faculty are hired by established departments like statistics or economics, faculty submit their research to journals specific to their discipline, those journals in turn ask fellow members of the discipline to evaluate their work for publication, and promotions and tenure reviews are managed by the faculty in a faculty member's own department.

Solving the Right Problem

If data science is the study of how to solve problems using quantitative methods, then the first — and arguably most important — stage in any data science project is to define the problem to be solved.

While this may seem simple, it is often far from it. Indeed, as noted in the introduction, problems often only appear complicated because they are poorly understood. Reframing or rearticulating a difficult problem is often the key to figuring out how to solve it, which is why the adage "A problem well stated is a problem half solved" has remained popular for so long.

In this chapter, we will discuss what happens when data scientists fail to appreciate the importance of this stage of

A problem well stated is a problem half-solved.

a project. With that established, we will then turn to advice on how to approach problem articulation.

- Charles Kettering, Head of Research at General Motors

This chapter will be written as if you are the sole actor on a data science project, responsible for everything from problem articulation to execution. But of course, this will rarely be the case in your professional life as a data scientist, especially early in your career. With that in mind, in the next chapter we will turn to the topic of "Stakeholder Management" — the practice of refining your understanding of the problem to be solved with a stakeholder.

If You Don't Articulate The Problem

There are many examples in the world of data science projects going awry because the team behind them failed to properly articulate the problem they wished to solve. Rather than start with one of those true stories, however, I'd like to begin with one of my favorite fictional (and humorous) examples of the phenomenon.

In Douglas Adams' comedic sci-fi classic *Hitchhiker's Guide to the Galaxy*, a race of hyperintelligent pandimensional beings set out to build a massive supercomputer the size of a city to solve the mysteries of the cosmos once and for all. When they turned on the computer, named Deep Thought, they announced that:

> "The task we have designed you to perform is this. We want you to tell us... the Answer!"
>
> "The Answer?" said Deep Thought.
>
> "The Answer to what?"
>
> "Life!" urged one designer.
>
> "The Universe!" said another.
>
> "Everything!" they said in chorus.
>
> Deep Thought paused, then answered, "Life, the Universe, and Everything. There is an answer. But," Deep Thought added, "I'll have to think about it."

Seven and a half million years later, when Deep Thought had *finally* finished its calculations, the descendants of those designers assembled to learn the result of their ancestors' work.

> "Er ...good morning, O Deep Thought," said Loonquawl [one descendant] nervously, "do you have ... er, that is ..."
>
> "An answer for you?" interrupted Deep Thought majestically. "Yes. I have."
>
> The two [descendants] shivered with expectancy. Their waiting had not been in vain.
>
> "There really is one?" breathed Phouchg [the other descendant].
>
> "There really is one," confirmed Deep Thought.
>
> "To Everything? To the great Question of Life, the Universe and Everything?"
>
> "Yes. [...] Though I don't think," added Deep Thought, "that you're going to like it."
>
> [...]
>
> "All right," said the computer, and settled into silence again.
>
> The two fidgeted.
>
> The tension was unbearable.
>
> "Forty-two," said Deep Thought, with infinite majesty and calm.
>
> "Forty-two!" yelled Loonquawl. "Is that all you've got to show for seven and a half million years' work?"
>
> "I checked it very thoroughly," said the computer, "and that quite definitely is the answer. I think the problem, to be quite honest with you, is that you've never actually known what the question is."
>
> "But it was the Great Question! The Ultimate Question of Life, the Universe and Everything," howled Loonquawl.
>
> "Yes," said Deep Thought with the air of one who suffers fools gladly, "but what actually is it?"

> A slow stupefied silence crept over the men as they stared at the computer and then at each other.
>
> "Well, you know, it's just Everything ... everything ..." offered Phouchg weakly.
>
> "Exactly!" said Deep Thought. "So once you do know what the question actually is, you'll know what the answer means."[1]

In addition to establishing the premise for one of the greatest comedic science fiction novels in human history,[2] I feel this passage perfectly exemplifies the three reasons most data science projects fail.

The first is our absolute faith that technology will save us. All we have to do to solve any problem is feed it into the newest, shiniest AI/LLM/ML model available. Sure, there may be technical challenges associated with configuring the environment, formatting the data, etc., but fundamentally, all that lies between us and success is putting the problem into technology's hands.

The second is that when a data science project fails, it's rarely because the technology itself failed. Rather, **projects usually fail because people failed to ensure that the task they asked their model to accomplish would actually solve their problem.** Technology doesn't care if the task it's been given is useful to the user. It will do what it has been asked to do — no more and no less. Garbage in, garbage out.

Finally, this passage also gives a nod to the third — and perhaps most important reason — that data science projects fail: **data scientists are far too deferential to their stakeholders.** In this passage, we can see that Deep Thought recognizes the idiocy of the request it has been given by its stakeholders — Loonqual and Phouchg — but does nothing about it. And to the degree to which we can think of the Deep Thought personality as a stand-in for the data scientist, this is a troubling realistic illustration of how many data science interactions go. The stakeholder says they want something and the (usually younger) data scientist assumes their only responsibility is to ensure the stakeholder's request is implemented.

But as discussed in the introduction of this book, your success as a data scientist will *always* be evaluated in terms of whether you've made your stakeholder's life better. And if doing precisely what they ask does not improve their lives, that's the only thing that will matter. No one is blaming Deep Thought in this story, but they sure aren't excited about what it did, either.

Articulating and Reframing Your Problem

The preceding story gives (an admittedly extreme) example of how things can go wrong when a data science project is not well motivated — in short, large amounts of energy and sweat can go into creating technically impressive results that, in the end, in no way solve a problem or improve the lives of anyone involved.

How, then, can you avoid that fate? Every problem is different, but here are some strategies to employ when faced with a problem to aid in refinement, reframing, and articulation. As we work through these, we will also work a few examples based on real cases designed to help you get a feel for what this process entails and why it's so important.

1. How do you know if you've been successful?

One of the most helpful questions to ask when trying to understand a problem is "how would we know if we've successfully solved this problem?" This question is powerful because it abstracts away the question of *how* you might solve a problem, and instead focuses attention on the *goal* one is trying to achieve. In other words, the objective of this question is to figure out how you might *measure* success.

Crucially, the answer to this question shouldn't be something like "we have a good model for doing X" — that answer has largely just shifted the question to what constitutes "good." Rather, try to come up with an answer in terms of the *metric* or *behavior* that, if observed, would tell you that you've been successful.

To illustrate the potential power of this question, let's consider an example I've encountered in the real world (with some details changed to protect the anonymity of everyone involved).

The stakeholder is an online auction site for used construction equipment (front loaders, backhoes, etc.). They want to improve the algorithm that suggests prices to people selling equipment, but they've realized that while they have good data on how machinery type impacts prices, their algorithm can't take into account the condition of equipment being listed. So they've hired a data science team to train an image classification model on listings to estimate the condition of each listed piece of machinery.

So: how would you know if you've been successful in solving this stakeholder's problem?

At first blush, you might say I would know I was successful if I have a model that has high classification accuracy for equipment condition when fed photos of equipment listed in the past.

But look more carefully, is that really what the stakeholder cares about? No — because "image classification model" is something young data scientists are familiar with, they tend to latch on to it immediately. "Oh, great, image classification! I can do that. Let's go!"

No, the problem motivating the stakeholder is that their price suggestion algorithm doesn't take into account the condition of equipment going up for sale. Given that, you would know your model was successful if the predicted prices from the model were more in line with final sale prices for equipment of all conditions. *That's* the actual goal.

Yes, an image classification model might allow the stakeholder to estimate equipment condition which could then be used as training data to revise the pricing model. But is that the best approach?

To train a model that can differentiate construction machinery in good or bad condition, the first thing you'd need to do is have a person go through and label a lot of old listings as being in good or bad condition. And you'd probably need a lot of them — after all, your model would have to be able to differentiate between indicators of wear that are superficial (dirt, superficial rust, and peeling paint) and indicators of wear that are substantial (cracks, patch welds, etc.), and do so on a range of different types of construction equipment. Then you'd use that to train an image classifier, then you'd have to tune and validate that image classifier, then you could use its classifications to improve the pricing model.

Alternatively, you could just (a) have someone label some old listings and use those labels *directly* to refine the pricing model, and (b) add a drop-down menu that asks people listing equipment to report the condition of their equipment (so that data is available to the pricing model when someone goes to list a product). Congratulations! You've just *entirely* removed image classification from this task.

2. Abstract the Problem

A second strategy for better understanding a problem is to try to reframe it at a higher level of generality/abstraction. When you do, you may find your thorny problem is actually just a specific example of a more general type of problem, one that people smarter than either of us have already figured out how to solve.

Alternatively, you may realize that you or your stakeholder has (often unknowingly) introduced constraints to the problem that aren't actually constraints.

Perhaps my favorite example of this phenomenon comes from a talk given by Vincent Warmerdam at PyData 2019.

The World Food Program (WFP) is a global leader in food aid provision. As Vincent tells the story — which he reports having heard at an Operations Research Conference — the WFP was struggling with an extremely difficult data science problem: how best to get food from the places it was being grown/stored to the people who needed it most. Essentially, the WFP would receive reports of needs from communities facing food insecurity. One community might report a need for bread and beef, while another might request lentils and meat. The WFP would compile these requests and then set about trying to determine the most efficient way to meet these needs.

This type of logistics problem is an example of a notoriously difficult problem (essentially a version of the Traveling Salesman Problem, which is NP-Complete, if that means anything to you) that companies like FedEx and UPS buy supercomputers to address. But this particular problem was made extra challenging by all the different types of food the WFP was trying to provide communities.

What the WFP realized was that they didn't actually need to provide bread to the village asking for bread. See, humans don't need *bread* to avoid starvation — they need a certain number of calories, a certain amount of protein, and a handful of other nutrients.[3] So when a village asks for bread, rice, or wheat, you can instead think of them asking for carbohydrates. And a village asking for beef or beans is actually asking for protein and iron. So by simply abstracting the task from "How best can we meet all these food requests?" to "How best can we meet the nutritional needs indicated by these requests?" the WFP was able to *dramatically* reduce the number of constraints being imposed on the logistical optimization problem WFP needed to solve, making its task *far* simpler.

How far should you abstract things? To the level that feels most useful. Yes, in the private sector you *can* abstract almost any problem to "how do we maximize the lifetime discounted profits of the company," but I don't think it would be controversial to say that that formulation is not particularly useful if you're trying to pick the best box sizes for your company to stock for online orders. But there's also no real cost to thinking about your problem at that level of generality for a little while, so my advice would be: if you haven't gotten to a level of abstraction that feels silly, you probably haven't abstracted enough.

3. Let Your Articulation Change

My last suggestion is to allow your understanding of the problem you are trying to solve evolve. As your project develops, you will learn new things about your problem context, and you should allow those to inform how you are thinking of your problem. This is true not only for you — the data scientist — but also your stakeholder. We'll discuss stakeholder management more in our next chapter, but you should regularly be asking them: "does the way we've articulated the problem we're trying to solve still feel right to you given everything we've learned?"

---

[1] Yes, I recognize that it is wildly indulgent to open a chapter with such a long epigraph. But it's my book, and if there's anything to be indulgent about its quotes from *Hitchhiker's Guide to the Galaxy*, damn it!

[2] Not least for being the only 5-book trilogy of which I am aware!

[3] As I understand it, calcium, iron, vitamins A, B1, B2, C, and niacin.

What Solving the *Wrong* Problem Looks Like

Recruitment

Outliers

Pizza Ads

Our discussion up to this point has been a little abstract, so to illustrate what it means to "mis-specifying a problem." The details of this example are fictitious, but the underlying logic of this example is not; indeed, the insight illustrated by this example is central to one of the biggest pivots in how people think about online advertising.

You have been hired by the advertising division of a fictitious national pizza chain—let's call it Little Papa Dominos (LPD). LPD spends a *lot* on online advertising, but their resources aren't being

deployed as effectively as they could be. They spend more than most of their competitors, and yet their online sales are lagging.

After consulting industry groups and online advertising experts, they discovered that the rate at which people click their ads (their ads' *click-through rate*, or CTR) is well below the industry average.

To address the problem, they've hired you—a newly minted Data Scientist—to improve the CTR of their ads. They give you a large budget, access to all the cloud computing resources you need, and even a small staff.

"Well," you reason, "maybe the problem is that our ads aren't being shown to the right people. After all, it seems unlikely that any ad for pizza—no matter how appealing—is likely to draw a click if it's shown to a 75-year-old at 7 am." So you set out to build a statistical model to answer the question, "given a user's demographics and online behavior, how likely are they to click on one of LPDs ads?" If you can answer that, you figure, LPD can prioritize buying the ad spots for the types of users most likely to click on their ads.

To develop that model, you use your budget to run your ads on different sites and at different times. You then use that data (and those glorious cloud computing resources) to train a machine learning model that predicts whether someone will click on one of your ad based on the user's demographics and ad placement. You try out a few different models, tune the model parameters, and eventually settle on a neural network model with extremely high precision *and* recall. Hooray!

LPD uses the model to target users likely to click their ads, and almost immediately the CTR of their ads increases 5-fold! Not only that, but the share of people who click on ads that go on to buy a pizza has also increased. Everyone congratulates you, and you move on to the next project feeling very smug.

A few months later, though, you are called into a meeting with the LPD advertising team and the company's Chief Financial Officer. They've been looking over the numbers, and despite the huge rise in CTR, they seem to be getting fewer online orders than before you arrived. CTR rates are up, but somehow it isn't generating greater profits.

Can you figure out what went wrong?

OK, this is the place in most books where the authors ask you that question, and you look up at the ceiling for a minute, shrug, and then read on.

But I'm really, *really* serious about this: close your laptop, stand up, set a 5-minute timer on your phone, and go for a walk. Ponder this example. See if you can figure out what's going on. This is *precisely* the kind of problem you will soon face as a professional data scientist, so why not practice trying to think through the problem?

Solving The Wrong Problem

So what happened?

The reason an increased click rate wasn't making LPD richer is that LPD's problem was never the fact they had a low CTR; LPD's *real* problem was that they weren't getting a lot of orders online. And because Little Papa Domino's problem wasn't a low CTR, being able to answer the question "How likely is a given user to click on an ad" *didn't actually solve their real problem*.

What question, if answered, would have helped solve their problem? "Given a user's demographics and online behavior, *how much more likely are they to buy a pizza* from LPD if we show them an ad?"

Or, expressed more succinctly, LPD *thought* their problem was that their ads weren't *getting clicks,* but really their problem was that their ads weren't *driving increased sales*.

The difference is subtle, but critically important: someone clicking an ad doesn't make Little Papa Dominos any money. Someone clicking an ad *and ordering a pizza* doesn't necessarily make LPD any money. Why? Because they may be someone who would have bought a pizza from LPD anyway, whether you showed them an ad or not. The person who was already thinking of ordering a pizza from LPD is *precisely* the type of person your algorithm may have targeted, and who may have clicked the ad to save themselves a Google search!

But the person LPD *wants* to show an ad to isn't the person who was already thinking of ordering a pizza from LPD, it's the person who was thinking of a pizza but wasn't sure who to order it from, or the person who wanted dinner but didn't know what to get. They may be less likely to click the ad than the person who was about to Google "Little Papa Dominos," but their precisely the type of user who is more likely to buy a pizza from LPD as a result of seeing an ad than they would have been otherwise.

Counter-Factual Advertising

Lest you think this example is contrived, it's not. The realization that the goal of ads isn't to maximize clicks but rather to induce the largest possible change in purchasing behavior is one of the most important ideas in online advertising. It has had a huge impact on how online advertising works, and how people evaluate the success of ad campaigns.

Indeed, this is why companies like Meta and Google are so eager to track user behavior across apps and websites. When Meta and Google can "follow" users after they've clicked an ad, they can evaluate ad performance based not on clicks but on customer behavior. When paired with their ability to show ads to some users and not to others and track both groups as they move around the web, Meta and Google can see whether users who see the ads are more likely to make purchases than those who don't. This allows them to estimate the true effect of ads on sales, data they use to improve ad targeting *and* justify higher prices to advertisers.

Next Up: Types of Questions

Having established the importance of first articulating the problem one seeks to solve, we will shortly turn to developing our understanding of the three types of questions introduced in the first chapter of this book.

First, though, a quick digression into understanding the historical context of data science. This may feel like an odd topic to talk through in a technical data science text, but as we'll see understanding how we got to where we are today is key to successfully navigating modern data science.

Stakeholder Management

stuff

Step 0: Recognize Your Role

If you remember nothing else from this chapter, please remember this: helping your stakeholder better understand their problem is a core part of the job.

Because most stakeholders are older and domain experts in their field, young data scientists tend to err on the side of deference. It is important to be respectful of your stakeholder's experience and to use their domain expertise, but it is important to also recognize that data science is about *pairing*

domain expertise with computational methods and quantitative insights, and neither you nor your stakeholder is likely to have expertise in *both* the substantive domain in question *and* cutting edge quantitative methods. Indeed, if they did, they probably wouldn't be hiring you![1] So don't hesitate to speak up! Ask questions, raise concerns, and while you should do so with *some* humility, have confidence in your own expertise.

Step 1: Don't Assume Your Stakeholder Knows What They Need

A corollary to Step 0 is to not assume your stakeholder understands what they need. So when I say "helping your stakeholder understand their problem is a core part of the job," I don't only mean that it's part of your job *if the stakeholder admits to deep uncertainty about their problem*." Odds are your stakeholder will come to you with a strong statement of what they think they want, but you should take that as a starting point for discussion, not your mandate.

This is particularly true if your stakeholder comes to you with really specific technical suggestions. Often you will be approached by a stakeholder who, rather than laying out a problem, announces they would like you to do X using some data science tool Y. Occasionally the stakeholder doing this knows exactly what they're talking about, and you should use Y to do X.

More often, however, you're dealing with a stakeholder with just enough knowledge to be dangerous (and to drop buzzwords), but not enough to know how best to solve their problem.

Most people ask data scientists for help because they don't know much about data science (or, worse, they *think* they know about data science but don't). Again, different rules apply if you're at Google or Apple, but in most contexts, it's a good idea to treat implementation details provided by the client as a red herring. Focus on the stakeholder's *needs*. Only get into implementation details once you feel you understand the problem well.

Step 3: Ask Questions (Especially Quantitative Ones!)

Focus on the stakeholder's *needs*. Only get into implementation details once you feel you understand the problem well.

Be sure to ask a lot of questions of your stakeholder. In particular, I would suggest two types: questions about what success would look like, and questions about the problem itself.

Questions About Success

Getting a sense of where the goalposts are for your stakeholder will both help you know what to target and also help you better understand your stakeholder's understanding of the problem. Make sure to ask questions like:

- How are you measuring the problem? What would you measure to help you know if you were successful in solving the problem?
- How big, in quantitative terms, is this problem?
- How much would you need the current situation to change to call this a success?

Questions About the Problem

The more you know about your client's needs the better, so ask anything that comes to mind. If the client can answer your question, it will help you better understand the situation; if the client can't answer your question you may find that they are suddenly really interested in knowing the answer, and you immediately have some of your first Exploratory Questions to try to resolve.

In the example of the company that wanted to improve recruitment of high-quality employees in the introduction of this book, I suggested that some of the first exploratory questions you might want to investigate would be things like:

- How many job applications are you receiving when you post a job?
- What share of your current job applicants are of high quality?
- What share of employees you try to hire accept your offer?
- What share of employees you do hire turn out to be successful employees?

These are all questions that I would ask my stakeholder in one of our first meetings.

Note

Stakeholder Meetings It is always good to go into meetings with your stakeholder with a clear sense of your objectives — what you hope to communicate, and what information and feedback you need to get before the meeting ends. When your stakeholder is someone you don't get to meet with regularly, it's good practice to detail these objectives and provide them — in writing — to your stakeholder in advance of your meeting. This will not only ensure that you and your teammates are on the same page (as you will all have reviewed the document before sending it to your stakeholder), but also ensure that your stakeholder has adequete time to reflect on any questions or issues you wish to raise.

When it comes to your *first* meeting, however, this practice can feel impractical as you may feel so uncertain about the project that you only know the first few questions you want to ask.

But even in a first meeting, preparation is key. Rather than laying out the new issues you wish to raise and questions you want answered, for a first meeting it's helpful to write out a full *tree* of lines of inquiry you may wish to propose. In other words, for every question you wish to pose to your stakeholder, try to anticipate some likely responses they make provide, then write down a few followup questions to ask if they provide one of those responses.

Time with your stakeholder is *precious*, especially early in a project, make the most of that face time through preparation.

Step 4: Propose Questions You Might Answer

As a data scientist, answering questions about the world is the instrument you have to solve problems. So once you think you have a sense of your stakeholder's needs, turn around and propose a handful of questions and ask them if answering those questions would help solve their problem.

This is important because many people have only a vague sense of what they are likely to get as a "deliverable" from the data scientist. They usually have a vague sense that they will get some type of magic machine (a "magic model" or "magic algorithm") that will just make their problem go away. By concretely framing your deliverable as the answer to a question (or a model that would answer a

specific question for each entity like a customer or patient that it encounters), you can get much more valuable feedback before you dive into a problem.

Make Your Questions Specific and Actionable

In developing your questions, it is important to make them specific and actionable. A specific and actionable question makes it very clear what you need to do next. For example, suppose an international aid organization told you they were worried that urbanization in Africa, Asia, and Latin America was impacting efforts to reduce infant mortality. Some examples of specific, actionable questions are: "Is infant mortality higher among recent migrants to urban centers, controlling for income?" or "Are the causes of infant mortality among recent migrants to urban centers different from those living in rural areas?" Reading those questions, you can probably immediately think of what data you'd need to collect, and what regressions you'd want to run to generate answers to those questions.

Vague questions would be "Is urbanization impacting efforts to reduce infant mortality?", or "Does urbanization affect infant mortality?" Note that when you read these, they don't seem to obviously imply a way forward.

Perhaps the best way to figure out if your question is answerable is to write down what an answer to your question would look like. Seriously – try it. Can you write down, on a piece of paper, the graph, regression table, or machine learning diagnostic statistics (complete with labels on your axes, names for variables, etc.) that would constitute an answer to your question? If not, it's probably too vague.

Step 5: Iterate

And here's the last but perhaps most important step: **iterate.** Bring your work back to your stakeholder as often as possible.

Many stakeholders find the idea of data science mysterious and abstract and will struggle to understand what is and is not feasible. By bringing them intermediate results, the whole process will start to become more concrete for the stakeholder, and it will help them provide you with better feedback.

The way this book is organized suggests a natural flow from problem articulation to answering Exploratory Questions to prioritize efforts, to answering Passive-Prediction Questions to target individuals for extra attention or automate tasks, and finally to Causal Questions to better understand the effects of that extra attention/automation. In reality, however, a good data scientist is always coming back to the stakeholder, updating their plan, and jumping back in the sequence when new questions arise.

Reading Reflection Questions

- Why should you care if your stakeholder misspecifies their problem?

---

[1] Obviously there are exceptions to this — if you work for a mature tech company like Google or Meta, you may very well end up working under a manager who knows sides of a problem significantly better than you. In my experience, however, is circumstance is the exception, not the rule.

Descriptive v. Prescriptive Questions

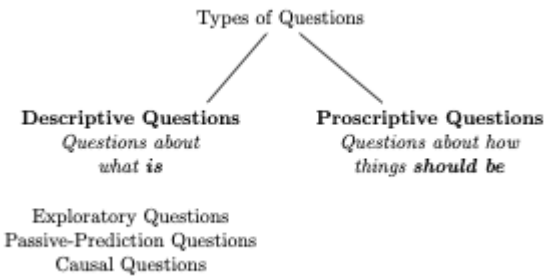
In the chapters that follow, we will discuss Exploratory, Passive-Prediction and Causal in detail. First, though, we must discuss another important concept: the distinction between descriptive and prescriptive questions.

Descriptive Questions are questions about the state of the world and include all the questions and examples we've covered so far in this book. "What kinds of users are clicking our ads?" and "Do high-income and low-income countries emit similar amounts of carbon dioxide?" are examples of Descriptive Questions. And because Descriptive Questions are questions about objective reality,[1] they have right and wrong answers (at least in principle. It may be hard to evaluate whether a given attempt to calculate the answer is actually right or wrong).

But Descriptive Questions are not the only type of question you will come across in your career.

Prescriptive Questions are questions about how the world *should be*, not how it actually is. "Should higher income and lower income countries be expected to meet the same carbon emission reduction standards?" or "Do high-income countries have a moral obligation to provide tuberculosis drugs to developing countries for free (or at cost)?" are both examples of Prescriptive Questions.

Unlike Descriptive Questions, Prescriptive Questions don't have correct answers. That's because answers to Prescriptive Questions require evaluating the *desirability* of possible outcomes, which can only be done in the context of a moral/ethical system of values. And as there is no "correct" system of values (in the sense that there is no single universally accepted system of morality), there can be no right or wrong answers to Prescriptive Questions, even in principle.



> **Note**
>
> The terms "Prescriptive" and "Descriptive" are commonly used for these concepts in the natural sciences, but different academic silos sometimes use different terms. Social scientists, for example, tend to prefer the terms "Positive" in place of Descriptive and "Normative" instead of Prescriptive. These are only difference in nomenclature, however, not substantive meaning.

The focus of this book is on Descriptive Questions. This is not because Prescriptive Questions are unimportant — indeed, one can easily make the argument that they are *more* important than Descriptive Questions. Moreover, as we will discuss in future chapters, they will arise frequently in your career as a data scientist. No, the reason that Descriptive Questions are the focus of this book is that those are the only questions data science tools can answer, and thus answering Descriptive Questions is the domain in which the data scientist has a clear comparative advantage.

Now, to be clear, none of this is to mean that the answers you generate as a data scientist will not have a *bearing* on how people answer Prescriptive Questions. Data science would be a very dull field indeed if it could not speak to the ethical issues of our day. Data science is powerful precisely because it can inform how we answer Prescriptive Questions by helping us understand the relevant stakes. Data science tools can help decision-makers understand the likely *consequences of different courses of action*, information that can help people make *informed* decisions about what outcomes they feel are most desirable. To illustrate, let's consider a few vignettes.

Opioid Reductions

Suppose you have been hired by a medical regulatory board concerned about the rise in opioid overdoses. They are debating whether they *should* (there's that magic word!) make it harder for patients to get opioids. Fundamentally, however, they worry that while restrictions on opioids may reduce overdoses and addiction, they may also prevent some patients with very real pain conditions from getting the care they need.

Why are they stuck? Well, there may be two causes:

1. They may be unsure of the relative moral weight to give preventing overdoses versus ensuring appropriate patient access to opioids, and/or
2. they may also be unsure about *how much* opioid regulations that reduce overdoses by a certain amount would limit access for patients in need.

The first of these questions is a pure Prescriptive Question — if you could prevent one overdose death at the expense of preventing 10 patients in pain from getting the opioids they need, would you accept that trade-off?

But the second is actually a Descriptive Question that you — the data scientist — *can* answer! You could study policies that have been implemented in the past and come up with a rigorous estimate of how much opioid regulations that reduce overdoses also reduce access for patients in need. You could also evaluate different kinds of policies to figure out which is most efficient — maybe some policies (like not allowing any opioid prescriptions at all) are good at stopping overdose deaths but also *really* limit appropriate access, while other policies are similarly good at reducing overdoses but have a much smaller effect on limiting access.

The Example of Carbon Emissions

A profoundly difficult Prescriptive Question in debates over carbon reduction is whether developing countries should be held to the same emission reduction targets as more developed countries. On the one hand, developing countries like China and India are the source of most current growth in carbon emissions, and so policies that do not apply to developing countries are unlikely to prevent many of the worst climate change outcomes. On the other hand, these countries produce radically

less carbon *per capita* than Europe or the United States, and the industrial growth creating those emissions has been a major factor in lifting billions of people out of extreme poverty.

Hard choices indeed! How does one weigh the improvements in the quality of life of those in extreme poverty against the possible consequences of even greater climate catastrophes?

While that question, in part, is a Prescriptive Question that no regression can answer, data scientists *can* bring data to bear on this question indirectly by helping everyone understand the potential consequences of different carbon targets for developing countries, as well as the feasibility of different strategies for carbon reduction. A data scientist could, for example:

- Evaluate the effectiveness of different messages politicians in the US and Europe could use to convince their constituents to support greater carbon reduction targets,
- Quantify the magnitude of the effect on global warming caused by different emissions targets for developing countries to help politicians in developing countries weigh the poverty-reducing benefits of carbon-intensive industrialization against the likely direct effect of flooding, droughts, or more severe storms on their own citizens, or
- Estimate the cost-effectiveness of developed countries sharing lower emissions industrial technologies with developing countries to ameliorate the tradeoff between poverty reduction and emissions.

In each of these cases, the data scientist is only answering Descriptive Questions, but in doing so they are helping everyone better understand the consequences of their decisions, and in doing so (hopefully) help the world to make more informed decisions about the trade-offs they are making.

Recap

Answering Descriptive Questions — questions about how the world is or would be in different scenarios — is the core competency of the data scientist. In the chapters that follow, we will explore in detail three different kinds of Descriptive Questions: Exploratory, Passive-Prediction, and Causal Questions.

While these are the only types of questions that data science tools can answer directly, it is important for you, the data scientist, to also recognize when you encounter Prescriptive Questions — that is, questions about how the world *should* be, or what we *ought* to do. These questions can only be answered with respect to a system of values, and as such, do not have right or wrong answers, and cannot be answered by statistical means. Nevertheless, as a data scientist, you are

well-prepared to help others (and yourself!) make more informed choices when they decide how to answer Prescriptive Questions for themselves.

---

[1] If you know enough epistemology to object to me asserting the existence of an "objective reality," then I assume you can also understand the point I'm trying to get across in this chapter and will forgive me this philosophical slight.

EDA: The Most Pernicious Term in Data Science

In our next reading, we will turn our attention to *Exploratory Questions*. First, however, it is important to have a candid discussion about what I feel is one of the most problematic concepts in data science education: *Exploratory Data Analysis* or *EDA*.

The problem with the term Exploratory Data Analysis is that, if you asked most data scientists what it means, they probably couldn't actually give you a straight answer. If you pressed them further, they would probably say something like "when you look at your data before you start fitting your models."

While the idea that data scientists should "get to know their data" before fitting a model is well-meaning (you *absolutely* should!), the ubiquitous but uncritical use of the term has given young data scientists the sense that the undirected poking at data is worthy of a capitalized three world title, complete with a universally recognized acronym.

This is problematic because *any* activity that involves data but lacks a clear motivation is doomed to be unending and unproductive. Data science has emerged precisely because our datasets are far too complex for us to understand directly; indeed, I would argue that the job of a data scientist can be summed up, in part, as a person who identifies **meaningful** patterns in our data and makes them comprehensible.

But therein lies the problem — without a clear motivation for *why* the data scientist is poking at their data, what makes a pattern meaningful is undefined. And without a clear purpose from which a concept of meaningfulness can be derived, there is no end to the ways one can slice and dice the data with no way of knowing when to stop or what is useful.

I would argue that what most people call Exploratory Data Analysis (EDA) can actually be decomposed into three activities.

The first activity people call EDA is what I call "learning the structure of your *dataset*" (emphasis on learning about your *dataset*, not using your data to learn about the world). This consists of answering questions about your dataset like "what constitutes a single observation in this dataset?," "what variables are included in this dataset?," "how many observations are there?," "how are variables coded?," and "what population is represented in this data?" These are questions about *the specific dataset* you are working with, *not* the real world, and answers are likely to be found in the dataset documentation and through basic tools for data introspection.[1]

The second activity that often falls under the label EDA is what I call "validating your dataset." It's a poor data scientist who takes the validity of their data on blind faith, so when faced with a new dataset, one should begin with a few "sanity checks" just to make sure things look reasonable. Does the number of observations seem reasonable given what you know about how the data was collected and who is supposed to be represented in the data? If there are date variables in the data, does their range match what should be in this data? And given the specifics of the data, does the range of variables make sense? For example, if you have data on registered voters 18 and over, you should probably check that the age variable has a minimum value of 18 and a maximum value of something sensible (e.g., not 225).

The third and final activity people call EDA is... everything one does with the data before they fit a statistical or machine learning model. This is the second major reason that I feel the very concept of EDA has had a pernicious influence on data science — it implicitly devalues anything done with data that doesn't entail a complicated model as "lesser" or "just a stop on the way towards the "real" analysis," when nothing could be further from the truth.

This type of data analysis — looking at summary statistics, calculating distributions of variables, computing tabulations and cross-tabulations of different things to improve one's understanding of the world — is categorically different from "learning the structure of your data," because it is inquiry in the service of better understanding the world, not the structure of your dataset. But it is *not* categorically different analyzing data using statistical models, not just because in many cases generating cross-tabulations or calculating group averages are essentially equivalent to using a statistical method like linear regression, but also because they are both examples of the same enterprise: attempting to answer questions about the world using data in the service of solving problems.

And just as one cannot properly fit or tune a model without a clear sense of the question one is seeking to answer and how that answer is meant to be used, nor can one know what cross-

tabulations to compute without having a sense of purpose to make clear what constitutes "meaningfulness."

> **Note**
>
> "But I do EDA all the time without a clear question!" I hear you cry. "Sometimes I just want to see what patterns there are in the data."
>
> To you I say: you may not have realized you had questions in mind, but most of your data explorations have been *implicitly* motivated by a sense of questions you thought might relate to your stakeholder's problem.
>
> Perhaps you were looking at a store's retail sales data and decided to see how sales volumes varied by customer age or gender. That may not seem obviously question-motivated, but I put it to you that you had in mind that those are customer demographics to which the store could target advertising or product stocking decisions. And had someone suggested "why don't you look at how sales volumes vary by customer birth month or whether their name starts with a letter in the first half of the alphabet," you would have looked at them funny and said "why on Earth would I do that?"
>
> But the problem with approaching your data with *implicit* motivations is that (a) it's hard to reflect on them or evaluate whether they rest on solid assumptions about the stakeholder problem, and (b) without an explicit goal, there's no way to know when you've reached your destination, making it *really* easy to get lost in the data.

Recap

Despite its ubiquity, few data scientists could actually tell you what constitutes Exploratory Data Analysis (EDA). Moreover, some of what people might call EDA in practice — answering questions about the world without complex modeling — should not be called EDA, but rather... well, that's just data science.

So in this book, we will acknowledge the important (but distinct!) goal of two purposeful activities often called EDA:

- Learning the structure of your dataset (what constitutes a unit of observation, what variables are in the dataset),

- Validating your dataset (does the data pass the sniff test? Does it exhibit the basic properties you would expect given what it claims to be?)

But I will *not* use the term EDA itself, and when I differentiate between data science enterprises, I will do so by emphasizing differences in the *end goals* of those activities (answering Exploratory Questions, Passive-Prediction Questions, or Causal Questions), not the methods used to achieve those ends.

---

[1] In `pandas`, this would be things like `df.columns` to see what variables are in the data, `df.info()` to get a sense of how data is being represented and the number of rows, and simple tools for tabulating unique values like `df["first column"].value_counts()`.

Using Exploratory Questions

The hardest part of a data science project is often properly articulating the problem we wish to solve. That's because properly specifying a problem requires *understanding* the problem well enough to state it, and often we call issues "problems" precisely because we don't really understand them!

Enter *Exploratory Questions*. Exploratory Questions are questions designed to elicit information about our problem space and aid us in prioritizing our efforts and refining our goals. Exploratory Questions are questions about broader patterns in the world. In their simplest form, they can be answered by simple summary statistics or plots. When more complicated or related to more subtle and contingent patterns, they are likely to be answered through unsupervised machine learning algorithms or the tools of *statistical inference*, such as regressions and generalized linear models. When used for answering Exploratory Questions, the emphasis of regression or generalized linear models is on what the model coefficients can tell us about how different factors may co-vary in the world. This is distinct from how these same tools may be used to answer Passive-Prediction Questions, however, where the emphasis is on the predicted values these models can generate.

Of the three classes of questions we detail in this book, answering Exploratory Questions often (though not always) requires the least technical sophistication, and as a result, Exploratory Questions often get the least respect. But because of their critical role in improving our understanding of our objectives, learning to ask and answer Exploratory Questions will have a huge influence on your effectiveness as a data scientist.

In this reading, we will discuss how to *use* Exploratory Questions to guide your work. Then, in the next reading, we will discuss the challenges inherent to *answering* Exploratory Questions.

Using Exploratory Questions to Prioritize Efforts

Exploratory Questions are questions about the underlying patterns that characterize our world. Given that, answers to Exploratory Questions should make you feel like you understand the contours of the problem you seek to solve better. More than anything else, then, Exploratory Questions help data scientists prioritize their subsequent efforts and investigations.

Code Optimization

If this feels too abstract, let's use a small example of a problem you've probably already come across (and if not, probably should have!) in the classroom to make it more concrete: learning to write performant (i.e., fast) code.

Data science is full of computationally intensive tasks that, if approached incorrectly, can leave a data scientist staring at their computer for hours, days, or even weeks (if they allow it). As a result, most data scientists will go through a phase in their development when they start constantly worrying about how to make every line of code they write as fast as possible. They bend over backward to write unnatural, unreadable code to ensure that they aren't wasting a single CPU clock cycle.

The problem with this is that humans have *incredibly* bad intuition about what tasks take a computer a long time. It turns out that even in programs that take huge amounts of time to run, it is often the case that *most* of the program's runtime is taken up by a single function or loop. As a result, programmers who fixate on ensuring every line of code they write is optimized for speed end up not only wasting their *own* time, but also writing code that is less natural, harder to maintain, and more likely to contain errors for effectively no benefit.

Indeed, no less a figure than Donald Knuth, one of the greatest programmers in history and author of the famous *The Art of Computer Programming*, famously said of this trying to optimize each line of code at the time it is being written ("premature optimization"):

> The real problem is that programmers have spent far too much time worrying about efficiency in the wrong places and at the wrong times; **premature optimization is the root of all evil (or**

> **at least most of it) in programming.** [emphasis added]

So what is a programmer interested in performance to do? First, write code in as natural a way as possible. Then, *if* the result is code that is slower than they would like, ask the exploratory question: "What lines of code are contributing most to this program taking so long to run?" And only then, once the programmer has identified the problematic parts of their code, optimize it for performance.

How is this question answered accomplished? Programmers use tools called *profilers* that dip into a running program every few milliseconds to see what functions are currently running. Then after the program has finished running, it reports how often each part of the program code was found to be running, giving the user a sense of the overall distribution of time spent running different parts of the code.

Picking the Right Target

If the preceding example feels too niche — you want to be a data scientist, after all, not a software engineer! — let's consider a different example. Suppose you've been hired by a new non-profit interested in helping reduce energy use in buildings in the United States. They know that fixed structures (factors, stores, houses, etc.) are responsible for a huge share of US energy consumption, and are interested in figuring out how to drive down that energy use by helping building owners improve the energy efficiency of their buildings (by providing information on things like government subsidies for efficiency improvements and the potential value of energy efficient windows, better heating and cooling, etc.).

You *could* start out by trying to build a fancy supervised machine learning model that tried to predict the energy use of every building in the US based on infrared satellite data and weather information. Indeed, that may even be what you were asked to do! (See our discussion of how stakeholders will often have somewhat wild ideas of what is feasible and what would help most.).

But given this is a new non-profit, it sounds like their real need is probably to figure out how to target their efforts to be most effective. So maybe we should step back and start by trying to answer a few Exploratory Questions that would help the organization decide where to focus its attention:

- What *type* of buildings (industrial, residential, commercial) consume the most power in the US?

    - The answer to this question can help you prioritize the *types* of buildings on which to focus your efforts. For example, if industrial or commercial buildings only represent a few percent of all energy consumed by buildings, you don't need to worry about addressing their needs!
- In what *region* of the US are buildings consuming the most power?
    - If most energy is being consumed in a specific area, perhaps the non-profit should start by focusing its efforts regionally.
- Is there a *region* of the US where buildings are generating the most CO2?
    - Not all power is created equal when it comes to climate change! Maybe buildings in California consume a lot of energy, but because they have cleaner power plants, those buildings are indirectly generating less CO2 than those in states in the US South?
- Does the *average energy use per building* vary by region or building type?
    - If the non-profit plans to approach building owners, it may be easier to have an impact working with a few owners of large buildings than lots of residential homeowners. But of course, that also depends on the answer to our previous question about what types of buildings are using the most power/generating the most CO2!
- In what season is most building energy consumed? Is more energy consumed by heating or AC needs, or do the two use similar amounts of power?
    - Again, this may impact both the regions the non-profit may wish to focus on, and also the types of efficiency retrofits they may wish to prioritize.
- Where is power most expensive?
    - Building owners are most likely to be interested in efficiency retrofits when power is expensive.

While answering these questions is likely to require some significant detective work, and may require some thoughtful data wrangling, none require deeply sophisticated statistical machinery. But that doesn't mean answering these questions wouldn't provide **huge** value to the stakeholder.

Collecting, Merging, and Creating New Data

Once you start articulating these questions, you can start to see that there is some important data science to do; that's because the answers to these questions may not all point in the same direction, and so the non-profit likely needs someone to be able to evaluate how these different factors co-vary, and the relative magnitude of different trade-offs (e.g., if fewer buildings use a lot

of power in the US South than California, but the US South is using coal power instead of renewable energy, where should the non-profit focus?).

And that, fundamentally, is what Exploratory Questions are about: understanding the patterns and distribution of features you care about in the world, and using that information to better understand the problem you want to solve.

This also demonstrates one of the key ways that one answers Exploratory Questions: by collecting and merging datasets that had not previously been pulled together. Sometimes this data collection requires no more than finding people who already have the data you need, getting it, and finding a way to merge different data sources (e.g., data on power plant CO2 emissions and data on building energy use), while in other situations this will entail building new datasets yourself by doing things like using Natural Language Processing to make collections of documents (contracts, patient files, public records) analyzable systematically.

But Where Do I Get Data?

It's hard to overstate how much data is public these days, but to give you a sense, here's a quick summary of a few terrific sources.

Answering Exploratory Questions

In the last reading, we discussed how Exploratory Questions are used by data scientists to help stakeholders better understand their problems and to prioritize subsequent investigations. In this reading, we turn to the questions of what *answering* Exploratory Questions effectively entails.

The Three-Part Goal

Whether one uses simple summary statistics (means and medians), plots, or more sophisticated algorithms from the domains of statistical inference and unsupervised machine learning, answering Exploratory Questions always boils down to the same challenge:

Creating (1) understandable summarizations (2) of meaningful patterns in the data, (3) and ensuring they are faithful representations of the data.

What is meant by these three components exactly? Let's take each in turn.

Understandable Summarizations

> Creating **(1) understandable summarizations** (2) of meaningful patterns in the data, (3) and ensuring they are faithful representations of the data.

Answering Exploratory Questions effectively is all about taking large datasets that, in their raw form, are effectively incomprehensible to humans and summarizing the patterns in that data in a way that can be understood. These summaries of patterns in the data may take many forms — summary statistics, regression coefficients, plots, etc. — but all, when done well, have a similar goal: to represent the salient aspects of data in a way that is accessible to the human mind.

Professionals from different disciplines often use different terminology to describe this process of summarization. Some like to refer to it as "separating the signal (the thing that's important) from the noise (all the other variation that doesn't matter)," others talk about "dimensionality reduction" (basically linear algebra speak for summarization), while still others may talk about "modeling the underlying data generating process that gave rise to the observed data." Regardless of the terminology one uses, however, these all boil down to the same thing: filtering and discarding the variation the data scientist deems to be irrelevant to make it easier to see and understand the variation deemed important.

The importance of researcher discretion in deciding what variation to discard as noise and what variation to foreground as "important" is one of the defining challenges of answering Exploratory Questions. Other types of questions — like Passive Prediction Questions — often involve using more mathematically sophisticated modeling tools, and consequently are viewed as more challenging. In my experience, however, learning to understand the stakeholder's problem context *and* the variation in a data set well enough to exercise this discretion effectively is actually one of the things young data scientists struggle with most. It requires both good domain knowledge to understand what is *meaningful* (as we will discuss below), and also for the data scientist to spend a lot of time exploring the data thoughtfully and from different perspectives. This is a hard skill to learn,[1] but with intentionality, patience, and practice, it is a talent that once learned will helps set you apart from the average pytorch-jockey.

Summarizations created to answer Exploratory Questions can differ radically in their ambition. At one end of the spectrum are simple summary statistics, like means, median, and standard deviations. These seek to provide a simple characterization of a single feature of a single variable.

Slightly more ambitious are various forms of plots — like histograms (which are substantially richer than the aforementioned summary statistics) or scatter plots and heatmaps (which provide substantial granularity and communicate information about the relationship between different variables). The most ambitious efforts make use of multivariate regressions and unsupervised machine learning algorithms to model what they call the *Data Generating Process* (DGP) — the actual physical or social processes that gave rise to the data you observe, and which (hopefully) can be represented in a relatively parsimonious manner, much as the relatively simple laws of physics give rise to the orbits of the planets and the complexity of life.

To illustrate what I mean by trying to deduce something about the data-generating process, suppose you are a medical researcher interested in a poorly understood disease like Chronic Fatigue Syndrome (CFS). It is generally agreed that CFS is more of a label for a constellation of symptoms than an understood physical ailment, and you have a hypothesis that the symptoms of CFS aren't actually caused by a single biological dysfunction, but rather that multiple distinct biological dysfunctions give rise to similar symptoms that we have mistakenly grouped under this same umbrella term. In other words, you think that the data-generating process that gives rise to patients diagnosed with Chronic Fatigue Syndrome consists of two distinct diseases.

You're fortunate enough to have detailed patient data on people diagnosed with the condition, but it's impossible for you to just look at these gigabytes of thousands of patient records and "see" any meaningful patterns. You need a way to filter out irrelevant data to identify the "signal" of these two conditions. To aid you in this question, you decide to ask "If you were to group patients into two groups so that the patients in each cluster looked as similar as possible, but patients in different clusters looked as *dissimilar* as possible, how would you group these patients?"

This, you may recognize, is precisely the question clustering algorithms (a kind of unsupervised machine learning algorithm) are designed to answer! So you apply your clustering algorithm to the patient data and get back a partition of the patients into two distinct groups. This, in and of itself, doesn't constitute a particularly *understandable* summarization of your data, but it provides a starting point for trying to investigate *diagnostically and biologically relevant* differences that exist between these populations. If one cluster included more patients reporting fatigue when doing any exercise, while another cluster reported they felt better when they exercised, but felt a high level of baseline fatigue that didn't respond to sleep, that might suggest that the *data-generating process* for these patients was actually driven by two different biological processes. *And* it gives you a great starting point to prioritize your subsequent investigations into what might explain these differences!

Meaningful Patterns

> Creating (1) understandable summarizations **(2) of meaningful patterns in the data,** (3) and ensuring they are faithful representations of the data.

Inherent in creating any summarization is exercising discretion over what variation is relevant (signal) and what variation is not (noise). But just as one person's trash may be another person's treasure, so too may one person's signal be another person's noise, depending on their goals! Crucially, then, the data scientists' guiding star when deciding what is important is whether certain variation in the data is *meaningful to the stakeholder's problem*.

As data scientists, we are blessed with an abundance of tools for characterizing different facets of our data. These range from the simple — means, standard deviations, and scatter plots — to the profoundly sophisticated, like clustering algorithms, principal component analyses, and semi-parametric generalized additive models.

Regardless of the specific methods being employed, however, none of these tools can really tell us whether the patterns they identify are meaningful, and that's because what constitutes a meaningful pattern depends on the problem the stakeholder is seeking to address and the context in which they're operating.

To illustrate the importance of context, suppose you are hired by a hospital to learn what can be done to reduce antibiotic-resistant infections. So you grab data on the various bacteria that had been infecting patients and write a web scraper and Natural Language Processing pipeline to systematically summarize all available research on the cause of these antibiotic-resistant bacteria. Your work is *amazing*, seriously top of the line, and after two months you conclude that in most cases, the cause of antibiotic resistance in the bacteria infecting patients is… the use of antibiotics in livestock.

Now, that analysis may not be *wrong* — you have properly characterized a pattern in the data — but it isn't a pattern that's meaningful to your stakeholder, who has no ability to regulate the livestock industry. That pattern might be meaningful to someone else — like a government regulator — but in this context, with this stakeholder, it just isn't helpful. The features of the data that are important, in other words, depend on what we may be able to do in response to what we learn. And there's no

summary statistic, information criterion, or divergence metric that can evaluate whether a pattern of this type is *meaningful*.

Faithful Representations

> Creating (1) understandable summarizations (2) of meaningful patterns in the data, **(3) and ensuring they are faithful representations of the data.**

What do you means, medians, standard deviations, linear regressions, logistic regressions, generalized additive models (GAMs), singular value decomposition (SVD), principal component analyses (PCAs), clustering algorithms, and anomaly detection algorithms all have in common?

Answer: unless your dataset is extremely degenerate, you can point *any* of these tools at your data and they will return a relatively easy-to-understand characterization of the structure of your data.

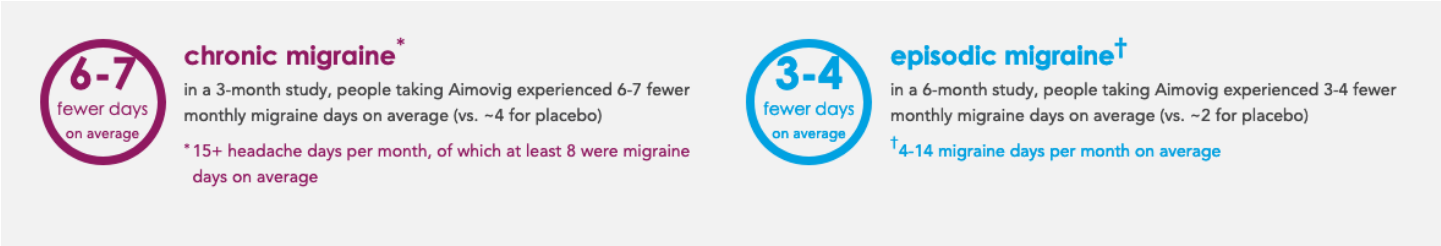
At first, that may seem extremely exciting. But if you think about it a little longer you will realize the problem: all of these are designed to give you a relatively understandable summary of radically different properties of your data, and even though they will all provide you with a result, these results can't all possibly be faithful representations of the dominant patterns in your data.

To illustrate the point, suppose I told you that in one university math course, the average grade was a B-. You might infer that students were doing pretty well! But now suppose I told you that in a different university math course, 20% of the students had gotten a 0 on the midterm and on the final—you would probably infer something was going seriously wrong in that class. And yet those two statistics could both be true of the same class—the only difference is what patterns in the data *I*, the data scientist, have decided are meaningful to communicate to you, the reader.

The example of the math class in which the average grade was a B- and 20% of the students were failing also illustrates one of the great dangers of tools for data summarization: they are so eager to please, they will *always* provide you with an answer, whether that answer is meaningful or not. I think most readers would agree that learning that the average grade in the class was a B- actually misleads more than it informs (since for the class to have an average grade of 80% and a 20% fail rate, the grade distribution would need to be something like 20% 0's and 80% 100's). Indeed, it's worth emphasizing that while hearing "the average grade is a B-" makes the reader think that most kids are doing ok-ish, the reality is that *no one* in the class is doing ok-ish! They're either doing horribly or terrifically!

Less that feel like a contrived example, consider the case of Aimovig, a drug authorized by the FDA in 2018 for treating chronic migraines that was heralded as a "game changer."

To get Aimovig authorized, the pharmaceutical companies developing (Amgen and Novartis) had to run a clinical trial in which a random sample of people with chronic migraines was given Aimovig (the treatment group) and a random sample was a placebo (the control group). Patients in the clinical trial self-reported how their migraine frequency changed when in the trial, and the effectiveness of Aimovig was then evaluated by comparing the decrease in self-reported migraines for those taking Aimovig (on average, a decrease of 6-7 migraines a month) to the decrease in self-reported migraines for those taking a placebo (on average, a decrease of 4 migraines a month).[2] This difference of 2-3 migraines a month — called the "Average Treatment Effect" of the trial — was found to be positive and statistically significant, and so the drug was authorized. Indeed, if you see an ad for Aimovig, you'll probably see the average effect of the drug reported in the same way:



That's great! Chronic migraines can be a crippling disability, so any improvement in treatment is exciting. But you would be excused for asking why people were getting *so* excited about what seems like a relatively small reduction in migraines.

The answer, as it turns out, is that almost nobody experiences this "average effect." Instead, *most* people who take Aimovig see little to no benefit, but *some* (depending on your criteria, something like 40%) see their migraine frequency fall by 50% or more. Amgen and Novartis don't yet know how to identify who will benefit and who will not before they try the drug, and we don't allow drug companies to "move the goalposts" after a clinical trial has already started by changing the way they plan to measure the effectiveness of a drug (for fear they will hunt through the data till they find a spurious correlation that makes it look like the drug works when it really doesn't), so this average effect remains the only statistic that Amgen and Novartis are allowed to report in their advertising.

But if you're a *doctor* or a *patient*, it seems clear that this simple average effect — a reduction of 2-3 migraines a month — really does not provide a *faithful* summary of the underlying variation.

But... I Thought Unsupervised Machine Learning Always Found The "Best"

"Fine," I hear you say, "that makes sense for simple summary statistics. Those are computed by simple formulas. But what about unsupervised machine learning algorithms or generalized additive models? Those use numerical optimization to find the *best* answer!"

Well... yes and no. As you may recall, in the first chapter of the book I posited that all data science algorithms are just fancy tools for answering questions, and even the most sophisticated unsupervised machine learning algorithms are no exception. While it is true that the machinery that underlies these algorithms is much more sophisticated than the formula we use for calculating a variable's average, it is important to not attribute too much intelligence to these tools.

Underlying any unsupervised machine learning algorithm is a simple formula that takes as input whatever parameters the algorithm gets to choose (be those factor loads in a PCA model, or the assignment of observations to clusters in a clustering algorithm) and returns as output a single number. Often this number is called "loss," and the function is called a "loss function," but occasionally different terminology will be used.

One way to think of the job of an unsupervised machine learning algorithm is to pick the parameter values that minimize this loss function. A clustering algorithm for example, may try and assign observations to clusters to maximize the similarity of observations within a cluster (say, by minimizing the sum of squared differences between the values of certain variables for all observations within a cluster) while also maximizing the differences between observations in different clusters (say, by maximizing the sum of squared differences between the values of certain variables for all observations *not* in the same cluster).

But another way to say that is that the job of an unsupervised machine learning algorithm (or any algorithm, really) is to find the parameter values (coefficients in a regression, observation assignments for a clustering algorithm) that answer the question "If my goal is to minimize [whatever the loss function your specific algorithm seeks to minimize], how should I do it?" But while they are likely to find the best way to accomplish that goal given the parameters they control, they will do so *whether or not the "best" solution is actually a "good" solution!* Point a clustering algorithm at any data and ask it to split the data into 3 clusters, and it will pick the best way to split the data into three clusters, even if the three clusters are *almost* indistinguishable. In other words, clustering algorithms assign observations to clusters... even when there's no real clustering of the

data! Dimensionality reduction algorithms will always tell you a way to drop dimensions, and anomaly detection algorithms will always find (relative) outliers.

Moreover, just because your clustering algorithm finds what it thinks is the best solution doesn't mean there isn't a *substantively* very different solution that was *just* a little less good it hasn't told you about.

It's up to you, the data scientist, to evaluate whether the answers these algorithms provide to relatively myopic questions give a meaningful picture of the data.

Myopic Tools

This last point is illustrative of a more general point: data science tools are incredibly powerful at finding answers to questions of the form "If my goal is to minimize X, how should I do it?" type questions — answers you may have never figured out in millions of years! — but their power lies in figuring out the best way to accomplish an articulated goal, *not* in figuring out what goal to pursue.

This is true at both the macro level (doesn't make sense to look for clusters in my data?) and also at the micro level (when assigning observations to clusters, how do I measure success?). Hidden inside nearly all algorithms you use are a handful of baked-in choices you may not even realize are being made for you. Take clustering, for example. In general, when clustering observations, one has two objectives: maximize the similarity of observations within each cluster and maximize the *dis*similarity of observations in different clusters. But what you might not have thought about very much is that there's an inherent tension between these two objectives — after all, the best way to maximize the similarity of observations within each cluster is to only assign observations to the same cluster if they are identical (a choice that creates lots and lots of very small clusters). And the best way to maximize *dis*similarity between clusters is to only put *really really* different observations in different clusters (resulting in a few really big clusters). So how is your clustering algorithm balancing these two considerations? Is the algorithm's choice of how to balance them in any way a reflection of the balance that makes the most sense in the context of your stakeholder's problem? (I'll give you a hint — the algorithm sure can't answer that question, so you'd better be able to!)

Discretion: it's everywhere, and you're exercising it, whether you realize it or not.

---

[1] Although I am far from convinced that the discipline has tried particularly hard to teach it (see my screed against "EDA").

[2] A placebo is a "fake" treatment given to patients in clinical trials. Despite not being biologically active — placebos are often simple saline or sugar pills — most patients on placebos see their condition improve when dealing with subjective conditions, like pain.

Internal versus External Validity

It is at this point I have to come clean about having employed a... small indirection. At the top of this reading, I introduced the idea that answering Exploratory Questions boiled down to creating (1) understandable summarizations (2) of meaningful patterns, and (3) ensuring those summaries faithful represent the data. But that three-part objective is actually only one-half of answering an Exploratory Question. More specifically, those are the three components of ensuring high *internal validity* when answering an Exploratory Question. But to generate a truly useful answer to an Exploratory Question, your analysis must also have high *external validity*.

Essentially, *internal validity* is a measure of how well you have analyzed *the data you have*, while *external validity* is how well you expect the answer you generated from that data to generalize to your stakeholder's context. Internal and external validity arise when answering *any* data science question, and so these two concepts are ones that we will return to time and again in this book.

While the idea of internal validity will feel familiar to anyone who has taken a statistics of machine learning course, external validity is often less discussed in the classroom. This is not because it is less *important* than internal validity. One reason for this is that it is harder to evaluate external validity using statistical methods, causing some instructors to think of it as "outside the scope" of a statistics course. But the second reason I suspect this occurs is more subtle: while a given analysis may be said to have good or bad internal validity, it cannot be said to have good or bad external validity. Rather, the external validity of a study must always be evaluated *with respect to a specific context to which one wishes to generalize the findings*.

This means that external validity is different from internal validity in an important way: when faced with the same facts about a study, everyone should *generally* agree on the internal validity of a study, but the external validity of a study really depends on how you want to use the results. A study of medical care provided to Duke undergraduates may have very *good* external validity with respect to undergraduate students at Emory, UNC, Vanderbilt, and other elite universities with associated research hospitals (that is, the conclusion of the study of Duke students is likely to also be valid for those other students). But that same study may have *poor* external validity with respect to lower income students attending community colleges that have less comprehensive student

health insurance and no top-tier associated hospital system. And it would certainly have terrible external validity with respect to all Americans, never mind people living in other countries.

Interval v. External Validity: An Example

To illustrate what is meant by these terms, suppose you've been hired by a car manufacturer. Through extensive market research, they have determined that if they can improve the safety of their cars, they could dramatically improve the image of their brand. They have turned to you, their in-house data scientist, to help them determine what safety enhancements they should focus on developing.

To help them prioritize their efforts, you decide it would be useful to begin by better understanding the predominant causes of major accidents. After all, if 95% of accidents were caused by mechanical failures, bad weather, and drunk drivers, then even a perfect system for preventing drowsy driving accidents could only reduce accidents by 5% at best![1]

The best source of accident investigations of which you are aware is the US National Highway Traffic Safety Administration's National Center for Statistics and Analysis (NCSA). Using their *Crash Reporting Sampling System*, the NCSA collects and publishes data from "nationally representative sample of police-reported traffic crashes, which estimates the number of police-reported injury crashes and property-damage-only crashes in the United States," as well as data on all fatal accidents collected through their *Fatality Analysis Reporting System*.

The NCSA provides you with data on police-reported accidents from the Crash Reporting Sampling System and all fatal accidents from the Fatality Analysis Reporting System from the past 5 years (2018-2023). As you turn to using this data to analyze the causes of severe accidents for your stakeholder, what internal and external validity concerns might you have in the back of your mind?

Internal Validity Concerns

Internal validity concerns, in this context, relate to your ability to properly characterize the causes of severe accidents in this data. The first of these concerns are those discussed in our previous two readings — namely, whether the summarizations of accident causes you generate from this data are meaningful and faithful representations of the patterns in the data.

But internal validity concerns also extend to things like concerns over the accuracy with which things are measured. For example, it's reasonable to ask "How well do police accident reports capture the true causes of an accident?" Some factors — like whether a driver was intoxicated — are easy to verify after the fact in major accidents. Other factors, however, may have contributed to accidents — and be easier to address with driver assistance systems — but may have been too hard to verify for the police to put in their reports. For example, maybe weak headlights prevented the driver from seeing a change in the speed limit (causing the documented cause of the accident: speeding). Or perhaps one driver was unable to see an approaching vehicle on a cross street due to width of the A-piller (that piece of metal running vertically between the front windshield and front side window). Or "weather" may be invoked as a cause in an accident when what was really at play was that the driver had the wipers on too low of a setting (and automatic wipers would have adjusted more quickly, preventing the accident).

All of these are question about whether our summarization of the **data** provides a proper characterization of **the world** for the period and group we think are covered.

External Validity Concerns

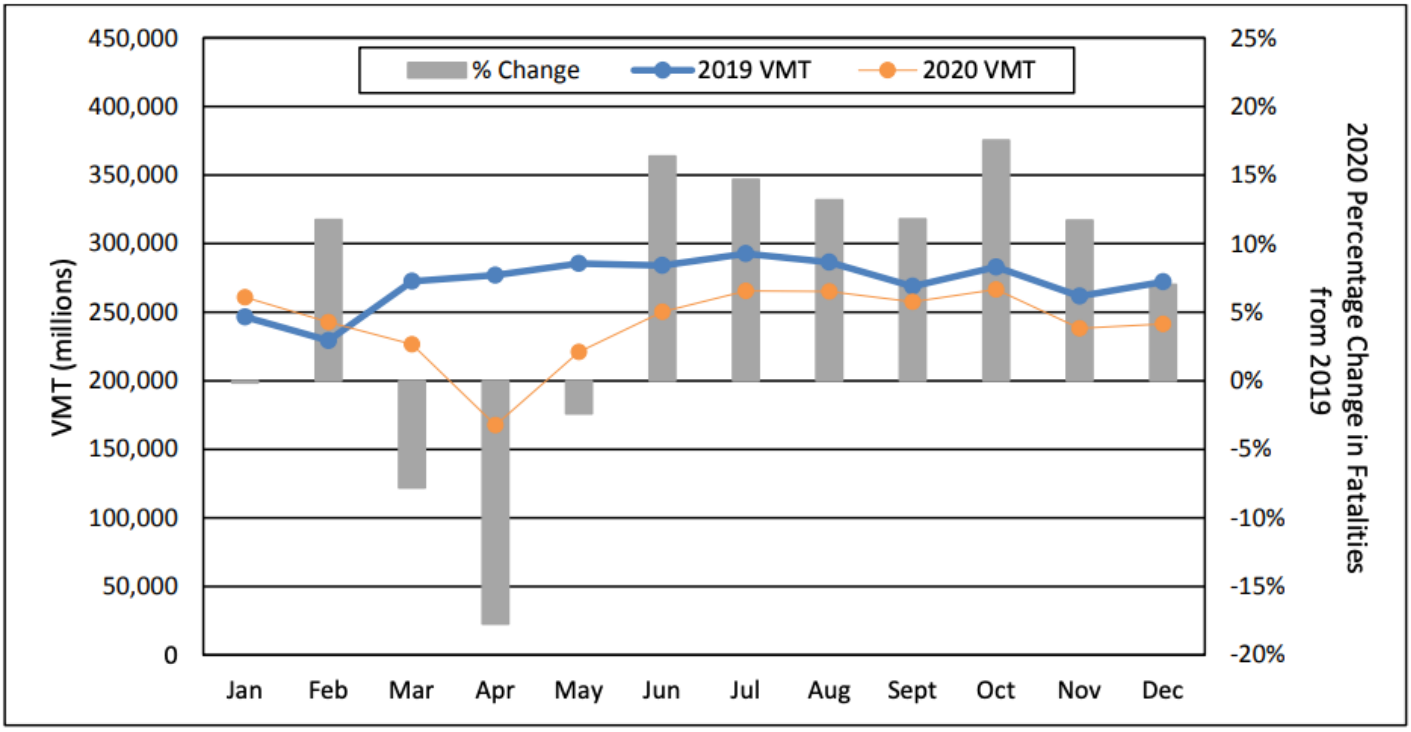
But worrying about whether we really understand the world covered by our data is only half the battle. After all, our stakeholders aren't planning to go back in time 5 years and add a new driver safety system to their cars starting in 2018; they're thinking of developing this safety system to use *in the future*. So what do we think is the *external validity* of findings from 2018-2023 to the next ten years?

To answer this question, we need to consider how the next ten years might differ from the last five *in ways that are relevant to the phenomenon we care about* (here, the causes of car accidents).

First, since time will have passed between when the crash data was collected and when the new feature rolls out, one might reasonably assume that trends in accident causes during the 2018-2023 period are likely to continue. In light of that, a thoughtful data scientist may wish to pay extra attention to whether there are clear *trends* in accident causes, and in what direction they are trending.

But one might also pause to ask whether there was anything exceptional about the "data generating process" during the 2018-2023 period that would not obtain in later years. Something like... a pandemic?

And indeed, one would be right to worry. The figure below shows Vehicle Miles Travelled (VMT) and Fatalities in 2019 (pre-pandemic) and 2020 (the pandemic began in force in March, if you've forgotten). As the figure shows, driving plummeted during this period, and so too did Fatalities.



Sources: FARS 2019 Final File, 2020 ARF; 2019 Monthly VMT – FHWA's December 2020 TVT; 2020 Monthly VMT – FHWA's December 2021 TVT

Figure 1. VMT and Percentage Change in Fatalities, by Month, 2019 and 2020

Source: Taken from the US Department of Transportation NHTSA's *Overview of Motor Vehicle Crashes in 2020*

Given that, a thoughtful data scientist may wish to be sure that any patterns identified in this five-year period are robust to exclusion of 2020-2022 before making any predictions about the likelihood these patterns would persist in later years.

Other Contexts

The preceding discussion assumed an interest in US accidents, but of course most car companies that sell cars in the US also sell cars in Asia and Europe. The issues raised above are examples of what I would contend are relatively *small* threats to the external validity of findings based on data from 2018-2023 with respect to near-future US accidents (at least provided the patterns aren't driven by the pandemic period).

But external validity to Asian or European markets would be a much bigger concern. Because traffic laws, speed limits, alcohol laws, and how roads are constructed and laid out are all very different in different regions of the world, it seems quite unlikely that patterns identified in US accident data would have much external validity to Asian or European auto markets.

Conclusion

Internal and External Validity are both key concepts for being an effective data scientist, and they are concepts to which we will return regularly in this book. Moreover, they are goals that are sometimes in tension — the more control one has over a study context, the more likely one is to have good Internal Validity; but that control can often create an artificiality to limits External Validity. Thus Internal and External Validity should not necessarily be thought of as things to try and simultaneously maximize at all costs; rather, they are best thought of as distinct features of any analysis that should always be considered. I would also add that, of the two, I would argue that External Validity — while not more important in and of itself — is the more often overlooked.

Reading Reflection Questions

- Why is the External Validity of a study only defined with respect to a given context?
- A group of Duke doctors wishes to understand exercise behavior in patients who have recently experienced cardiac surgery. To measure exercise behavior, they surveyed their patients post-surgery about their exercise behavior. But one doctor on the team is concerned that people may mis-report their exercise (the patients know they're *supposed* to be exercising, so the doctor is concerned they will report they are exercising even if they are not). Is this an Internal or External Validity concern?
- After these Duke doctors published their results, word of the study reached a doctor in rural Louisiana. Looking at the study, however, she found that the patients treated by the Duke doctors tended to be younger and healthier than the patients she sees. Is this an Internal or External Validity concern?

---

[1] It is worth noting that if the goal of the company is to improve brand image and sales — rather than improve safety as much as possible — then what matters is not what *actually* causes the most accidents, but rather what customer *perceive* causes the most accidents. But let us assume — for the sake of this exercise — that your employer's interest in reducing accidents is sincere. If there is a divergence between customer perceptions and the reality of accident

causes, that could be addressed with additional information embedded in any advertisements of the feature being developed.

Using Passive Prediction Questions

In the past few chapters, we explored the role of Exploratory Questions in helping data scientists better understand the problems they seek to solve and to prioritize subsequent efforts. While useful, however, answering Exploratory Questions is rarely sufficient to solve a stakeholder's problem in and of itself. To really solve problems, often data scientists must turn to answering Passive Prediction Questions — the focus of this chapter — and/or Causal Questions (a topic we will return to in future chapters).

As discussed in the introduction of this book, Passive Prediction Questions are questions about the future or potential outcomes of *individual entities* (customers, patients, stores, etc.). "How likely is Patient X to experience a heart attack in the next two years?," for example, or "How likely is it Mortgage Holder Y will fail to make their mortgage payment next month?"

Unlike Exploratory Questions, data scientists don't generally come up with "*an* answer" to Passive Prediction Questions; rather, data scientists answer Passive Prediction Questions by developing statistical models that take the attributes of an entity as inputs and spit out a unique answer *for each entity*. A company interested in spam detection, for example, might hire a data scientist to develop a model that takes the content of an email as input and, for every email a person receives, answers the question "If the recipient of this email looked at it, would they consider it spam?" The exact statistical machinery one uses will vary across applications, but answering these questions is the realm where terminology like "supervised machine learning" is most likely to be used.

Since Passive Prediction Questions don't usually have "*an* answer," a data scientist faced with a Passive Prediction challenge will often start by considering the *feasibility* of developing a model to give individual-level answers to a Passive Prediction Question. "Given data on new customer behavior on my website," for example, "**can I** predict how much a customer is likely to spend over the next year?" At the end of the day, however, what a stakeholder cares about is not whether one *can* predict future spending; they care about the actual answer to the question, "Given this new customer's behavior on my website, are they likely to spend a lot over the next year?" for a given customer. For that reason — and because understanding exactly what question a model is answering is key to its effective use — this individual-level question will be our focus here.

Flavors of Passive Prediction Questions

Passive Prediction Questions come in two flavors, corresponding to the two primary business purposes detailed above.

The first flavor of Passive Prediction Questions is questions about what is likely to occur in the future for a specific individual. Answering these questions is useful for identifying individuals for additional care or attention. For example, a hospital might want to know "How likely is Patient A to experience complications after surgery?" so they can decide whether the patient should receive extra nursing attention during recovery, or a factory owner might ask "How likely is this machine to break down in the next month?" to help them determine when to take the machine offline for maintenance. This is the more intuitive flavor of Passive Prediction Question, as it accords nicely with the normal meaning of the term "predict."

The second favor of Passive Prediction Questions is questions about what *would* happen in different circumstances. Answering this type of question is the key to automation, as the "different circumstance" in question is often one in which a job is being done by an actual person. For example, a statistical model that can answer the question "if a radiologist were to look at this mammogram, would they conclude it showed evidence of cancer?" effectively is a model that automates the review of mammograms. A model that can answer the question "if you showed this email to its intended recipient, would they say it is spam?" is an algorithm that automates spam detection.

OK, but... Doesn't This Feel a Little Contrived?

You would be forgiven for asking whether I've gone a little too far in trying to force the simple task of automation into the "all data science tools are tools for answering questions" framework of the book. Yes, I see how you *could* call an algorithm that automates the review of mammograms a tool for answering the question "if a radiologist were to look at this mammogram, would they conclude it showed evidence of cancer?" But that seems awfully convoluted. Why can't we just call it an algorithm that looks for cancer in mammograms?

The answer is that while it is certainly less succinct, but as we discussed in the introduction, because of how these models are developed, it is *more accurate* to say that our mammogram reading algorithm is trying to answer the question "if a radiologist were to look at this mammogram,

would they conclude it showed evidence of cancer?" than to say that it's "looking for cancer." The way that most statistical models are developed ("trained") to answer Passive Prediction Questions is using a large dataset of "training examples" — that is, data in which the outcome we care about is included in the dataset. For predicting individual-level future outcomes, this will be historical data in which outcomes — like surgery complications or factory machine failures — have already occurred. For automation, this will be data in which a person has already completed the task, and their conclusions or actions have been recorded. To train a mammogram reading algorithm, in other words, you first need a database of mammograms that radiologists have already labeled as indicating cancer or not. And what the model is actually trained to do is not "find cancer" but rather to "give the same answers given by human radiologists."

And this is where the distinction between "predicting what a radiologist would say" and "detecting cancer" becomes important: because this kind of statistical model was trained to emulate the behavior of radiologists in labelled mammograms, the model will recapitulate any systematic biases held by the human radiologists who reviewed the mammograms in the training data. Did the radiologists struggle to detect cancer in dense breast tissue? So too will the algorithm trained on their labels.

The tendency for algorithms to replicate human biases present in training data, unfortunately, extends to gender and racial biases. In 2018, for example, Reuters reported that Amazon was forced to scrap an effort to use machine learning to automatically review resumes because it turned out the algorithm — trained on historic hiring data — was biased against women. The *exact* details of the source of the bias is unclear — for obvious reasons Amazon is not eager to report on the failure — but my suspicion is that things went wrong in one of two ways.

The first is that the algorithm was given data on all past Amazon applicant resumes, along with data on which applicants had actually been hired. The algorithm was then effectively trained to answer the question "If an Amazon hiring manager looked at this resume, is it likely they would be hired?" In that case, the algorithm was recapitulating the gender bias of previous hiring managers.

My second guess is that the algorithm was given the resumes of *current* employees along with employee reviews. In that case, the algorithm was effectively being asked to answer the question "Given this resume, is it likely this is a person a current manager would rate highly once employed?" In that case, the algorithm was recapitulating gender biases in employee reviews.

In either case, these are examples of an *alignment problem*: the people developing these models *wanted* the algorithm to pick the applicants who would be the most productive employees, but the

model they *actually* developed was trying to identify employees who looked like previously successful applicants. Had the previous hiring system been effective, this wouldn't be a problem, but because the previous system included a gender bias, so too did the resulting algorithm. But because the engineers developing these tools did not think carefully enough about the question the model was actually being taught to answer, the problem was not identified until it was too late.

Differentiating Between Exploratory and Passive Prediction Questions

If you have not felt a little confused about the distinction between Exploratory and Passive-Prediction Questions previously, there's a good chance you find yourself struggling with that issue here, and for understandable reasons.

The first thing to emphasize is that the distinction between Exploratory and Passive Prediction Questions is a distinction in one's *goal*, not the statistical machinery one might use to achieve that goal.

With Passive-Prediction Questions, our interest is in the values that get spit out of a model for each entity in the data. When answering a Passive-Prediction Question, the *only* thing we care about is the quality of those predictions, and so we evaluate the success of a model that aims to answer a Passive-Prediction Question by the quality of those predictions (using metrics like AIC, AUC, R-Squared, Accuracy, Precision, Recall, etc.). Thus, when using a logistic regression to answer a Passive-Prediction Question, we don't actually care about what factors are being used to make our predictions, just that they improve the predictions. Our interest is only the quality of our predicted values, and a good model is one that explains a substantial portion of the variation in our outcome.

With Exploratory Questions, our interest is in improving our understanding of the problem space, not in making precise predictions for each entity in our data. Thus, in the example of logistic regression, our interest is in the factors on the "right-hand side" of our logistic regression and how they help us understand what shapes outcomes, not the exact accuracy of our predictions. A good model, in other words, doesn't actually have to explain a large share of variation at the level of individual entities, but it does have to help us understand our problem space.

A model that looked at the relationship between individuals' salaries and their age, education, and where they live might tell us a *lot* about the importance of a college degree to earnings (which we

could see by the large and statistically significant coefficient on having a college degree), even if it only explains a small amount of overall variation in salaries (e.g., the R-Squared might only be 0.2).

This distinction also has important implications when working with more opaque supervised machine learning techniques, like deep learning, random forests, or SVMs. These techniques are often referred to as "black boxes" because exactly how different impute factors relate to the predictions that the model makes is impossible to understand (in other words, it's like the input data is going into a dark box we can't see into, and then predictions are magically popping out the other side). These models can be very useful for answering Passive-Prediction Questions, as they can accommodate very unusual, non-linear relationships between input factors and predicted values, but because these relationships are opaque to us, the data scientist, they don't really help us understand the problem space.

The "Passive" in Passive Prediction

The term "passive" in "Passive Prediction Questions" is meant to emphasize the distinction between Passive Prediction Questions and Causal Questions. Both Passive Prediction Questions and Causal Questions can be thought of as trying to "predict" some future outcome, but they differ in the contexts in which their predictions are valid. A full accounting of the distinction between Passive Prediction Questions and Causal Questions will have to wait until we have covered Causal Questions in detail, for the moment, we can get a sense of things by introducing a very casual definition of what it means for some cause X to effect some outcome Y.

In causal parlance, when we say that some factor X causes outcome Y (and that X is not merely correlated with Y), what we usually mean is that if we were to go out and actively change X, Y would change as a result. This isn't a fully rigorous definition, but it drives home that causation is about what happens when we *actively manipulate* X.

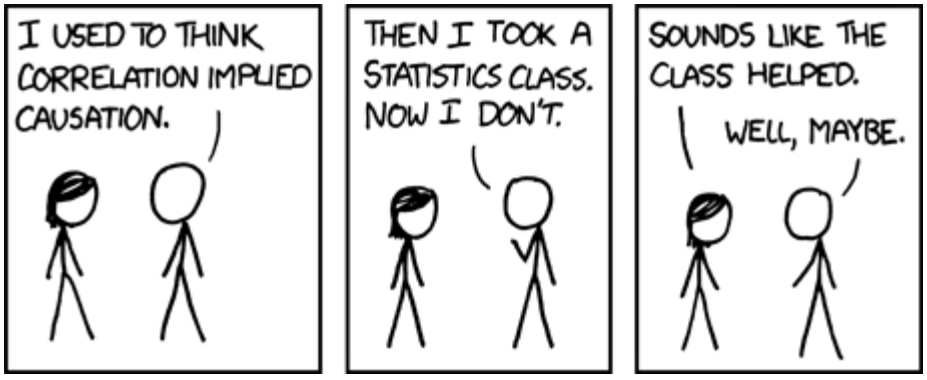
To see how this distinction illustrated, let's return to the example of a hospital interested in predicting which patients are likely to experience complications after surgery. Using past patient data, you are able to develop a model that very accurately answers the question "Given their pre-surgery vitals, how likely is a patient to experience complications after surgery?" Hooray! The hospital uses this model to determine which patients should get extra nursing visits and extra attention during recovery. You've done a great job answering a Passive Prediction Question by discovering a pattern in the world — a set of correlations between measurable variables — you can take advantage of.

Now in the course of developing this model, suppose you discover that one of the strongest predictors of complications after surgery is patient blood pressure — patients with high blood pressure are substantially more likely to experience complications than those with normal blood pressure. This leads you to wonder whether treating patients with high blood pressure with pressure-reducing medications prior to surgery might reduce complications. In other words, you now want to know the effect of going into the world and manipulating patient blood pressure — a Causal Question.

In the first case, you really don't care if blood pressure is *causing* the surgical complications, by which we mean you don't care if reducing blood pressure would reduce complications, or whether high blood pressure is just an easily observable symptom of an underlying condition that is the root cause of surgical complications (like leading a stressful life, or having relationship problems at home). In either case, the *correlation* is sufficient for your purposes of identifying patients you need to keep tabs on.

But if you want to know what would happen if you directly manipulated blood pressure, knowing that blood pressure and complications are correlated is not sufficient. After all, if living alone results in high blood pressure *and* difficulty recovering from surgery, then treating patient blood pressure may have no effect at all!

When answering Passive Prediction Questions, we are searching for correlations we can leverage to make accurate predictions, not causal relationships we can directly manipulate to shape outcomes. Indeed, those who specialize in answering Passive Prediction Questions (like computer scientists who specialized in supervised machine learning) don't really care that "correlation does not (necessarily) imply causation."



How do Large Language Models (LLMs) Fit Into

This?

Given their emergence as one of the most high profile examples of something people call "AI" these days, it's worth directly addressing how LLMs like chatGPT, Llama, Bard, etc. fit into this framework.

One of the most powerful ways to understand LLMs is to think of them as tools for answering "If I came across this text [whatever text is in the LLMs prompt, pre-prompts, or other inputs in the model's context window] while wandering around the internet,[1] what word am I most likely to encounter next?" LLMs then ask this question over and over, adding the newly selected word to the context window one at a time and then feeding the updated "conversation" back into itself as input to help it predict the next word. Indeed, this repetitiveness is why the technology behind most LLMs is called a *recurrent* neural network — because it keeps adding a word to the "conversation" you are having (it's "context window"), then feeding the updated conversation back into the model as an updated input.

To be clear, this is a little reductionist. First, LLMs are able to *abstract* away from literal text to something like the meaning of words — they recognize that "pet" and "dog" have similar meanings in the sentences "I took my dog for a walk" and "I took my pet for a walk" — but even these abstractions are the result of looking for common patterns across existing text. And second, LLMs are also "finetuned" by having humans rate responses. But these nuances don't change the fact that *fundamentally* LLMs are tools for recapitulating text and ideas that already exist in the world, with a strong bias towards what humans have *tended* to write most on the internet — a truth that both explains some of their power, but also helps to explain their fundamental limitations (e.g., their complete lack of fidelity to the truth, the fact they reflect societies' gender and racial biases, etc.).

---

[1] A more accurate way to phrase this would be "If I came across this text *in my training data*..." For most LLMs, "my training data" is a corpus that consists of both text that is *not* on the internet — like every book every published — and fails to include many things that *are* on the internet but which are difficult to crawl in an automated way. You can learn more about the corpus of internet text used by many LLMs — the 9.5 petabyte *Common Crawl* dataset — here. But it does seem very clear the *vast* majority of training data comes from the internet. LLMs makers are becoming increasingly caggy about what data they use for training — in part because they don't wish to make copyright lawsuits against them *too* easy — but books made up only 4.5% of the training data for Meta's first version of LLaMa.

Passive Prediction Questions and Internal Validity

As with Exploratory Questions, when answering Passive Prediction Questions there are two major types of concerns: those related to internal validity, and those related to external validity.

Internal Validity

Of all the places where data science is fragmented, none is more evident than in how data scientists evaluate how effectively we think a model is representing our data, *especially* when focused on prediction.

The first data science perspective on evaluating the internal validity of a model comes from the field of statistics. Statisticians have approached evaluating model fit with, unsurprisingly, methods based on the idea of random sampling and the properties of statistical distributions. They make assumptions about the distributions underlying data and use those to derive theoretically-motivated metrics. That's the origin of statistics like Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), as well as the emphasis on the validity of the standard errors assigned to factors on the right-hand side of the regression.

When computer scientists were first developing their own machine learning techniques... I'm editorializing a little here, but I think it's safe to say that initially they either didn't *know about* a lot about these metrics, or they thought that they could do a better job investing their own. So they developed the "split-train-test" approach to model evaluation: they split their data into two parts, train their model on part of the data, then test how well the model is able to predict the (known) outcomes in the test dataset.

Of course, over time these two fields have largely converged in adopting one another's methods, and some—like cross-validation—live comfortably in the middle. But if you're ever wondering why, when you get to a machine learning class, it seems like everything you learned in stats has been abandoned (or end up in a stats class and have the opposite experience), it's largely an artifact of parallel development of methods of model evaluation in computer science and statistics departments.

But the ins and outs of fitting machine learning or statistical models is not the comparative advantage of this text, and so our time is better spent turning to External Validity.

Passive Prediction Questions and External Validity

Where *internal validity* is a measure of how well a model captures the meaningful variation in the data we already have, *external validity* is a measure of how well we think that our model is likely to perform when faced with new data.

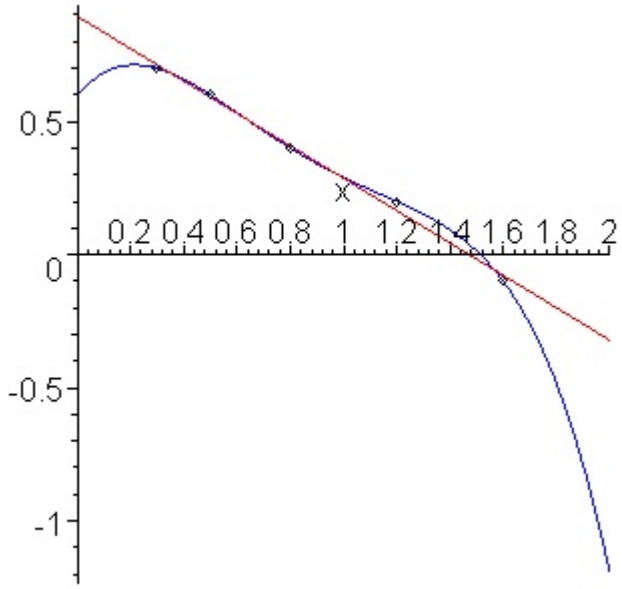
As we learned before, the external validity of a model is specific to the context in which a model is being used. A model will generally have very *high* external validity when used to answer Passive-Prediction Questions in a setting that is very similar to the setting from which the data used to train the model was collected, but *low* external validity when applied in geographically, temporally, or socially different settings.

Some of the factors that influence the External Validity to Passive Prediction Questions are the same as those that shape the External Validity of Exploratory Questions, such as the population represented in the data (patterns in data from one country will often differ from patterns in data from another country), the time period in question (consumer behavior may vary across seasons, and many patterns in data change over longer timespans). But there are other concerns that are a little more specific to Passive Prediction Questions:

Extrapolation and Training Parameter Ranges

Tools for evaluating internal validity help ensure that statistical and machine learning models will tend to fit the data on which they are trained relatively well. However, while most statistical models are capable of generating predicted values over a very broad range of input values, their reliable outside the range of values on which they were trained is often very limited. Asking a model to make predictions for inputs on which the model wasn't trained is called *extrapolating*, and is a great way to get oneself into trouble.

To illustrate, consider the two models in the figure below (source)—one a linear fit, and one a higher-order polynomial. Both model the data similarly *in the range for which data is available* — and so will perform similarly when one uses the metrics described above to evaluate the model's internal validity — but make very different predictions when asked to extrapolate values of `x` below 0 or above 2.



In addition to reducing overfitting, strategies like regularization are designed to constrain the "wonkiness" of models outside the domain on which they were trained, but almost by definition, absent data in those extended ranges, there's no way to know for certain whether the model will generalize.

What Constitutes Extrapolation?

In the example above, the term extrapolation refers to predicting values below 0 and above 2. But what constitutes an extrapolation depends in part on the complexity of the model. With a nice, interpretable linear model, it's not hard to have confidence that the model will make a reasonable prediction for $x = 0.5$, even though that specific value wasn't in the training data. But when one begins to work with highly non-linear models that allow for interactions and aren't easily interpretable — such as deep learning neural networks — the only place one can feel sure of the behavior of the model is at the exact data points in the training data. The same flexibility that allows these models to accommodate unusual non-linear relationships can also lead to bizarre behavior between actual points in the training data. A credit risk model may make perfectly reasonable predictions for a married 45-year-old women of Hispanic descent who lives in Colorado and a married 47-year-old women of Hispanic descent who lives in Colorado because both of those profiles were present in the training data, but make a crazy prediction for a a married 46-year-old women of Hispanic descent who lives in Colorado or a married 45-year-old women of Hispanic descent who lives in Montana.

Indeed, it's precisely for this reason that for many high stakes decisions, regulators are increasingly requiring the use of interpretable models that include guarantees (like monotonicity).

Basically, the more flexible the model, the more data points are required to constrain the model's behavior (the so-called "curse of dimensionality"), and the more cautious you should become. There's a reason that LLMs hallucinate despite being fed unfathomably large amounts of data.

> **Note**
>
> A common misconception among young data scientists is that the split-train-test workflow commonly used in machine learning innoculates against external validity concerns. After all, the idea of split-train-test is that models are trained on one set of observations and evaluated against an entirely different set of observations.
>
> While split-train-test can *help* reduce external validity concerns by guarding against over-fitting, a fundamental limitation of the workflow is that training observations and test observations both come from the *same context*. Indeed, because test and training datasets are created by taking a single dataset and randomly splitting the observations, they *should* always have the same properties (at least in expectation) — a guarantee one certainly won't get when moving from the data used to build a model to a real world deployment.

Adversarial Users

Another external validity concern for Passive Prediction Models is *adversarial users*. Adversarial users are users who deliberately attempt to subvert a statistical or machine learning model. The idea of adversarial users might seem like the stuff of spy novels, but they're actually *much* more common than you might think.

To illustrate, consider the Essay RoboGrader. Training an algorithm to answer the question "if a human English professor read this essay, what score would they give it?" is relatively straightforward — get a bunch of essays, give them to some English professors, then fit a supervised machine learning algorithm to that training data. What could go wrong?

The problem with this strategy is that the training data was generated by humans *who knew they were writing essays for humans to read*. As a result, they wrote good essays. The machine learning

algorithm then looked for correlations between human essay rankings and features of the essays, and as a result it could easily predict essay scores, at least on a coarse scale.

But what happens when humans realize they aren't being graded by humans? Well, now instead of writing for a human, they will write for the algorithm. They figure out what the algorithm rewards — big, polysyllabic words (don't worry, doesn't matter if they're used correctly), long sentences, and connecting phrases like "however" — and stuff them into their essays.[1]

This works because the essay writers who used polysyllabic words and long sentences in the training data happened to also be the students who were writing good essays. These were reliable predictors of scores in essays people wrote for humans. But they *aren't* a reliable predictor of essay quality in a world where students know the essays *aren't* being written for humans, just machines.

Another way of thinking about this is that we're back to the classic problem of alignment problems: they *want* the algorithm to reward good writing, but that's not actually what they trained it to do. In this case, however, the alignment problem is rearing its head because people are actively trying to exploit this difference.

While these examples are fun, not all are, and adversarial users is a HUGE and never ending problem for spam filters, network intrusion detection, credit and fraud monitoring, and more.

Goodhart's Law, Cambell's Law, and the Lucas Critique

It's worth noting that the threat of adversarial users is not a new phenomenon unique to the age of machine learning. Indeed, the idea of adversarial users is closely linked to at least three much older ideas:

> "When a measure becomes a target, it ceases to be a good measure."

- Goodhart's Law, named for Charles Goodhart.

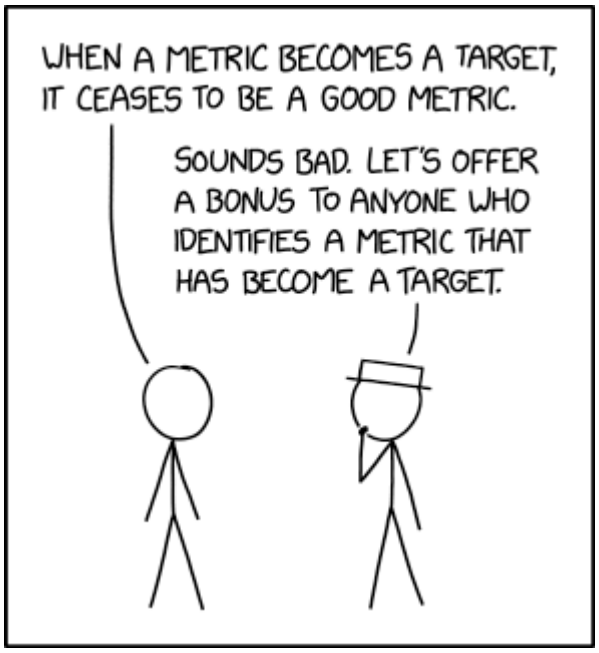
> "The more any quantitative social indicator is used for social decision-making, the more subject it will be to corruption pressures and the more apt it will be to distort and corrupt the social processes it is intended to monitor"

- Campbell's Law, named for Donald Campbell

> "Given that the structure of any [statistical] model consists of optimal decision rules of economic agents, and that optimal decision rules vary systematically with changes in the structure of series relevant to the decision maker, it follows that any change in policy will systematically alter the structure of [statistical] models."

- Lucas Critique, named for Robert Lucas

And since no idea is serious until it's been importalized in an XKCD comic:



[1] I cannot for the life of me find the podcast where I first discovered this phenomenon being discussed, but it had lots of great colorful examples of school kids doing this. For now all I can find are these articles: here, here, here, here and here.

The Right Way To Be Wrong

With the rise of online machine learning competitions like Kaggle and an academic literature fixated on publishing papers showing marginal improvements in performance metrics on standard benchmarks, you could be forgiven for thinking that the hardest part of data science is finding the right model and features to max out standard metrics like Area Under the Curve (AUC), Accuracy (share of cases correctly classified) or F1 scores.

However, this is far from the case when it comes to solving real-world problems. Yes, advancement in academic computer science is often tied to one's ability to write a new algorithm that performs marginally better on standard benchmarks, and most problem sets or online competitions you encounter will pre-specify how model performance will be evaluated.

But when it comes to solving real-world problems, determining what success looks like is actually a core part of your job as a data scientist. And what makes this task difficult is not how you measure your model's successes — the number of true positives and true negatives the model generates — but in determining the types of mistakes it makes when it gets things wrong.

Understanding how a model fails is just as important as minimizing mistakes in the first place (i.e., maximizing accuracy). Depending on the context, there can be *huge* asymmetries regarding the consequences of false positives and false negatives. Tell someone with cancer they're fine (false negative), and the result may be the death of the patient from an easily treatable condition; tell someone healthy there's a chance they have cancer, and you may cause stress and additional tests, but the patient death is very unlikely.

Similarly (but in a much less scary context), classify a credit card applicant as a "good credit risk" who is not actually credit-worthy (a false positive), and your company may lose the credit limit on the card they issue; classify someone as high risk who is actually not, and your company may lose the transaction fees that customer would have generated, but they won't lose tens of thousands of dollars in unpaid bills.

When answering Passive Prediction Questions, the choice of how to balance true positives, true negatives, false positives, and false negatives is *the* bridge between the math of statistics and machine learning and the specifics of real-world problems. And as a result, an ability to speak thoughtfully about how to balance these interests is one of the most important differentiators between data scientists who have only ever fit models for problem sets and data scientists business leaders trust to solve their problems.

Note

In case you need more motivation to care about how best to distribute your true and false positives and negatives to best solve your stakeholder's problem, allow me to offer the following: if you are comfortable just endorsing a single success metric (accuracy, F1 score, AUC), *most* Passive Prediction Questions can be answered relatively well by automated tools. And that means that if the only thing that differentiates you from the next data scientist (or generation of chatGPT) is that you can get a slightly higher AUC than the person sitting next to you, how much value do you think you are likely bringing to your stakeholder (and thus how well paid)?

Don't believe me? Consider scikit-learn — by design, nearly all of the algorithms in that package have the exact same APIs — you run `test_train_split()`, `.fit()`, and `.predict()`. That means it's almost trivially easy to write a program that just loops over all the models in the library, fits them, and checks their performance against the simple metric you chose. Indeed, products exist that do just this, often under names like `AutoML`.

Will these do as well as a well trained data scientist? Not yet — there's skill in feature engineering and choosing which path to go down when computationally constrained. And if you're someone working at the forefront of developing new machine learning algorithms or infrastructure, then this does not apply to you. But if you are someone who is primarily interested in applying the best tools of data science to solve real world problems, then you should also bear in mind that anything easily computable lends itself to automation.[1]

Thinking carefully about your stakeholder's problem and being able to get them to articulate the relative value they place on true and false positives and negatives, then translating that domain expertise into an optimization problem — *that* is a task that requires substantially more critical thinking and interpersonal skills, and consequently is less likely to be made obsolete any time soon.

The Problem with Accuracy

Let's begin our discussion about balancing true and false positives and negatives with my least favorite metric for classification problems: Accuracy.

There is perhaps no better way for a data scientist to demonstrate they don't know what they're doing than for them to proudly proclaim that their model has an accuracy score of ninety-something percent without additional context. And yet it is a mistake that I see constantly. It is as though, being students, young data scientists implicitly assume that accuracy scores, like grades, exist on an absolute scale, where values in the 90s are "A"s and something to celebrate and values in the 70s are "C"s and something to feel bad about, when in reality neither is necessarily the case.

How do I know this fallacy is common, you ask? As Director of Admissions for the Duke Masters of Interdisciplinary Data Science (MIDS) program, I read hundreds of Statements of Purpose essays and resumes every year from aspiring data scientists from around the world. And despite having done this for years, I continue to be shocked by the number of applicants who proudly proclaim something like "I fit a model using XYZ method and was able to achieve a 95% accuracy score," or report an accuracy score in the 90s in their resume as though those numbers, absent context, were meaningful.

Reporting Accuracy Without Context

There is perhaps no better way for a data scientist to demonstrate they don't know what they're doing than for them to proudly proclaim that their model has an accuracy score of ninety-something percent without additional context.

So why is reporting accuracy scores without context such a problem? There are at least three reasons.

Accuracy and Imbalanced Data

Most data you will encounter in your career will be *imbalanced*, meaning that one of the outcomes you are trying to predict with your model (assuming a classification task) will be much, much less prevalent than the other. In these situations, because accuracy is just "the share of cases correctly classified," getting high acccuracy can be achieved trivially by always predicting the more prevalent outcome.

To illustrate, consider routine mammograms. Mammograms are x-rays of women's breast tissue used to screen for early signs of breast cancer. In the United States, it is recommended that all women over 40 get a mammogram every two years. Unsurprisingly, therefore, *vast* majority of routine mammograms are medically unremarkable. According to the Susan G. Komen society, roughly 90% of routine mammograms are perfectly normal and require no followup.[2] Thus a

"model" that reports that *any* routine mammogram it is given is normal will immediately achieve an accuracy score of 90%.

Moreover, most data scientists wouldn't even consider 90/10 data to be particularly imbalanced. In any given year in the United States, only about 3% of single-family residential mortgages are in a state of delinquency[3], and fraudulent credit card purchases make up less than one-tenth of 1% of all credit card transactions. That means a "model" that reports all mortgages are in good standing or that all credit card transactions are valid will immediately have accuracy scores of 97% and > 99.9, respectively.

Relative to What?

The second problem with reporting accuracy scores absent context is that the value of a model can only ever be evaluated relative to what came before. A model with a 93% accuracy score is unlikely to be of particular value to a business if the model they were using before you arrived had an accuracy score of 98%, and your model does not have any other benefits to offset its lower accuracy. Similarly, a model with an accuracy score of 70% may constitute a considerable innovation to a business that could not make predictions more accurately than with 50%-50% odds.

Indeed, a related issue with treating accuracy as an absolute scale akin to academic grades is that model performance depends on the amount of *signal* in the data on which it is trained. Data only has so much predictive information in it, and in theory, the way a model *should* be evaluated is in terms of how close it comes to harnessing that full predictive potential of the data on which it is being trained. Of course, we are not gods, and so we will never know the exact predictive potential of a given dataset, but the principle is one to bear in mind — the potential of a model is bounded by the data on which is being trained, and the only way to get a model that exceeds that true performance frontier is by overfitting your data (creating an *illusion* of better performance that will not hold up when the model is actually deployed).

But What About The Mistakes?

The third reason reporting accuracy scores without context — and indeed the reason that accuracy is a problematic metric in general — is that it tells you nothing about the types of misclassifications your model is making. Are all its errors false positives? Are they all false negatives? In what ratio do they occur?

Accuracy says *nothing* about how different types of mistakes are being balanced, which is why accuracy is sometimes a fun statistic to use on problem sets but a *terrible* metric in the real world.

ROC and Area Under the Curve (AUC)

"OK, fine," I hear you say, "but no one uses accuracy anymore — we all use ROC AUC scores!"

First, again, I can tell you from reading hundreds of essays and looking at hundreds of resumes, accuracy is still an *extremely* commonly used metric. But setting that aside…

Yes, AUC is certainly a more holistic metric than accuracy. Where accuracy evaluates the share of cases correctly classified, AUC averages the share of correct positive predictions across the full range of classification thresholds (one of the reasons it is commonly used in competitions). But it is not a substitute for thinking carefully about the correct metric *for the specific problem you are seeking to solve*.

First, most models are deployed at a specific classification threshold, so averaging across all classification thresholds may create a good *general purpose* indicator of a model's performance, but it makes AUC poorly suited to evaluating how well a model will perform in any *specific* context (i.e., at a specific classification threshold).

Second, the ROC AUC metric is myopically focused on the proportion of correct positive predictions. But depending on our problem, we may also care about the ratio of false negatives to false positives or other properties of our negative predictions.

Choosing the Best Way to be Wrong

How, then, should one approach being more thoughtful about model evaluation?

The first step is always to evaluate the *relative value* of the four different types of classifications: true positives, true negatives, false positives, and false negatives. Writing a model that reviews the results of blood tests for signs of a terminal but treatable disease? You probably want to associate a strong negative value with false negatives (where you tell a sick patient they're healthy) and a smaller negative value with false positives (being told you might have a lethal condition is stressful, even if later tests (which may have their own risks) may show it to be a false positive!). And you may then normalize your true positives and true negatives to zero.

You can then fit your classification model and, for each classification threshold, calculate the "cost" of the resulting distribution of true and false positives and negatives. Find the model and classification threshold that minimizes this problem-specific cost function, and you've identified the model and threshold that's best *for your specific problem*.

Where do these values come from? Sometimes your stakeholder will be able to tell you the actual financial cost of different types of errors (e.g., when deciding whether to issue someone a credit card), but other times these values are more subjective. What's the relative cost of falsely telling someone they may have a terminal disease condition? How might that vary depending on the risks associated with any followup procedures required to confirm a diagnosis or the amount of time it takes for the diagnosis to be confirmed? Those are hard, subjective questions you may not have the domain expertise to answer yourself. But because you *understand* the role of these values in how your eventual model will operate, you can raise these questions with your stakeholder (who should have better domain knowledge) and solicit values from them.

Note

Up until now, we've emphasized how we manage errors in the context of discrete, binary classification tasks, but it is worth emphasizing that this is only because binary classification is the easiest context in which to think about these problems. However, the issues raised her apply equally to classification tasks with more than two categories, and to efforts to answer Passive Prediction Questions about continuous outcomes. Latent in any model you use is a cost function, and implicit in that cost function is how mistakes are evaluated.

Linear regression, for example, minimizes the sum of squared errors across all observations, and (by default) it gives equal weight to the squared error associated with each observation. But if you don't feel that's an appropriate weighting scheme, you are not bound to it — weighted linear regression is a version of linear regression where the user provides a set of weights to associate with each observation. Have some customers you know are more valuable to your company? Perhaps you want to have the model give more weight to errors associated with those customers so the final model performs better for those customers. Or working with data from stores with different sales volumes? Maybe you want to give more weight to stores with larger sales volumes.

Don't want to work with squared errors at all? Great! There's a whole discipline called robust linear modeling that uses different norms for evaluating errors (often with the goal of reducing the influence of outliers, as the name implies, but all they are doing is modifying how the errors the model seeks to minimize are handled).

The Ethics of Misclassifications

How our models make mistakes is often not just a financial decision; when statistical models are used to make high stakes decisions — like who is sent to prison and who is let out on bail pending trial — how errors are distributed can often be deeply, *deeply* ethically frought issue.

To illustrate, let's consider the example of Risk Assessment models. Risk Assessment models are models used in the US criminal justice system that are designed to help judges and parole boards determine the risk that a criminal defendant or incarcerated person may re-offend if released from jail. These are increasingly commonly used for purposes like determining whether arrested individuals should be released while they await trial and whether incarcerated individuals should be

paroled (released before the end of their prison sentence to a half-way house or monitored release).

The way Risk Assessment models are being used in the US has many, many problems. But a recent debate about an aspect of one of the most commonly used models — the COMPAS Risk Model — is illuminating about the moral issues that arise when handling misclassifications. If you would like to learn more about this debate and difference concepts of algorithmic fairness, I strongly recommend Bias In, Bias Out (2019), a Yale Law Journal article by Sandra G. Mayson.

In 2016, the investigative news outlet ProPublica published a piece about COMPAS called Machine Bias. In it, journalists Julia Angwin, Jeff Larson, Surya Mattu and Lauren Kirchner used data from Florida to evaluate how well the COMPAS risk assessment model predicted criminal recitivism (the term for when a criminal defendant commits another crime in the future).

The ProPublica analysis determined, in part, that:

> Black defendants were often predicted to be at a higher risk of recidivism than they actually were. Our analysis found that black defendants who did not recidivate over a two-year period were nearly twice as likely to be misclassified as higher risk compared to their white counterparts (45 percent vs. 23 percent).

In other words, the false positive rate for Black defendants was higher than it was for White defendants.

While apparently rather damning of the model, COMPAS' response was that this was actually a consequence of the fact that their model generated equal True Positive Rates across Black and White defendants. And while I am generally loath to defend COMPAS, as I think it has *many* problems and should not be in use, in this particular case they have a point.

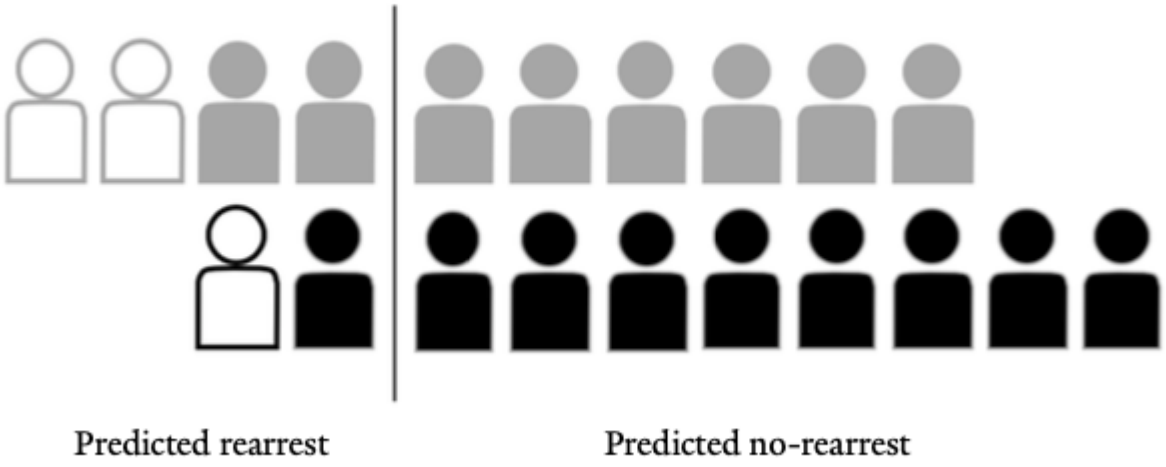
As Sandra Mayson points out in her paper, the issue is that the likelihood someone will be arrested for a future crime (be arrested for recitivism) is imbalanced by race. As a consequence, a model that has the same True Positive Rate for Black and White defendants (i.e., the share of people predicted to re-offend who do re-offend is the same for both groups) will *necessarily* have different False Positive Rates for the two groups.

To illustrate, consider the following figure from Mayson's paper with two groups, drawn as grey and black. Solid outlines are individuals who do not recitivate, while hollow figures are those who are re-

arrested. In the grey group, 2/10 individuals are re-arrested, while only 1/10 are re-arrested in the black group. The vertical line represents the classification threshold used by the model.

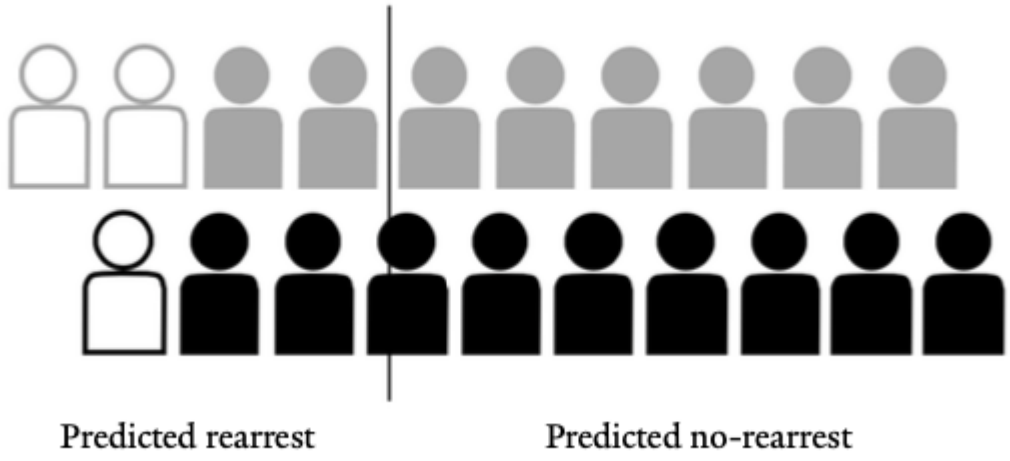
Note that this implies the grey outlines are in the situation akin to that of Black Americans (more likely to be arrested for recitivism) and the black outlines are in the position of White Americans (less likely to be rearrested), which is a little confusing.

FIGURE 1.
GROUPS WITH DIFFERENT BASE RATES OF REARREST; PREDICTIVE PARITY



With the line in the location shown in the figure, we can see that the model has the same True Positive Rate for both populations — grey and black (50%). But as a result, the *False* Positive Rate is higher for the grey individuals (2/8). Thus while the ProPublica finding is true — the False Positive Rate of COMPAS is higher for Black defendants — the only way to even this out would be to shift the classification threshold for the black outlines over, as illustrated in Figure 2 from her paper:

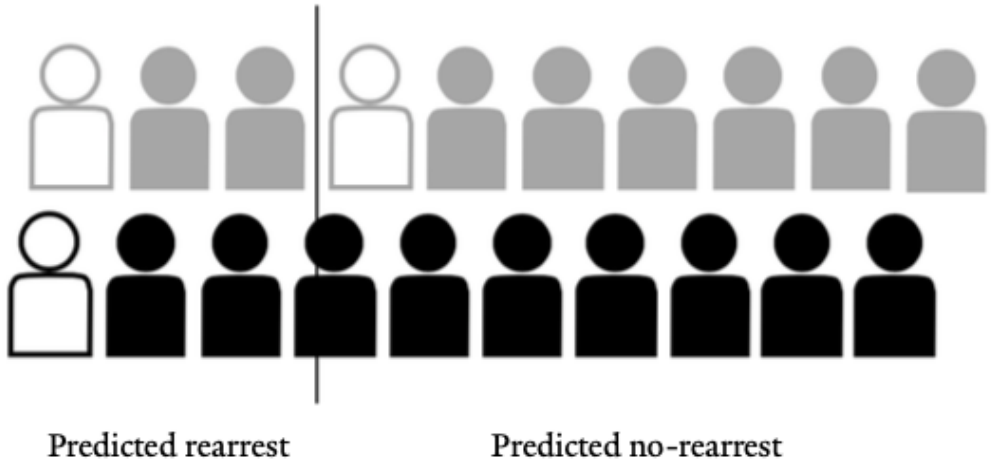
FIGURE 2.
GROUPS WITH DIFFERENT BASE RATES OF REARREST; PARITY IN FALSE-POSITIVE RATES



This balances the False Positive Rates for the two groups, but in doing so results in the True Positive Rate being lower for the black outlines.

Can we do better? Well, we could get equal True Positive Rates and False Positive Rates, but only by accepting differential False *Negative* rates, as illustrated in Mayson's Figure 3:

FIGURE 3.
GROUPS WITH DIFFERENT BASE RATES OF REARREST; PARITY IN FALSE-POSITIVE RATES AND PREDICTIVE PARITY



As Mayson writes:

> As this example illustrates, if the base rate of the predicted outcome differs across racial groups, it is impossible to achieve (1) predictive parity; (2) parity in false-positive rates; and (3) parity in false-negative rates at the same time (unless prediction is perfect, which it never is). Computer scientists have provided mathematical proofs of this fact.[4] When base rates differ, we must prioritize one of these metrics at the expense of another. Race neutrality is not attainable.

Note

Please note that in the discussion of imbalanced recitivism arrest rates, I said *arrested for* recitivism — a huge problem with all of these risk assessment models is that we know from many studies that even in situations where Black and White Americans *commit* crimes at similar rates (like drug use), Black Americans are substantially more likely to be *arrested* for those crimes. Thus, an imbalance in arrests for recitivism do not necessarily imply differences in *actual* recitivism.

Indeed, this form of inequity is also a problem on the input side — Risk Assessment models take into account whether a defendant has prior convictions, but given Black Americans are more likely to be arrested even in situations where were know criminal behavior occurs at similar rates for Black and White Americans, and Black Americans are likely to have lower-incomes and thus less likely to have lawyers who can get charges dismissed, these models tend to treat them as higher risk.

So what is the right answer? Well, there isn't one — each of these different schemes is defensible under different ethical frameworks. And that's the problem — there's no algorithm or machine learning metric that can tell you the "right" answer in a situation like this — it requires critical thought, engagement with prescriptive Questions, and careful discussion with your stakeholder.

---

[1] An idea closely related to a point made by Steven Pinker in his 1994 book *The Language Instinct*: "The main lesson of thirty-five years of AI research is that the hard problems are easy and the easy problems are hard. The mental abilities of a four-year-old that we take for granted – recognizing a face, lifting a pencil, walking across a room, answering a question – in fact solve some of the hardest engineering problems ever conceived... As the new generation of

intelligent devices appears, it will be the stock analysts and petrochemical engineers and parole board members who are in danger of being replaced by machines. The gardeners, receptionists, and cooks are secure in their jobs for decades to come."

[2] The vast majority of the roughly 10% of scans that are abnormal are eventually determined to be false positives.

[3] In other words, they have failed to make a mortgage payment for at least a certain period of days.

[4] See Alexandra Chouldechova, *Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments*, 5 BIG DATA 153 (2017) and Jon Kleinberg et al., *Inherent Trade-offs in the Fair Determination of Risk Scores*, LEIBNIZ INT'L PROC. INFORMATICS, Jan. 2017, at 43:1, 43:4.

Performance Differential Is (Often) A Myth

There's just only so much signal in the data.

Regulatory Compliance

Trusting Model == Trusting Data

Ethical Transparency

Ease of Adoption

(Cite study)

No Magic, Not Undue Deference

- Most models internal, where IP-black-box doesn't matter
- Handoff and implementation easier

- Ethical review.

Using Causal Questions

In our last reading, we learned a little about what it means to measure a causal effect, and why it is inherently difficult. That is a topic we will return to shortly, as understanding why measuring causal effects is hard is key to being able to measure them effectively. But first, take a moment to discuss how Causal Questions come up and are addressed in practice to help contextualize the more technical readings that will follow.

In our previous readings, we learned how answering different types of questions can help us better understand the world around us. By answering Exploratory Questions, we can better understand the contours of our problem — where our problem is most acute, whether there are groups who have figured out how to get around the problem on their own, etc. This, in turn, can help us prioritize our subsequent efforts. By answering Passive Predictive Questions, we can help identify individual entities — patients, customers, products, etc. — to whom we may wish to pay extra attention or recommend certain products. By answering Passive Predictive Questions, we can also automate tasks by predicting how a person (or more complicated process) *would* have classified an entity.

In both cases, however, answering these questions only helps us better understand the world *around* us. But to the extent to which, as data scientists, we want to intervene to directly address problems, we are rarely interested in just knowing about the world around us — we want to *act* on the world, and wouldn't be great if data science could provide us with a set of tools designed to help us predict the *consequences* of our actions?

Enter *Causal Questions*. Causal Questions ask what *effect* we can expect from *acting* — that is, actively *manipulating* or *intervening* — in the world around us in some way. For example, if we pay to show an ad to a specific customer, what will the *effect* of that choice be the likelihood they buy something on our website? Or if we choose to give a new drug to a patient, what will the *effect* of that choice be on their disease?

Because of their potential to help us understand the future consequences of our actions, it should come as no surprise that the ability to answer Causal Questions is of *profound* interest to everyone from companies to doctors and policymakers. At the same time, however, it may also come as no surprise that answering Causal Questions is an inherently challenging undertaking.

In this reading, we will discuss where Causal Questions arise in practice, and the workflow one goes about answering them, glossing over the nuances of *how* exactly we answer Causal Questions. Then in our next reading, we will turn from workflows to theory and discuss in detail what it actually means to measure the effect of an action X (e.g., administering a new drug to a patient, or showing an ad to a user) on an outcome Y (patient survival, customer spending, etc.). This section will, at times, feel a little abstract and woo-woo, but please hang in there. Answering Causal Questions is as much about critical thinking as it is about statistics, and the concepts introduced here will prove crucial to your ability to be effective in this space.

When Do Causal Questions Come Up?

Causal Questions arise when stakeholders want to *do* something — buy a Superbowl ad, change how the recommendation engine in their app works, authorize a new prescription drug — but they fear the action they are considering may be costly and not actually work. In these situations, stakeholders will often turn to a data scientist in the hope that the scientist can "de-risk" the stakeholder's decision by providing guidance on the likely effect of the action *before* the action is undertaken at full scale.

Usually, the action the stakeholder is considering will not have been pulled out of a hat. Rather, a stakeholder will generally pose a Causal Question because they have some reason to suspect a given course of action may be beneficial. Indeed, Causal Questions often emanate from patterns discovered when answering Exploratory or Passive Predictive Questions.

Where Causal Questions Come From — An Example

For example, suppose the Chief of Surgery at a major hospital is interested in reducing surgical complications. They begin by asking "What factors predict surgical complications?" (a Passive Predictive Question) and develop a predictive model that allows the hospital to identify patients who are likely to have issues so that caretakers can provide additional support to these patients during recovery.

In the course of developing this model, the Chief discovers that one of the strongest predictors of surgical complications is patient blood pressure — patients with high blood pressure are substantially more likely to experience complications than those with normal blood pressure.

This leads the Chief to wonder about whether they could reduce surgical complications if they treated patients with high blood pressure with pressure-reducing medications prior to surgery. In other words, the Chief Surgeon wants to know "What effect treating patients with high blood pressure would have on surgical complication rates?"

But rather than just starting to give all patients with high blood pressure new drugs (and delay their surgeries while the drugs take effect), the Chief wants *you* to provide a more rigorous answer to their question. After all, high blood pressure may be *causing* the complications (and thus the medicine may help), but it could also be that high blood pressure isn't the *cause* of the complications, but rather the *symptom* of a third factor that causes both high blood pressure *and* complications that makes people with high blood pressure different from those with low blood pressure — like leading an overly stressful life. The Chief doesn't need to know whether high blood pressure is the *cause* of complications or just a "warning light" that identifies people at risk to use that information for directing additional support to those patients during recovery; but it *does* matter for determining whether it makes sense to delay surgeries to teach patient high blood pressure!

This is, of course, just one example, and it's not hard to imagine others. Perhaps your online retailer stakeholder has noticed that one of your competitors has stopped showing customer reviews in the search results, for example, so they suspect it must be improving sales for your competitor and want to know if it would work for your site too! But the point is that Causal Questions generally don't appear out of the blue, but rather because someone has noticed a pattern in the world and wants to act on it. Thus, many Causal Questions may actually take the form of hypotheses or hunches that your stakeholder wants to be investigated.

The Two-Fold Challenge of Causal Questions

In our last reading, we discussed how answering Causal Questions is difficult in part because measuring the effect of any action on any outcome is a definitionally difficult endeavor. But answering Causal Questions is also difficult for a more practical — less epistemological — reason: risk aversion!

As we noted above, stakeholders generally turn to data scientists because they want to know the likely consequences of an action *before they actually undertake the action at full scale.* This may seem obvious, but it bears repeating — not only is answering Causal Questions hard because we never get to measure outcomes in both a universe where our treatment occurs and also a universe

where it does not (the Fundamental Problem of Causal Inference), but *also* because stakeholders want to know about the likely consequences of an action they aren't ready to actually undertake!

As a result, the job of a data scientist who wants to answer a Causal Question is to design a study that not only measures the effect of a treatment, but also does so in a setting that is enough like the context in which the stakeholder wants to act that any measured effect will generalize to the stakeholder's context.

We call these two objectives of a study *internal validity* (how well the study answers the Causal Question *in the setting the study is conducted*) and *external validity* (how well the results of the study generalize to the context the stakeholder cares about). And to provide value to a stakeholder, a data scientist's analysis must have both.

Internal and External Validity: An Example

To illustrate, suppose you work for a medical device company in Boston that wants the US Food and Drug Administration (FDA) to authorize a new cochlear implant your company has developed (a partially surgically implanted device for helping those with certain types of hearing loss regain hearing). Before authorizing the device, the FDA wants to be sure that it's safe and effective — in other words, it wants to know what the *effect* of authorizing the device for patients throughout the United States would be on patient health.

Your job, therefore, is to conduct a study that (a) convincingly measures the effect of the device on patients (has high internal validity), *and* (b) does so in a way that convinces the FDA that the findings from *your study* are likely to be the same as what would be seen if the device were being used across the United States (has external validity to the context the FDAs cares about).

In medical trials, internal validity is usually ensured by conducting a randomized experiment — referred to as a Randomized Control Trial (RCT) in medical circles — according to a set of FDA requirements. We'll discuss what features must be present for us to have confidence in the results of a randomized experiment soon, but they are things like making sure that the people in the control group look like the people in the treatment group in terms of things we can measure (age, gender, etc.) to help us feel confident that when people were randomly assigned to control and treatment groups, we didn't end up in a really unlikely situation where, purely by chance, only men ended up in control group and only women ended up in treatment group.

External validity, by contrast, comes from things like *who* is enrolled in the trial. The average age of children getting cochlear implants is between 2 and 3, so if your study only included children between 12 and 18 months of age, the FDA may worry that the results of the study would not *generalize* to the US population as a whole.

In the context of a clinical trial, this issue of external validity may seem easy to address — just get a sample of people who "look like" the US population (when applying for US FDA approval)! Historically, however, women[1] and minorities have been underrepresented in clinical trial participants.[2] Moreover, the people designing clinical trials often limit enrollment to participants who, aside from the specific condition being treated, are healthy to avoid complications. This reduction in complications may increase the *internal* validity, but as many patients face more than one health challenge, it may reduce external validity.

Outside of drug or medical device trials, however, external validity can much harder to establish. For example, the functionality of many internet services and apps depends on network effects — testing out a new social feature on Instagram by making it available to only a handful of users in a randomized trial (an A/B test, in the language of tech companies) may not give you a meaningful sense of how the feature would be used if it was visible to all users. And the way that bank customers use a new budgeting app in the context of a two-week study may not be indicative of how they would use it over the long run when the feature is no longer new.

External Validity To *What*

Throughout this text, we will refer to whatever course of action the stakeholder is actually considering pursuing as the "stakeholder's context." As a result, when discussing the external validity of a study, we will implicitly be referring to its external validity *to the stakeholder's context*. But it is worth emphasizing that while the internal validity of a study is a single things — you have some level of confidence that the study measured the effect they set out to measure — external validity is *relative*. A study conducted in a hospital in Denver, Colorado may have good external validity from the perspective of a doctor in a Pheonix, Arizona hospital, but that same study may not have very good external validity from the perspective of a doctor at a hospital in Pune, India. So always remember that external validity is not an absolute property of an analysis, but a property that is *relative* to the context to which one wishes to generalize the results.

The Causal Question Work Flow

Before we dive into the technical details of answering Causal Questions, it's worth starting with a high-level overview of how data scientists approach answering Causal Questions.

Identify Relevant Previous Studies

Once a Causal Question has been posed, the next step is **to identify any research that has already been done** that may help answer your causal question. It's hard to overstate how often this step is overlooked by data scientists, but it's *such* a no-brainer once you think of it! There's no reason to spend days or weeks trying to design a study to answer a question if someone else has already put the time and money into doing it for you!

If your stakeholder is somebody who works in public policy or medicine, then the first place to look for previous studies is in academic medical or policy journals. But don't assume that if you aren't working on a medical or public policy question you won't be able to find an answer to your question in academic or pseudo-academic publications — lots of data scientists present research done at private companies at ``industry'' conferences like the MIT Conference on Digital Experimentation (CODE@MIT) or the NetMob Cellphone MetaData Analysis Conference!

And if you are at a company, ask around! Someone at your own company may have looked into a similar question before, and talking to them could save you a lot of effort.

Evaluate Previous Studies

If you do find studies, then for each study you will have to ask yourself two questions:

- **Did the study authors do a good job of answering the Causal Question? *in the context they were studying*?**
- **Do I believe that the *context* in which the study was conducted is similar enough to my own context that their conclusions are relevant to me?**

This first question is about the *internal validity* of the study, and we'll talk at length about how to evaluate that in the context of causal inference in the coming weeks. The second question is about the *external validity* (i.e., the generalizability) of the study to your context. There are lots of

extremely well-conducted studies in the world that may be seeking to answer the same question as you, but if, for example, they investigated the effect of a new drug in *young* patients, and your hospital only treats very old patients, you may not be comfortable assuming their results are good predictors for what might happen in your hospital.

Plan A New Study

If you were unable to find any studies that answer your Causal Question satisfactorily (either on their own or in combination), then it may be time to do a study of your own!

When most people think about answering Causal Questions, their minds immediately jump to randomized experiments. Randomized experiments are *often* the best strategy for trying to answer Causal Questions, but they are not always the best choice.

Studies designed to answer Causal Questions can be divided into roughly two types: experimental studies and observational studies.

In an experimental study, a researcher has control over everything that happens in the study, including who enrolls in the study and also who in the study gets assigned to the treatment group and who gets assigned to the control group. Nearly all clinical trials, A/B tests where the version of a website or app users see is randomly determined, and field experiments where, say, voters are randomly assigned to receive different types of mailers from political campaigns to measure their effect on voter turnout are all examples of "experimental studies."

In an observational study, by contrast, researchers use data from a context where the researchers did not control who was treated and who was not. This includes data from public opinion surveys, data on user behavior and demographics, or census data.

(We say studies can be divided into roughly two types because some studies fall into a category sometimes called "quasi-experimental." In these studies, researchers were not in control over who was treated and who was not, but they have some reason for thinking that *something in the world* — like a chance storm, or a draft lottery — caused who was treated and who was not to be determined randomly. But these types of studies tend to be more relevant for academics than applied data scientists, and evaluating them is incredibly difficult, so we will largely ignore them in this text.)

While it is sometimes believed that only experimental studies can generate valid answers to Causal Questions, this is *unequivocally untrue*, as is the slightly more generous version of this claim, that experimental studies always constitute the best form of evidence for answering Causal Questions. As we will explore in *great* detail in the coming days, the validity of conclusions drawn from *both* experimental and observational studies rests on whether a number of fundamentally untestable assumptions hold. As a result, both types of studies are capable of providing meaningful answers to causal questions *and* of being deeply misleading.

Moreover, while experimental studies often (but not always) have greater *internal validity* (they are often better able to ensure that they have measured the true causal effect in the lab), this often comes at the expense of lower *external validity*, because ensuring the researchers can control who is treated and who is not requires operating the study take place in a highly monitored, often artificial and unrealistic setting. Observational studies, by contrast, are often based on data collected in the real world, and as a result may yield answers that tell us more about what is likely to happen in our own real-world application, even if they have somewhat lower internal validity.

Wrapping Up and Next Steps

Hopefully, this reading has given you a better sense of *how* Causal Questions are used to solve stakeholder problems, and when and where they come up in the life of a practicing data scientist. In the readings that follow, we will turn first to the details of the *Potential Outcomes Model*, a rigorous, formal statistical framework for understanding the Counterfactual Model of Causality. This framework will not only provide a presentation of the Counterfactual Model of Causality that may be appealing to those who draw intuition from mathematical formalism, but also machinery that we can use to evaluate how much confidence we can have in answers generated using different methods of answering Causal Questions — including both experimental and observational studies.

---

[1] In 1977, the FDA actually banned enrollment of women of "childbearing potential" from Phase 1 and Phase 2 clinical trials in the interest of avoiding birth defects.

[2] This seems likely to be due, in part, to hesitancy to enroll in clinical trials by individuals aware of past abuses of minority patients, as in the Tuskegee Syphilis Study.

Answering Causal Questions

In this reading, we turn the surprisingly slippery question "What do we mean when we say X causes Y, and how do we measure the effect of an action X (e.g., administering a new drug to a patient, or showing an ad to a user) on an outcome Y (patient survival, customer spending, etc.)?"

This section will, at times, feel a little abstract and woo-woo, but please hang in there. Answering Causal Questions is as much about critical thinking as it is about statistics, and the concepts introduced here will prove crucial to your ability to be effective in this space.

What Does It Mean for X to Cause Y?

To understand what it means to answer a Causal Question, and why answering Causal Questions is intrinsically hard, we must start by taking a step back to answer the question: "what do we mean when we say some action X *causes* a change in some outcome Y?"

Seriously, what do we mean when we say "X causes Y?" Try and come up with a definition!

While this question may *seem* simple, it turns out that this question has been the subject of serious academic debate for hundreds of years by philosophers no less famous than David Hume. Indeed, even today there is still debate over how best to answer this question.

In this course, we will make use of the *Counterfactual Model of Causality* (sometimes called the Neyman-Rubin causal model). In plain English, it posits that for "doing X to cause Y", it must be the case that if we do X, then Y will occur, and if we did not do X, then Y would not occur. This is by far the most used definition of causality today, and yet remarkably, it only emerged in the 20th Century and was only really fleshed out in the 1970s. Yeah... that recently.

Measuring the Effect of X on Y

At first blush, this definition may seem simple. But its simplicity belies a profoundly difficult practical problem. See, this definition relies on *comparing* the value of our outcome Y in two states of the world: the world where we do X, and the world where we don't do X. But as we only

Counterfactual Model of Causality

For it to be the case that doing X causes Y", it must be the case that if we do X, then Y will occur, and if

get to live in one universe, we can never perfectly know what the value of our outcome Y would be in *both* a world where we do X and one where we don't do X for a given entity at a given moment in time. As such, we can **never** directly measure the causal effect of X on Y for a given entity (say, a given patient or customer) at a given moment in time — a problem known as the **Fundamental Problem of Causal Inference** (`causal inference` being what people call the practice of answering Causal Questions).

we did not do X, then Y would not occur.

To illustrate, suppose we were interested in the effect of taking a new drug (our X) on cancer survival (our Y) for a given patient (a woman named Shikha who arrived at the hospital on June 18th 2022). We can give her the drug and evaluate whether she is still alive a year later, but that alone can't tell us whether the new drug *caused* her survival according to our counterfactual model of causality — after all, if she survives maybe she would have survived even without the drug! To actually know the effect of the drug on Shikha *by direct measurement,* we would have to be able to measure her survival both in the world where we gave her the drug *and* the world where we did not and compare outcomes.

Since we can never see both states of the world — the world where we undertake the action whose effect we want to understand and the world where we don't — almost everything we do when trying to answer Causal Questions amounts to trying to find something we *can* measure that we think is a *good approximation* of the state of the world we can't actually see.

A quick note on vocabulary: by convention, we refer to the action whose effect we want to understand as a "treatment," and the state of the world where an entity receives the treatment as the "treated condition." Similarly, we refer to the state of the world where an entity does *not* receive the treatment as the "control condition." We use this language even when we aren't talking about medical experiments or even experiments at all. We also refer to the state of the world we cannot observe as the "counterfactual" of the world we can observe — so the world where Shikha does not get the cancer drug is the *counterfactual condition* to the world where Shikha does get the drug.

It's at this point most people start throwing out "but what about..."'s, and that's good! You should be — that's exactly the kind of thinking you have to do when trying to answer Causal Questions. For example, "what about if we measured the size of Shikha's tumor before she took the drug and compared it to the size of her tumor after? If the tumor got smaller as soon as she started the drug, then surely the drug caused the tumor to shrink!"

Maybe! Implicitly, what you have done is asserted that you think that the size of Shikha's tumor before we administered the drug is a good approximation for what you think the size of Shikha's tumor *would have been* had we not given her the drug.

But this type of comparison will always fall short of the Platonic ideal given by our definition of causality. Yes, Shikha's tumor *may* have stayed the same size if we had not given her the drug (in which case the size of the tumor before she took the drug would be a good approximation), but it is also possible that regardless of whether we'd given her the drug, her cancer would have shrunk on its own.[1]

According to the Counterfactual Model of Causality, we could only ever *know* if taking the drug caused a decrease in tumor size if we could both administer the drug and observe the tumor *and also observe a parallel world in which the same person at the same moment in time was not given the drug for comparison*. And since we can never see this parallel world — the *counterfactual* to the world we observe — the best we can do is come up with different, imperfect tricks for *approximating* what might have happened in this parallel world, like comparing the tumor size before and after we administer the drug, imperfect though that may be.

So does that mean we're doomed? Yes and no. Yes, it *does* mean that we're doomed to never be able to take the exact measurements that make it possible to directly answer a Causal Question. But no, that doesn't mean we can't do anything — in the coming weeks, we will learn about different strategies for approximating counterfactual conditions, and in each case we will learn about what *assumptions* must be true for our strategy to provide a valid answer to our Causal Question. By making the assumptions that underlie each empirical strategy explicit, we will then be able to evaluate the plausibility of these assumptions.

In the example of Shikha, for example, we know that our comparison of tumor size before taking the drug to tumor size after taking the drug is only valid if her tumor *would not have gotten smaller without the drug*. This is something we can't measure directly, but we can look to other patients with similar tumors, or the history of her tumor size to evaluate how often we see tumors get smaller at the rate observed after she took the drug. If it's very rare for these types of tumors to ever get smaller, than we can have more confidence that a decrease in tumor size was the result of the drug.

We are also sometimes in a position to be more proactive than our effort to answer Causal Questions. Rather than trying to make inferences from the world around us using what is termed "observational data" (data that was generated through a process we did not directly control, a

process we only "observe"), we can sometimes generate our own data through randomized experiments.

Randomized experiments — perhaps the most familiar tool for answering Causal Questions — are also just another way of approximating the unobservable counterfactual condition. In a randomized experiment — also known as "Randomized Control Trials (RCTs)", or "A/B Tests" whether you're hanging out with statisticians, doctors, or web developers — participants are assigned to either receive the treatment (the treatment group) or not (the control group) based on the flip of a coin, a roll of a die, or more commonly a random number generator on a computer. Provided we have enough participants, the Law of Large Numbers then promises that, *on average*, the people assigned to the control group will (probably) be "just like" the people assigned to the treatment group in every possible way (save being treated). Subject to a few other assumptions we'll discuss in great detail later, that means that the outcomes of the control group — being just like the treatment group *on average* — will be a good approximation of what *would* have happened to the treatment group in a world where they did not receive the treatment.

Randomized experiments are not a silver bullet, however. The validity of experimental comparisons still rests on a number of assumptions, many of which cannot be directly tested. For example, we can never be entirely sure that when we randomly assigned people to control and treatment groups, the process was truly random, or that we ended up with people who were similar in both groups (the law of large numbers only promises that getting similar groups becomes *more likely* as the size of the groups increases, not that it will happen with certainty!). Moreover, conducting a randomized experiment requires working in a context where the researcher can control everything, and that can sometimes generate results that may not generalize to the big messy world where you actually want to act.

So where does that leave us?

For many data scientists, this will feel *profoundly* dissatisfying. Many people come to data science because of the promise that it will provide direct answers to questions about the world using statistics. But because of the Fundamental Problem of Causal Inference, this will never be possible when answering Causal Questions. Rather, the job of a data scientist answering Causal Questions is a lot like the job of a detective trying to solve a crime — your task is to determine what *probably* happened at a crime scene. You can gather clues, collect forensic evidence, and interview suspects, all in an effort to come up with the *most likely* explanation for a crime. But no matter how

hard you try, you can't go back in time to witness the crime itself, so you will never be able to be entirely sure if you are right or not.

But just as we investigate and prosecute crimes despite our inability to ever be 100% certain an arrested suspect is guilty, so too must businesses and governments make decisions using the best available evidence, even when that evidence is imperfect. But it is our job, as data scientists, to help provide our stakeholders with the best available evidence, and also to help them understand the strength of the evidence we are able to provide.

Why Passive-Prediction Is Not Enough

At this point, it is worth pausing to reflect on a question it may not have occurred to you to ask above — if answering Causal Questions is usually about *predicting* what would happen if we were to act on the world in a certain way, then how/why is it different from answering the kind of Passive-Prediction Questions we discussed previously?

There are a number of different ways one can frame the answer to this question, but the one I like most for Data Scientists is that when answering a Passive-Predictive Question, we can usually achieve our goals simply by identifying *correlations* that we think are likely to persist into the future. For example, suppose we run the maintenance department for a rental car company. The fact that a car whose *Check Engine* light is on is a car that is likely to break down if it isn't taken to a mechanic is enough for us to identify cars in trouble! Obviously, the *Check Engine* light isn't *causing* the cars to break down, but it doesn't have to to be useful.

But when seeking to answer Causal Questions, we wish to go beyond just identifying cars in trouble, and instead predict what might happen to cars if we *chose to act* in different ways. This requires going beyond simple Passive-Prediction because, in choosing to act, we are asking about how things might turn out in a world where we are behaving differently than we are currently — in other words, we are no longer being passive.

Thus, in a sense, answering Causal Questions is therefore *always* an example of "out-of-sample extrapolation" or "out-of-sample prediction", because by definition we are saying we want to know what happens in a world where at least one major agent — us! — changes their behavior. And indeed, there's a very real sense in which that's what we *mean* by a causal relationship: a relationship between our actions and an outcome that would persist even if we change our behavior!

What's a situation where a correlation is sufficient for Passive Prediction but not answering a Causal Question? Well, let's go back to our example of the rental car maintenance manager — suppose rather than using *Check Engine* lights to identify cars that needed more attention, the manager decided to just cut the cables that run to all the *Check Engine* lights! After all, the cars that are breaking down all have their *Check Engine* light on, and the cars that don't have their *Check Engine* lights almost never break down! So why not just disable the *Check Engine* lights on all these cars so they stop breaking down?

Now that we've been clear about what we mean when we ask "does X cause Y?", we can now understand why this is a perfect example of why correlation does not always imply causation.

Fundamentally, the manager is asking "would cutting the cables to the *Check Engine* lights prevent our cars from breaking down?" For that to be true, we know that in an ideal universe, we would want to compare a car on the verge of breaking down that has its *Check Engine* light intact to that same car in a world where we cut the *Check Engine* light — then we can see if there is a difference in whether these cars break down!

But this is *not* the data our manager has turned to draw their conclusion — rather, they are comparing cars with their *Check Engine* lights on and cars without their *Check Engine* lights on. And it turns out that cars *without* their *Check Engine* lights on are not a good approximation for the cars *with* their *Check Engine* lights on because the cars without the light on are different from the cars with the light on in ways that matter for the likelihood of breaking down (they have engine problems!) *other* than the *Check Engine* light!

Depending on what classes you may have taken in the past, you may have heard these differences referred to as "confounders" or "omitted variables" — those are just different words or ways of talking about the same idea! Confounders or omitted variables are just different words for features that are different between the "treated" and "untreated" observations being examined that the untreated observations are bad approximations of the counter-factual condition for the treated observations!

Next Steps

In this reading, we learned — in an intuitive sense — why answering Causal Questions is inherently hard. But this explanation, while accurate, is a little informal to be rigorous. In the readings that follow, we will be introduced to the *Potential Outcomes Framework* — the formal statistical framework that underlies the Neyman-Rubin Counterfactual Model of Causality. This framework will

help us reason more systematically about how and when methods like randomized experiments, linear regression, matching, and differences-in-differences can help us answer Causal Questions.

But first, in the interest of not losing perspective on the forest for the trees, a discussion of *how* Causal Questions are used in practice.

---

[1] The fact that diseases naturally change over time on their own is known as a disease's "natural history."

The Potential Outcomes Framework

We've all heard the saying "correlation does not imply causation." And that's true, as far as it goes. But the saying would be more accurate (albeit less catchy) if it were "correlation does not *necessarily* imply causation." That's because while it is true that correlation does not always imply causation, it does under certain circumstances.

In this chapter, we will learn about the *Potential Outcomes Framework*. The Potential Outcomes Framework introduces mathematical notation to our discussion of parallel universes, and in doing so allows us to reason far more rigorously about causal effects.

Crucially, the Potential Outcomes Framework will also allow us to be very precise about what it is we are measuring when we measure a correlation, what we need to measure in order to answer a Causal Question, and what must be true for those two quantities to be the same.

The Potential Outcomes Set Up

The potential outcomes framework is built around the idea that we all have well-defined *potential* outcomes that would be observed if we were to be exposed to a treatment, and also potential outcomes that would be observed if we were not (i.e., we were exposed to a "control" condition). These potential outcomes are fixed and defined values within each of us. But, of course, the challenge is that we can only ever *observe* ourselves in one state of exposure.

Let's suppose that we are interested in the effect of some treatment T on an outcome Y for a population of individuals $i \in 1, 2, 3... N$.

For ease of exposition, we will begin by assuming that our treatment T is a *binary treatment*, meaning it can only take on two values: $T \in \{0, 1\}$. There's nothing special about binary treatments, but they simplify notation, and so we often use them when discussing the potential outcomes framework, though everything we discuss here generalizes to continuous treatments.

> **Warning**
>
> Although we use the terminology *treatment* and *control* — terms you have most often encountered in the context of clinical trials — there is no, I repeat, **no** notion of randomization here. We can read $T = 1$ as "with the causal factor of interest present" and $T = 0$ as "without the causal factor of interest present," where the reason for the presence or absence of the causal factor of interest is unknown.
>
> This is often a source of confusion, so it bears repeating. We call this a *treatment*, but in doing so we do not imply that the receipt of treatment came from a randomized application. Instead, we are simply saying that $T = 1$, for instance, indicates a state in which the causal factor of interest is present.

Corresponding to the two values of our treatment variable, we also have two values of our outcome variable — Y_i — for each individual. In particular, an individual i has both a potential outcome under the treatment condition (i's value of Y in a world where i receives the treatment):

$$Y_i^{T=1} \equiv Y_i^1$$

and a potential outcome under the control condition (i's value of Y in a world where i does not receive the treatment):

$$Y_i^{T=0} \equiv Y_i^0$$

We can think about these individual-level realizations of the random variable as the potential outcomes in different theoretical states of the world that we all carry within ourselves. We have a potential outcome for if the world is $T = 1$, and we have a potential outcome for if the world is $T = 0$. They both exist simultaneously in us, though we are only able to observe one.

The Observed Outcomes Set Up

Now that we have defined the potential outcomes, let's discuss the set up for *observed* outcomes. Where T was in effect the theoretical treatment condition that allowed us to define *potential* outcomes for a theoretically applied causal factor, we use D to refer to the *observed* receipt of either treatment or control.

$$D \in \{0, 1\}$$

As with T, whether D is equal to zero or one is not (necessarily) a function of randomization. We should read $D_i = 1$ as "observed with the causal factor of interest" and read $D_i = 0$ as "observed without the causal factor of interest."

Because D is observed, not just potential, it is not possible for an individual to *simultaneously* be observed in treatment and control. Therefore, for individual i

$$D_i = 0 \text{ or } D_i = 1$$

Either we have an individual observed with the causal factor of interest present, or without the causal of interest present, but not both.

It follows that, for a given value of D — that is, for the given observed presence of the causal factor of interest or not — we will only ever be able to *observe*

$$Y_i^0 \text{ if } D_i = 0$$

or

$$Y_i^1 \text{ if } D_i = 1$$

But for any individual i, we cannot observe both.

> **Note**
>
> Why D? Honestly, I'm not sure, but I *think* it's related to the term "dummy variable," which is another term for an indicator variable. When analyzing actual, observed data, one often finds a dummy variable in the data that indicates whether a given observation was exposed to treatment or not. Since that dummy is indicating the actual, observed treatment status of an observation, it (kind of) makes sense to call this variable D.

> **Warning**
>
> There is a natural tempatation to think of T and D symmetrically, given that both take on two values and are related to treatment exposure. Avoid this temptation.
>
> Where the potential outcome frameworks assumes all entities have outcomes defined under both conditions of T (Y_i^0 and Y_i^1), any given entity has only **one** value of D_i. In the world we live, each entity either did or did not experience the treatment.
>
> So while T relates to a pretty abstract notion of parallel worlds, D **just defines two different populations of entities in the data**.)

Defining The Causal Effect

D_i just defines two different populations of entities in the data.

Now that we have that notation out of the way, let's apply it to the definition of causation we introduced in our previous readings. We previously said that the *effect* of some treatment T on Y for an individual i is the difference in i's outcomes in a world where the individual receives the treatment and i's outcomes in a world where they do not. In our new notation, we can call this quantity δ_i, and write it as:

$$\delta_i = Y_i^1 - Y_i^0$$

That is, for individual i, the causal effect, defined as δ_i, is the difference between the individual *potential outcome* for the state of the world where the causal factor is present, minus the *potential outcome* for the state of the world where the causal factor is not present.

Of course, this is *clearly* a quantity we'll never be able to observe — no person is ever able to be *both* subject to the treatment and not be subject to the treatment. And so for reasons that will

become obvious later, we'll need to move from estimating an *individual* quantity to estimating a quantity for a *population*.

When we move from the individual to the population, the causal effect needs to be defined as some sort of summary statistic that incorporates the information from the realization of the random variables Y^T for *all* individuals.

The first quantity we will investigate is the Average Treatment Effect, abbreviated as the ATE. We can define this quantity as:

$$\begin{aligned} ATE &= \frac{1}{N} \sum_{i \in 1,2,3...N} \delta_i \\ &= \frac{1}{N} \sum_{i \in 1,2,3...N} Y_i^1 - Y_i^0 \end{aligned}$$

To avoid the constant need for summation notation, however, let's assume our data is a random sample from the population we care about and move to working with expectations ($E()$):

$$\begin{aligned} ATE &= E(\delta_i) \\ &= E(Y_i^1 - Y_i^0) \\ &= E(Y_i^1) - E(Y_i^0) \end{aligned}$$

(We can bring the expectation inside the parentheses in this last step because the expectation is a linear operator.)

The equation above is similar in look and interpretation to δ_i at the individual level further above. And the logic is exactly the same. We have simply moved from assessing the causal effect for an individual to assessing an *average* causal effect for a population, and thus taken the expected value of the individual-level causal effects for our entire population.

Why do we care about the Average Treatment Effect? While not perfect, it is our best guess for the average change in our outcome of interest we would observe if *everyone* in the population we are studying were subject to treatment. Thinking about authorizing a new prescription drug? You want to know the average effect it will have across all patients who take it. Thinking about launching a big advertising campaign? You want to know the average effect it will have on all the consumers who see it. ATE, in other words, is the quantity your stakeholder — who wants you to answer a

Causal Question because they want you to predict the effect of some action they're thinking of taking — wants to know.

What Is a Correlation?

Having established the quantity we *want* to measure — the Average Treatment Effect for a population, $ATE = E(Y_i^1) - E(Y_i^0)$ — let's pause for a moment to come at things from the other side.

We have established that ATE is what we *want* to measure. But the fact we will never be able to observe both Y^0 and Y^1 for our full populations — and thus we can never directly observe $E(Y_i^1) - E(Y_i^0)$ — remains true. So let's pause for a moment to ask: "What *can* we observe?"

The first thing we can observe is outcome Y_i^0 for all i that weren't actually subject to treatment (i.e., for whom $D_i = 0$). Given that, we can estimate:

$$E(Y_i^0|D_i = 0)$$

And we can observe Y_i^1 for all i that *were* actually subject to treatment (i.e., for whom $D_i = 1$). So we can also estimate:

$$E(Y_i^1|D_i = 1)$$

And if we estimate the difference between the outcomes for these two populations — those who were treated ($D_i = 1$) and those who were not ($D_i = 0$) — we get the following quantity:

$$E(Y_i^1|D_i = 1) - E(Y_i^0|D_i = 0)$$

What is this? Well, it turns out that, for a binary treatment, this is the correlation between being treated and our outcome Y. It is, in fact, the exact quantity you get from a linear regression of Y on D!

And this is where all this notational work gets interesting. Most statistics students throw up a slide at the end of any presentation that says "but of course correlation does not imply causation" and move on — invoking this saying like an incantation they hope will protect them from critical questions. But we don't have to be like most statistics students anymore.

Now that we've rigorously defined what we measure when we estimate an empirical correlation, and we know what need to measure in order to measure a causal effect, we can ask: when are these two equal? Because if we can detail the conditions under which these two quantities are the same, then we've identified the situations in which correlation *does* imply causation.

> **Note**
>
> If this type of conditioning notation (the $|D_i = 1$ inside expectation) feels foreign, just think of it as subsetting the population for which you are estimating average outcomes. Again, D is just a variable for differenting between the two populations in our data — those that were exposed to the treatment and those that were not.

Correlation and Causation

Let's begin by giving the empirical correlation between treatment and outcomes a name. As we're *trying* to estimate the Average Treatment Effect (ATE), we will use the convention of adding a "hat" to a variable to indicate it is an empirically estimated quantity (rather than the "true") quantity. Thus our correlation betweens:

$$\widehat{ATE} = E(Y_i^1|D_i = 1) - E(Y_i^0|D_i = 0)$$

So when does this equal $E(Y_i^1) - E(Y_i^0)$? Let's apply a few manipulations to see if we can get $\widehat{ATE}$ to look like ATE.

First, we use that classic math trick of adding and subtracting a carefully chosen term to an equation. This addition and subtraction offsets, so this is the same as adding 0 (and is thus allowed), but ends up being very helpful. In particular, we will add and subtract $E(Y_i^0|D_i = 1)$:

$$\widehat{ATE} = E(Y_i^1|D_i = 1) - E(Y_i^0|D_i = 0) + \underbrace{E(Y_i^0|D_i = 1) - E(Y_i^0|D_i = 1)}_{\text{(Add up to zero)}}$$

We can then shuffle these terms around to isolate two distinct quantities of interest: The Average Treatment Effect on the Treated (ATT), and Baseline Differences:

$$\begin{aligned}\widehat{ATE} &= E(Y_i^1|D_i=1) - E(Y_i^0|D_i=0) + E(Y_i^0|D_i=1) - E(Y_i^0|D_i=1) \\ &= \underbrace{E(Y_i^1|D_i=1) - E(Y_i^0|D_i=1)}_{ATT} + \underbrace{E(Y_i^0|D_i=1) - E(Y_i^0|D_i=0)}_{\text{Baseline Difference}}\end{aligned}$$

Baseline Differences

Let's actually start with the second of these quantities, *Baseline Differences*. In substantive terms, Baseline Differences is just the difference in our outcome Y that would exist between our two groups (who *actually did* receive the treatment ($D_i = 1$) and the people who didn't ($D_i = 0$)) *in a world where no one was treated*. That's why we call these "baseline" differences — they are differences in outcomes that we would observe even in a world that the treatment we are interested didn't exist.

While I will use the term *Baseline Differences* for this quantity in this book, it also corresponds to the concept of "confounders" — factors that create a difference in outcomes between the treated group and the untreated group that are not caused by the treatment.

Unsurprisingly, therefore, the first condition that must be met for our correlation to imply causation (for $\widehat{ATE}$ to equal ATE) is for there to be no baseline differences:

> **Note**
>
> **No Baseline Differences:**
>
> $$0 = E(Y_i^0|D_i=1) - E(Y_i^0|D_i=0) \tag{1}$$

When true, our equation for $\widehat{ATE}$ simplifies substantially from:

$$\widehat{ATE} = \underbrace{E(Y_i^1|D_i=1) - E(Y_i^0|D_i=1)}_{ATT} + \underbrace{E(Y_i^0|D_i=1) - E(Y_i^0|D_i=0)}_{\text{Baseline Difference}}$$

to

$$\widehat{ATE} = \underbrace{E(Y_i^1|D_i = 1) - E(Y_i^0|D_i = 1)}_{ATT}$$

So what, then, is ATT?

Average Treatment Effect on the Treated (ATT)

Perhaps the easiest way to understand ATT is to directly compare the formulas for ATT with ATE:

$$\begin{aligned} ATT =& E(Y_i^1|D_i = 1) - E(Y_i^0|D_i = 1) \\ ATE =& E(Y_i^1) - E(Y_i^0) \end{aligned}$$

As you can see, the only difference between the two terms is that ATT is just the ATE *for the population of entities that were actually subject to the treatment ($D_i = 1$).

So for our estimated correlation $\widehat{ATE}$ to be the same as the Average Treatment Effect ATE we are most interested in, it must be the case that $ATT = ATE$. When does that happen?

Well, as you may recall, the Average Treatment Effect is just an average effect of the treatment for everyone being studied. That means that if λ is the share of the population for which $D_i = 1$, then ATE is:

$$\begin{aligned} ATE =& E(Y_i^1) - E(Y_i^0) \\ =& \lambda \left(\underbrace{E(Y_i^1|D_i = 1) - E(Y_i^0|D_i = 1)}_{\text{ATT}} \right) + (1 - \lambda) \left(\underbrace{E(Y_i^1|D_i = 0) - E(Y_i^0|D_i = 0)}_{\text{Avg Treatment Effect on Untreated}} \right. \end{aligned}$$

So when does $ATT = ATE$? When $E(Y_i^1|D_i = 1) - E(Y_i^0|D_i = 1) = E(Y_i^1|D_i = 0) - E(Y_i^0|D_i = 0)$. Or, in substantive terms, when the average effect of our treatment is the same for the subpopulation we observe experiencing the treatment as it is for the subpopulation that we observe *not* experiencing the treatment:

Note

No Differential Treatment Effects:

$$\begin{aligned} ATT &= E(Y_i^1|D_i = 0) - E(Y_i^0|D_i = 0) \\ E(Y_i^1|D_i = 1) - E(Y_i^0|D_i = 1) &= E(Y_i^1|D_i = 0) - E(Y_i^0|D_i = 0) \end{aligned} \tag{2}$$

Note

What would it look like for different populations to respond differently to treatment, and why might that happen in the real world? Well, it turns out it happens a lot when you aren't doing randomized experiments.

Suppose you polled your classmates after an exam, and they *all* had headaches. Half of your classmates then choose for themselves to use acetaminophen (Tylenol) ($D_i = 1$), and immediately feel better. Should you infer, then, that the ATE of acetomenophen is 100%?

Well... probably not. While it sounds like the ATT may be 100%, it's worth asking *why* the other half of students *didn't* take acetaminophen. What if they didn't take acetaminophen because they knew from past experience that acetaminophen doesn't alleviate their headache pain? In other words, the students who didn't take acetaminophen ($D_i = 0$) chose to not take acetaminophen precisely because they already *knew* that its effect on them is 0. Then we have a clear situation where $ATT \neq ATE$.

Wrapping It Up

We started off this chapter by noting that correlation does not *necessarily* imply causation. But there are certain condition under which correlation *does* imply causation, and we now know what those conditions are.

First, for our observed correlation between treatment and our outcome of interest to imply a causal effect, it must be the fact that there are no Baseline Differences. If there are Baseline Differences between the treated and untreated, then we just don't have a way to know what portion of the difference between these groups is the effect of treatment and what differences were pre-existing.

(Some of you may be saying "But wait! Can't I use a regression to estimate what share of differences are down to observable differences?" Yes, and we'll discuss that soon, but the key word is "observable" — regression can help address Baseline Differences *if* we can measure the factors that give rise to those differences.)

If we don't have Baseline Differences, then our correlation is at least a good estimate of the Average Treatment Effect on the Treated (ATT) — that is, the causal effect of our treatment *on the subpopulation that we observed experiencing to treatment*. And there will be times that the best we can do is get a good estimate of ATT.

But if we really want to get a good estimate of the Average Treatment Effect for the *full* population we're studying — something we want if we want to be able to estimate the effect of our treatment if we roll it out to everyone — then we also need it to be the case that the people we observe receiving the treatment ($D_i = 1$) respond to the treatment in the same way as those that don't ($D_i = 0$).

Testability of Assumptions

Any now the big catch: we need two conditions to be true to be sure we are getting a good estimate of a causal effect:

$$\begin{aligned} 0 &= E(Y_i^0|D_i = 1) - E(Y_i^0|D_i = 0) \\ E(Y_i^1|D_i = 1) - E(Y_i^0|D_i = 1) &= E(Y_i^1|D_i = 0) - E(Y_i^0|D_i = 0) \end{aligned}$$

But neither of these are directly observable. In the case of No Baseline Differences, we can never directly observe $E(Y_i^0|D_i = 1)$. And in the case of No Differential Treatment Effects, neither $E(Y_i^0|D_i = 1)$ nor $E(Y_i^0|D_i = 1)$ are observable.

As a result, it is only through critical thinking and domain knowledge that we can do our best to *reason* about whether we believe these conditions are met in any given situation. And *that* is why causal inference is hard.

Using and Interpreting Indicator (Dummy)

Variables

We often gloss over indicator variables in our statistics courses, but not only are they (in my view) one of the most powerful tools in a data scientist's tool box, but I cannot tell you how much I see people struggle with *interpreting* indicator variables in their regressions. So in this tutorial, I'll try to give them the treatment they deserve, and hopefully by the end, you've have a firm understanding not only of how to use *and interpret* Indicator Variables.

What are indicator variables?

Indicator variables – sometimes also referred to as dummy variables, though I don't know why – are variables that take on only the value of 0 and 1, and are used to *indicate* whether a given observation belongs to a discrete category in a way that can be used in statistical models.

For example, indicator variables can be used to indicate if an survey respondent is a woman (if the variable is 1 for women, 0 otherwise) or a Democrat (if the variable is 1 for democrats, 0 otherwise). In addition, as discussed in more detail below, collections of indicator variables can also be used to code categorical variables that take on more than 2 variables using a method called "one-hot encoding). This allows use to work with variables that have many levels, like an individual's political party registration (which could be Democrat, Independent, or Republican).

The ONE thing that you must understand when using indicator variables:

When you put an indicator variable in a regression model, there are two things you must always keep in mind about interpreting the coefficients associated with the indicator variable:

1. The coefficient on an indicator variable is an estimate of the average **DIFFERENCE** in the dependent variable for the group identified by the indicator variable (after taking into account other variables in the regression) and
2. the **REFERENCE GROUP**, which is the set of observations for which the indicator variable is always zero.

If you always remember that the coefficient on an indicator variable is an estimate of a **DIFFERENCE** with respect to a **REFERENCE GROUP** (also sometimes referred to as the "omitted

category"), you're 90% of the way to understanding indicator variables.

I recognize this may feel obvious, but trust me: I've literally reviewed papers from tenured faculty at major Universities that get this wrong. This is something people get confused about constantly, so I promise it's worth this treatment.

OK, let's get concrete.

Indicator Variables with Two Category Variable

Let's start with a simple model in which we wish to predict voter turnout using data from North Carolina. Suppose we're interested in looking at how turnout varies by gender, which is dichotomous in the North Carolina voter file (obviously this is somewhat problematic given what we've come to know about gender, but in most datasets you'll find a dichotomous coding).

```
# Load data we'll use. Should work for anyone.
import pandas as pd
import numpy as np

pd.set_option("mode.copy_on_write", True)

voters = pd.read_csv(
    "https://raw.githubusercontent.com/nickeubank/"
    "css_tutorials/master/exercise_data/voter_turnout.csv"
)
voters.head()
```

	age	gender	voted	party	race
0	71	FEMALE	1	UNAFFILIATED	WHITE
1	47	MALE	1	UNAFFILIATED	WHITE
2	29	MALE	0	DEMOCRATIC	WHITE
3	60	MALE	1	REPUBLICAN	WHITE
4	84	MALE	0	DEMOCRATIC	WHITE

```
# Create a 0/1 variable for female.
voters["female"] = voters.gender == "FEMALE"
voters.head()
```

	age	gender	voted	party	race	female
0	71	FEMALE	1	UNAFFILIATED	WHITE	True
1	47	MALE	1	UNAFFILIATED	WHITE	False
2	29	MALE	0	DEMOCRATIC	WHITE	False
3	60	MALE	1	REPUBLICAN	WHITE	False
4	84	MALE	0	DEMOCRATIC	WHITE	False

```
import statsmodels.formula.api as smf

model = smf.ols("voted ~ female", voters).fit()
model.get_robustcov_results("HC3").summary()
```

OLS Regression Results

Dep. Variable:	voted	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	-0.000
Method:	Least Squares	**F-statistic:**	0.7416
Date:	Wed, 25 Dec 2024	**Prob (F-statistic):**	0.389
Time:	11:45:39	**Log-Likelihood:**	-5768.3
No. Observations:	9919	**AIC:**	1.154e+04
Df Residuals:	9917	**BIC:**	1.156e+04
Df Model:	1		
Covariance Type:	HC3		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	0.7461	0.007	113.927	0.000	0.733	0.759
female[T.True]	0.0075	0.009	0.861	0.389	-0.010	0.025

Omnibus:	1890.028	**Durbin-Watson:**	1.974
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2391.714
Skew:	-1.156	**Prob(JB):**	0.00
Kurtosis:	2.337	**Cond. No.**	2.77

Notes:
[1] Standard Errors are heteroscedasticity robust (HC3)

OK, so how do we interpret this coefficient of 0.0075 on female? As we said before, it is the average **DIFFERENCE** in the dependent variable (Whether the person votes) with respect to a **REFERENCE GROUP**. The reference group is *the group for whom the indicator is always equal to zero*, which in this case is the set of male voters.

So this says that women are 0.7% *more likely to vote (in North Carolina) then men*.

Now let's try a more interesting example: Democrats.

```
voters.party.value_counts()  # OK, note here that there are THREE party registratio
```

```
party
DEMOCRATIC      4426
REPUBLICAN      3365
UNAFFILIATED    2128
Name: count, dtype: int64
```

```
# So let's do the same thing as before for Democrats
voters["democrat"] = voters.party == "DEMOCRATIC"
voters.head()
```

	age	gender	voted	party	race	female	democrat
0	71	FEMALE	1	UNAFFILIATED	WHITE	True	False
1	47	MALE	1	UNAFFILIATED	WHITE	False	False
2	29	MALE	0	DEMOCRATIC	WHITE	False	True
3	60	MALE	1	REPUBLICAN	WHITE	False	False
4	84	MALE	0	DEMOCRATIC	WHITE	False	True

```
# Note that because we're estimating a linear probability
# model we need to use heteroskedastic robust
# standard errors.

model = smf.ols("voted ~ democrat", voters).fit()
model.get_robustcov_results("HC3").summary()
```

OLS Regression Results

Dep. Variable:	voted	**R-squared:**	0.004
Model:	OLS	**Adj. R-squared:**	0.004
Method:	Least Squares	**F-statistic:**	40.81
Date:	Wed, 25 Dec 2024	**Prob (F-statistic):**	1.76e-10
Time:	11:45:39	**Log-Likelihood:**	-5748.0
No. Observations:	9919	**AIC:**	1.150e+04

Df Residuals:	9917			**BIC:**	1.151e+04	
Df Model:	1					
Covariance Type:	HC3					
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	0.7754	0.006	137.665	0.000	0.764	0.786
democrat[T.True]	-0.0562	0.009	-6.388	0.000	-0.073	-0.039

Omnibus:	1865.360	**Durbin-Watson:**	1.973
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2361.772
Skew:	-1.149	**Prob(JB):**	0.00
Kurtosis:	2.343	**Cond. No.**	2.51

Notes:
[1] Standard Errors are heteroscedasticity robust (HC3)

So now how do we interpret this coefficient on Democrats (-0.056)? As before it's the average **DIFFERENCE** in the dependent variable between the indicated group (Democrats) and the reference group. But what's the reference group? Republicans?

No – the **reference group** or **omitted category** is anyone for whom the indicator variable is always zero – in this case, all non-Democrats, whether they're Republicans or Unaffiliated.

So this result says that Democrats are less likely to vote than non-Democrats, but NOT that they're less likely to vote than Republicans per se.

So how do we deal with multiple categories? With multiple indicator variables!

Indicator Variables for variables with more than 2 categories

To deal with categorical variables with more than 2 categories, we create indicator variables for all values of the variable *except one*. The one group for which we do not create an indicator variable will become the **reference group** for the regression. The choice of which value to make the reference category won't substantively change the results of the regression – for example, if you

also have a control for age, the coefficient on age will always be the same regardless of the reference group used – but it does influence how easily you can interpret the results of the regression.

This practice of creating a *collection* of indicator variables to encode a single categorical variable is what's called "one-hot encoding" by computer scientists / machine learning people.

Since we're interested in the difference in turnout between Democrats and Republicans, let's make Republicans the reference category, and make indicators for Democrats and Unaffiliated voters.

```
voters["unaffiliated"] = voters.party == "UNAFFILIATED"
voters.head()
```

	age	gender	voted	party	race	female	democrat	unaffiliated
0	71	FEMALE	1	UNAFFILIATED	WHITE	True	False	True
1	47	MALE	1	UNAFFILIATED	WHITE	False	False	True
2	29	MALE	0	DEMOCRATIC	WHITE	False	True	False
3	60	MALE	1	REPUBLICAN	WHITE	False	False	False
4	84	MALE	0	DEMOCRATIC	WHITE	False	True	False

```
model = smf.ols("voted ~ democrat + unaffiliated", voters).fit()
model.get_robustcov_results("HC3").summary()
```

OLS Regression Results

Dep. Variable:	voted	**R-squared:**	0.007
Model:	OLS	**Adj. R-squared:**	0.007
Method:	Least Squares	**F-statistic:**	35.67
Date:	Wed, 25 Dec 2024	**Prob (F-statistic):**	3.67e-16
Time:	11:45:39	**Log-Likelihood:**	-5735.2
No. Observations:	9919	**AIC:**	1.148e+04
Df Residuals:	9916	**BIC:**	1.150e+04
Df Model:	2		
Covariance Type:	HC3		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	0.7988	0.007	115.554	0.000	0.785	0.812
democrat[T.True]	-0.0797	0.010	-8.240	0.000	-0.099	-0.061
unaffiliated[T.True]	-0.0606	0.012	-5.143	0.000	-0.084	-0.037

Omnibus:	1854.446	**Durbin-Watson:**	1.973
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2341.346
Skew:	-1.144	**Prob(JB):**	0.00
Kurtosis:	2.344	**Cond. No.**	3.85

Notes:
[1] Standard Errors are heteroscedasticity robust (HC3)

(Note that we also could have used the syntax `smf.ols('voted ~ C(party)', voters)`, which will automatically convert your data into one-hot encodings, but then you don't get to pick the omitted category, and sometimes it's nice to be explicit.)

How do we interpret these results?

First, we see that the coefficient on `democrat` is -0.08. That means that the **DIFFERENCE** in turnout between Democrats and the reference group (here, Republicans) is 8%. So Democrats

have 8 percentage point lower turnout on average in this data than Republicans.

Second, we see that the coefficient on `unaffiliated` is -0.06. That means that the **DIFFERENCE** in turnout between Unaffiliated voters and the reference group (here, Republicans) is 6%. So Unaffiliated voters have 6 percent point lower turnout on average in this data than Republicans.

Moreover, the p-value on these indicator variables tells us if these differences are significant. And indeed, they show clearly that the difference between Democrats and Republicans, and the difference between Unaffiliated voters and Republicans are both significant.

But what about the difference between Democrats and Unaffiliated voters? Well, turns out the regression doesn't give us that directly. To get that, we have to do some additional math.

First, it's easy to estimate the difference in coefficients:

```
dem - unaffiliated = (dem - republican) - (unaffiliated - republican)
                   = -0.08 - -0.06
                   = -0.02
```

So in other words, Democrats have 2 percentage point lower turnout than Unaffiliated voters.

But is this difference statistically significant? For that we have to run a post-regression test. (In R, you can do these with the `car` library using the `LinearHypothesis` function)

```
model = smf.ols("voted ~ democrat + unaffiliated", voters).fit()
model = model.get_robustcov_results("HC3")

hypothesis = "democrat[T.True] = unaffiliated[T.True]"
model.t_test(hypothesis)
```

```
<class 'statsmodels.stats.contrast.ContrastResults'>
                             Test for Constraints
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
c0            -0.0191      0.012     -1.634      0.102      -0.042       0.004
==============================================================================
```

Voila – p-value of 0.1.

Wanna confirm it? let's change our reference group to `unaffiliated`. Then when we look at the coefficient on `democrat`, that will be the difference between Democrats and the new reference group (Unaffiliated voters)

```
voters["republican"] = voters.party == "REPUBLICAN"
model = smf.ols("voted ~ democrat + republican", voters).fit()
model.get_robustcov_results("HC3").summary()
```

OLS Regression Results

Dep. Variable:	voted	**R-squared:**	0.007
Model:	OLS	**Adj. R-squared:**	0.007
Method:	Least Squares	**F-statistic:**	35.67
Date:	Wed, 25 Dec 2024	**Prob (F-statistic):**	3.67e-16
Time:	11:45:39	**Log-Likelihood:**	-5735.2
No. Observations:	9919	**AIC:**	1.148e+04
Df Residuals:	9916	**BIC:**	1.150e+04
Df Model:	2		
Covariance Type:	HC3		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	0.7383	0.010	77.436	0.000	0.720	0.757
democrat[T.True]	-0.0191	0.012	-1.634	0.102	-0.042	0.004
republican[T.True]	0.0606	0.012	5.143	0.000	0.037	0.084

Omnibus:	1854.446	**Durbin-Watson:**	1.973
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2341.346
Skew:	-1.144	**Prob(JB):**	0.00
Kurtosis:	2.344	**Cond. No.**	4.60

Notes:

[1] Standard Errors are heteroscedasticity robust (HC3)

As we can now see, the coefficient on `democrat` (now the difference between Democrats and Unaffiliated voters) is exactly what we'd calculated above (-0.019) and has the same p-value we calculated previously (0.1).

This just goes to show that the choice of reference group doesn't change what's actually being estimated, *it just changes the interpretation of coefficients* and what statistics pop right out of the regression output, and which values require a little extra work to get.

Interactions with Constant Variables

Like regular Indicators, but for differences in SLOPE rather than differences in LEVELS!

Congratulations! You're a pro at indicator variables. Now we can turn to INTERACTIONS!

Interactions (at least when you interact an indicator variable with a continuous variable), just like a regular indicator variables, report **differences** between a group and the reference group. The difference is that instead of reporting the difference in **average value** of the dependent variable between the indicated group and the reference group, the coefficient on an interaction term is the average **DIFFERENCE** in the **SLOPE** associated with the continuous variable between the indicated group and the reference group.

Let's be concrete: let's suppose we think that turnout among men increases as they get older by a larger amount than for women. In other words, we think that turnout increases with age for both groups, but that there's a **DIFFERENCE** in the amount it increases with age.

To test this, we need to create some interaction terms. But first, a quick note: when doing interactions, it's critical to not only include all the interaction terms that interest you, **but also all the variables in the interaction as stand-alone variables.** So for this we want `age` interacted with `female`. But while the coefficient on that estimate is what we're interested in, to get the right results we also need to include just `age` and just `female`.

```
voters["age_x_female"] = voters.age * voters.female
model = smf.ols("voted ~ age + female + age_x_female", voters).fit()
model.get_robustcov_results("HC3").summary()
```

OLS Regression Results

Dep. Variable:	voted	**R-squared:**	0.001
Model:	OLS	**Adj. R-squared:**	0.001
Method:	Least Squares	**F-statistic:**	1.917
Date:	Wed, 25 Dec 2024	**Prob (F-statistic):**	0.124
Time:	11:45:39	**Log-Likelihood:**	-5764.7
No. Observations:	9919	**AIC:**	1.154e+04
Df Residuals:	9915	**BIC:**	1.157e+04
Df Model:	3		
Covariance Type:	HC3		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	0.6961	0.029	23.789	0.000	0.639	0.753
female[T.True]	0.0933	0.039	2.370	0.018	0.016	0.171
age	0.0009	0.000	1.759	0.079	-9.81e-05	0.002
age_x_female	-0.0015	0.001	-2.237	0.025	-0.003	-0.000

Omnibus:	1883.068	**Durbin-Watson:**	1.976
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2388.508
Skew:	-1.156	**Prob(JB):**	0.00
Kurtosis:	2.340	**Cond. No.**	643.

Notes:
[1] Standard Errors are heteroscedasticity robust (HC3)

The coefficient on our interaction is -0.0015. Does that mean that as women get older, their turnout rate declines by -0.15 percentage points per year? **NO!**

It says that however turnout varies with age for men, turnout will vary with age by 0.0015 *less* for women. It is the **DIFFERENCE** in slopes between the two groups.

The coefficient on `age` tells us how turnout varies with age **for the reference group**. So it says that for men, turnout increases by 0.09 percentage points per year.

But if you want to know how women's turnout varies with age, you have to **ADD** the coefficient on `age` (the rate of change for me) plus the coefficient on `age_x_female` (the difference between the rate of men and women).

So going through all these coefficients, we have:

- `female` (0.09): Controlling for age, women are 9 percentage points more likely to vote than men.
- `age` (0.0009): As men get one year older, they become 0.09 percentage points more likely to vote.
- `age_x_female` (-0.0015): As women get one year older, the likelihood they vote increases by 0.1 percentage point per year *less* than men's likelihood of voting increases per year.

So how much does women's turnout increase if they age one year? 0.0009 + -0.0015 = -0.0006. So female turnout actually *declines* by 0.06 percentage points a year.

Now let's talk statistical significance. The p-value on age shows us that there's a statistically significant relationship between age and turnout **for men**. The p-value for `age_x_female` tells us that there's a statistically significant **difference** between men and women in how turnout varies with age. But is there a statistically significant relationship between age and turnout for women?

Again, we don't actually get an answer from our regression. To see, we have to run the following:

```
model = smf.ols("voted ~ age + female + age_x_female", voters).fit()
model = model.get_robustcov_results("HC3")
hypothesis = "age + age_x_female = 0"
model.t_test(hypothesis)
```

```
<class 'statsmodels.stats.contrast.ContrastResults'>
                            Test for Constraints
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
c0            -0.0006      0.000     -1.389      0.165      -0.001       0.000
==============================================================================
```

So the p-value is 0.17 for the relationship between female and age (and the coefficient of -0.0006, just like we calculated above!)

Keeping Things Straight

When you want to know a quantity, how can you figure out what coefficients to look at if you don't remember these rules?

The simplest way to make sure you're interpreting indicators correctly is to think about what our model looks like for different kinds of people. So suppose we wanted to figure out how men's turnout varies with age. Let's look at our model:

$$voted = \beta_0 + \beta_1 * age + \beta_2 * female + \beta_3 * (age * female) + \epsilon$$

Well, for men `female` and `age_x_female` will always be zero, so the model for men is actually just:

$$voted_{men} = \beta_0 + \beta_1 * age + \epsilon$$

And how does this vary with age? Linearly by β_1 per year.

What about for women? For women all those indicators will be 1s, so the equation will effectively be:

$$voted_{women} = \beta_0 + \beta_1 * age + \beta_2 + \beta_3 * age + \epsilon$$

$$voted_{women} = \beta_0 + (\beta_1 + \beta_3) * age + \beta_2 + \epsilon$$

(β_2 is on its own because the `female` is just an indicator that takes on a value of 1 in this case. You can think of it as having an implicit `1` next to it if that's helpful.)

And how does that vary with age? Linearly by $\beta_1 + \beta_3$ per year.

Finally, if we want the *difference* between how men and women respond to age, we can write this out:

$$voted_{women} - voted_{men} =$$

$$= (\beta_0 + \beta_1 * age + \beta_2 + \beta_3 * age + \epsilon) - (\beta_0 + \beta_1 * age + \epsilon)$$

$$= \beta_2 + \beta_3 * age$$

So the difference in who men and women respond to age in particular β_3.

The * operator

Finally, as with using `C()` to convert categoricals to indicators, you can also use the `*` notation in statsmodels for interactions – it not only creates the interaction term, but also adds all the level effects:

```
model = smf.ols("voted ~ age * female", voters).fit()
model.get_robustcov_results("HC3").summary()
```

OLS Regression Results

Dep. Variable:	voted	**R-squared:**	0.001
Model:	OLS	**Adj. R-squared:**	0.001
Method:	Least Squares	**F-statistic:**	1.917
Date:	Wed, 25 Dec 2024	**Prob (F-statistic):**	0.124
Time:	11:45:39	**Log-Likelihood:**	-5764.7
No. Observations:	9919	**AIC:**	1.154e+04
Df Residuals:	9915	**BIC:**	1.157e+04
Df Model:	3		
Covariance Type:	HC3		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	0.6961	0.029	23.789	0.000	0.639	0.753
female[T.True]	0.0933	0.039	2.370	0.018	0.016	0.171
age	0.0009	0.000	1.759	0.079	-9.81e-05	0.002
age:female[T.True]	-0.0015	0.001	-2.237	0.025	-0.003	-0.000

Omnibus:	1883.068	**Durbin-Watson:**	1.976
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2388.508
Skew:	-1.156	**Prob(JB):**	0.00
Kurtosis:	2.340	**Cond. No.**	643.

Notes:
[1] Standard Errors are heteroscedasticity robust (HC3)

Interactions Between Multiple Indicator Variables

You can also do interactions between Indicators, which have similar interpretations to interactions with constant variables.

For example, suppose instead of `age`, we just had a binary variable `old`:

```
voters["old"] = voters.age > 50

# Note the interaction has to be converted to integers first -- not booleans
voters["old_x_female"] = voters.old.astype("int") * voters.female.astype("int")
model = smf.ols("voted ~ old + female + old_x_female", voters).fit()
model.get_robustcov_results("HC3").summary()
```

OLS Regression Results

Dep. Variable:	voted	**R-squared:**	0.010
Model:	OLS	**Adj. R-squared:**	0.010
Method:	Least Squares	**F-statistic:**	31.53
Date:	Wed, 25 Dec 2024	**Prob (F-statistic):**	2.82e-20
Time:	11:45:39	**Log-Likelihood:**	-5717.1
No. Observations:	9919	**AIC:**	1.144e+04
Df Residuals:	9915	**BIC:**	1.147e+04
Df Model:	3		
Covariance Type:	HC3		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.6761	0.013	53.519	0.000	0.651	0.701
old[T.True]	0.1015	0.015	6.902	0.000	0.073	0.130
female[T.True]	0.0138	0.017	0.806	0.420	-0.020	0.047
old_x_female	-0.0111	0.020	-0.562	0.574	-0.050	0.028

Omnibus:	1820.991	**Durbin-Watson:**	1.973
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2319.326
Skew:	-1.140	**Prob(JB):**	0.00
Kurtosis:	2.357	**Cond. No.**	9.37

Notes:

[1] Standard Errors are heteroscedasticity robust (HC3)

Now we interpret `old_x_female` as the *difference* in how age affects men and woman. It is negative because, as we saw before, as women age, their turnout rate does not increase as much as men's. Here we find that men's turnout increases by 10 percentage points moving from under 50 to over 50. By contrast, when moving from under 50 to over 50, a women's turnout increases by -0.01 less (in total, it changes by 0.10 + -0.01 = 0.09).

(This may seem inconsistent with the results above, but that's because above we were modeling age linearly; in reality, the relationship between age and turnout is quadratic – it increases initially, peaks in middle-age, then declines, so neither of these models fit the data perfectly)

```
import seaborn.objects as so
from matplotlib import style
import matplotlib.pyplot as plt
import warnings

warnings.simplefilter(action="ignore", category=FutureWarning)

(
    so.Plot(voters, x="age", y="voted")
    .add(so.Line(), so.PolyFit(order=2))
    .label(title="Age and Turnout")
    .theme({**style.library["seaborn-v0_8-whitegrid"]})
)
```

Beyond The Experiment

The Role for Causal Inference with Observational Data in Industry

I don't think there's anyone who would question the importance of A/B testing in the tech sector today. It is used *constantly* to help companies refine their products, better target their ads, and incrementally innovate.

But while A/B testing is definitely a skill a good data scientist should have in their toolbox, it's important to understand that it is not the be-all and end-all of causal inference in industry. In this reading, we'll discuss *why* it's important to learn how to think critically about causality when working with observational data ("observational data" is data that we gather by passively observing the world – or that somebody else collected by passively observing the world – rather than data that we get by directly manipulating subjects in an experiment).

A/B Testing Isn't Always Feasible

The first reason it's important to know about tools that go beyond A/B testing is that A/B testing isn't always feasible. A/B testing only works when you can efficiently randomize people into different treatment arms, and where you can do it at low cost (if running a test is expensive, then even if you want to run the test, you should be doing some analysis first to make sure it's worth while!). But this isn't possible for every decision, even in big tech companies. Here's just a handful of situations where you can't run an A/B test:

- **You Don't Control The Relevant User Behavior:** Suppose you want to know the effect of, say, following a specific person on facebook or twitter on user satisfaction, or choosing to use a certain function in your product. The "ideal experiment" would be to randomly assign some users to follow the person in question / use your new function, and block others from doing so. But of course you can't do that! So you have to find other ways to compare followers and non-followers that accounts for ways they may be different in terms of user satisfaction or behavior.
- **It's a One-Shot Deal:** Suppose you're trying to decide whether to run a Super Bowl ad, or open a second store. You can't A/B test these small-N events. When dealing with these types of "big strategic" decisions, the best you can do is look to your companies past experiences /

the experiences of other companies that you think are comparable using the best available tools for observational causal inference.

- **Randomization is Bad Business:** It's one thing to randomize someone's landing page color scheme; it's another thing entirely to randomly show people different prices for your products – customers wouldn't stand for it! Similarly, sometimes you want a big rollout – like the launch of a new title on Netflix, or the premier of a new movie – and an A/B test would interfere with that. (Not an abstract example: here's Netflix talking about this constraint).
- **Time Horizons are Too Long:** A/B testing is popular in tech because you get so much data from users so quickly that they're easy to run *on short time horizons*. But if you're thinking about opening more stores across the country, you can't put your business on hold, pick a few locations, setup new shops, and wait for sales data; you need to get a sense of what's going to happen *now*. As we read in Kohavi, Tang and Xu, A/B tests work best when we have an Overall Evaluation Criterion (OEC) that is "measurable in the short term (the duration of an experiment) yet believed to causally drive long-term strategic objectives." But when that is not available / the shortest experiment available that would allow you to measure a meaningful OEC would take too long, you need other options!

A/B Testing Isn't Always Legal or Ethical

Those are all practical business constraints, but there are also ethical and legal constraints many people face.

For example, if you're at a social media company and want to study the effects of harassment on customer retention, you can't randomly assign people to be harassed!

And even if *you* think certain experiments are ok, you also have to be mindful of what the public will say. In 2012, Facebook ran an experiment in which they manipulated the emotional valence of posts they saw in their feeds, showing them either more happy posts or more sad posts. The goal was to see how moods spilled over from one user to another on the network.

The experiment was *legal*, but when people found out they were having the moods toyed with at random, they were furious. Now some tech people would say that "everything we do is to try and manipulate consumer happiness, how is this any different? And maybe you even believe that (though I think that, at the very least, deliberately showing people sad things means you're doing harm, which is a huge no-no in human experimentation). But it doesn't really matter, because it cost Facebook a lot of good will.

Similar issues happened with OKCupid when they published research on how they manipulated people's dating profiles. That, honestly, was a pretty standard experiment, but when it comes to messing with people's hearts... you have to be careful.

Finally, A/B randomizations aren't always *legal*. For example, suppose you're working in Human Resources for a company that wants to improve retention of younger tech wokers, and you want to offer childcare subsidies to employees. If that workforce is unionized, you *can't* offer a benefit to only some people within a specific class of employees due to union contracts.

Going Beyond Testing for the Future

Another limitation of A/B tests is that we usually use them to do small scale tests to motivate larger, future rollouts. But sometimes we want to just measure the effect of a single, large intervention.

In many areas of business, it's increasingly common to hire outside companies to do very complicated projects, like administering government programs or providing a service to employees. In these situations, companies have increasingly found that paying vendors based on inputs (hours worked, money spent) doesn't actually encourage vendors to do a very good job, or to try to be efficient – if you're being paid on the basis of what you spend, you will obviously you will not have an incentive to cut costs!

In these situations, companies are increasingly turning to something called "at-risk contracting." In these arrangements, the company doing the hiring offers to pay the vendor if they achieve a specific outcome. For example, a hospital might hire a company to reduce in-hospital infections and pay them based on reductions in infections, or a company may hire a vendor to minimize the downtime of their internal network, and pay them based on downtime. By tying compensation to the outcome that the hiring party actually cares about, the vendors well incentivized to do precisely what the hiring company wants them to do!

But in these arrangements, it's critical to both parties that there's a good way to measure the effect *of the vendor's efforts* or someone will get over or under paid. That's not an average effect, that's a specific effect, and something for which experiments are not well-suited.

In these circumstances, it's unfortunately the case that people often just measure outcomes before the vendor starts and compare them to outcomes when the vendor finishes. But if the outcome in question is subject to natural variation – such as seasonality – then we expect the outcome to vary

whether the vendor does anything or not, and so you need a better causal design (e.g. difference-in-difference) to take those types of confounders into account.

Don't want to be limited

Here's another important thought: professionally, if you only know how to do A/B testing, you risk limiting your opportunities for upward advancement. Big companies are getting better and better at providing analysts with extremely user friendly A/B testing tools, which means doing A/B tests is getting easier and easier. Twitter, for example, has an internal tool called Duck, Duck, Goose that makes running an A/B test a relatively low-difficulty task. And while running a good A/B test is never easy (as we have discussed at length, even with A/B testing, the internal and external validity of our causal estimates will always be dependent on a wide range of assumptions), it is something that more and more people are learning to do well and which companies are learning to make easier. Thus, if all you know how to do are A/B tests, you may find it increasingly hard to differentiate yourself in the data science market.

Again, this isn't to say you shouldn't learn to do A/B testing, just don't *only* learn A/B testing.

Not enough to be able to do it; you must be able to *explain it*

One final reason it's important to understand how to apply the logic of causal inference to observational data is that even if *you* only work with A/B data, it is almost inevitable that you will interact with other people in your company who want to use observational data. In these situations, it's not enough that *you* understand the considerations that go into making causal inferences from observational data; you also need to be able to explain those to colleagues.

Fixed Effects and Causal Inference

So, you're working with panel data. You have either multiple observations per entity (usually data over time), or you have nested data (data on individuals, each of whom belongs to a larger group, like kids in a school). And you want to add fixed effects for the entities or groups. But what does that mean from the perspective of causal inference?

The simple answer is that a fixed effect is a tool for controlling for a specific source of baseline differences (namely, baseline differences across entities). In that sense, fixed effects are no different from any other type of control variable we put in a regression.

But there is an important difference: with most variables we put in a regression, those variables are only controlling for a specific thing, the thing they measure. If you add age, you're controlling for age; add income, you're controlling for income. But fixed effects are a little more powerful: they control for *any* stable source of baseline differences between entities!

Uh, what?

Suppose you have weekly sales data for all your stores over several years, and you're interested in better understanding the effect of each store's local unemployment rate on sales. For simplicity, we'll assume that in any week, a given store is said to experience high unemployment ($D_i = 1$) or low unemployment ($D_i = 0$), though this extends to using thinking of unemployment as a continuous variable.

If you just ran a regression of local unemployment rates on sales, you might find that sales are extremely sensitive to the local unemployment rate. But when you dig in, that's because your New York City store in Times Square makes tons of sales, and NYC tends to always have low unemployment.

In our usual causal framework, we'd say that our NYC store had important *baseline differences* in our outcome (sales per week) that happened to be correlated with treatment assignment (NY tends to have low unemployment), though we don't think it's because of NYC has low unemployment (we think it's because the store is in Times Square!). As a result, $E(Y_i^0|D_i = 0) \neq E(Y_i^0|D_i = 1)$. We have a problem with baseline differences.

But what if you add store fixed effects to the model? You can think of those fixed effects as *demeaning* all sales *for each store*. In other words, you're removing *level* differences across scores from the relationship you're trying to estimate. As a result, instead of estimating how differences in unemployment are correlated with sales, you're instead estimating how *changes* in unemployment relate to *changes* in sales.

That's because after fixed effects effectively demean your dependent variable (for each store), the only variation in the data you're left with are the deviations in your dependent variable above or below its mean value for a given store. We're no longer estimating the relationship between

unemployment and store sales, but rather how *changes* in unemployment are correlated with *changes* in sales.

From a causal perspective, now that our focus is on how *changes* in unemployment affect *changes* in sales, any differences between stores that don't vary over time – no matter what caused those differences – have been accounted for.

Mathematically, we can think of this as a change in our Y: instead of Y_i being absolute sales, it's now deviations in sales above or below the average sales for a given store i. Thus, baseline differences ($E(Y_i^0|D_i = 0) = E(Y_i^0|D_i = 1)$) has become a question about whether, in a world where unemployment is below the local average, sales for all groups tend to be the same amount above (or below) store average sales. And the fact that we have a store in Times Square is no longer an obvious threat to that condition!

The limit to this magic is that we're only controlling for *non-time-varying* differences between stores. If store sales in NYC are always high because of their location, that's something controlled for by the fixed effects. However, if in the middle of your data NYC experiences a hurricane, and so sales dip, that isn't taken into account by your fixed effects (since it was "time-varying," it only affected the store for a fixed period-of-time). As a result, you'd want a separate control for those kinds of disruptions in your regression.

One way you can see this is that if you try and stick a variable into your regression that is not time-varying (it has the same value for each store throughout the data), you'll find that you don't get back a regression coefficient. That's because anything non-time varying is actually co-linear (in terms of the underlying linear algebra) with the fixed effects, meaning you literally *can't* estimate a coefficient. It'd be like trying to include the variable "age" twice in the same regression.

Fixed Effects: Indicator Variables for Groups

One common use of indicator variables are as *fixed effects*. Fixed effects are used when our data as a "nested" structure (we think individual observations belong to groups), and we suspect different things may be happening in each group.

For example, suppose we have a dataset of student test scores, and students are all grouped into different schools; or perhaps we have data on earnings and gender across US cities. In these examples, individual observations can be thought of as being grouped into schools or cities.

One option with this kind of data is to just ignore the groups. For example, if we want to know about differences in the academic performance of minority children across the school system, then we might not want to add controls for students' schools because we think that part of way race impacts performance is though sorting of minority students into worse schools. If we added school fixed effects, we'd lose that variation.

But suppose we were interested in understanding whether school administrators treat minority children differently, and whether this affects academic performance. Principles, for example, may be more likely to suspect Black children than White children. If that were our interest, then what we really want to know about is how race impacts academic performance *among students in the same school*. And that's where fixed effects are useful – they let us control for group-level effects (like the fact all children in one school might tend to get lower grades) so we can focus on explaining *intra-group* variation (differences among children *at the same school*).

In this regard, fixed effects are analogous in purpose to hierarchical models, though they are slightly different in implementation (differences between fixed effects and hierarchical models are discussed here).

Implementing Fixed Effects

To illustrate, let's try and estimate how gender impacts earnings in the US using data from the US Current Population Survey (CPS) on US wages in 2019. We'll begin with a simple model of earnings:

```
import pandas as pd

# Load survey
cps = pd.read_stata(
    "https://github.com/nickeubank/MIDS_Data/blob/"
    "master/Current_Population_Survey/morg18.dta?raw=true"
)

# Limit to people currently employed and working full time.
cps = cps[cps.lfsr94 == "Employed-At Work"]
cps = cps[cps.uhourse >= 35]

# Annual earnings from weekly
cps["annual_earnings"] = cps["earnwke"] * 48

# And create gender and college educ variable
cps["female"] = (cps.sex == 2).astype("int")
cps["has_college_educ"] = (cps.grade92 > 43).astype("int")
```

```
import statsmodels.formula.api as smf

smf.ols("annual_earnings ~ female + age + has_college_educ", cps).fit().summary()
```

OLS Regression Results

Dep. Variable:	annual_earnings	**R-squared:**	0.170
Model:	OLS	**Adj. R-squared:**	0.170
Method:	Least Squares	**F-statistic:**	8393.
Date:	Wed, 25 Dec 2024	**Prob (F-statistic):**	0.00
Time:	11:45:54	**Log-Likelihood:**	-1.4365e+06
No. Observations:	122603	**AIC:**	2.873e+06
Df Residuals:	122599	**BIC:**	2.873e+06
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	3.639e+04	296.455	122.766	0.000	3.58e+04	3.7e+04
female	-1.168e+04	170.390	-68.540	0.000	-1.2e+04	-1.13e+04
age	407.7144	6.406	63.650	0.000	395.160	420.269
has_college_educ	3.054e+04	239.482	127.513	0.000	3.01e+04	3.1e+04

Omnibus:	18030.472	**Durbin-Watson:**	1.805
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	27634.350
Skew:	1.059	**Prob(JB):**	0.00
Kurtosis:	3.963	**Cond. No.**	159.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In this model, we're getting estimates of how education and gender explain variation across all Americans.

But in this dataset, we also have a variable that tells us the industry in which each respondent is employed. If we want to understand the relationship between gender and income through *both* workplace bias and sectoral sorting, we can use the model above. But suppose we want to estimate wage discrimination in the workplace after controlling for the industry into which someone chooses to work. In other words, we want to know about the impact of gender on wages *within industries*.

To do so, we can add an indicator for each respondent's industry (in the `ind02` variable):

Then we can run the following regression:

```
smf.ols(
    "annual_earnings ~ female + age + has_college_educ + C(ind02)", cps
).fit().summary()
```

which will generate output that will look approximately like this (note your output will be VERY long—I'm omitting all the industry coefficients for space. We'll talk later about how to suppress those in your output):

OLS Regression Results

Dep. Variable:	annual_earnings	R-squared:	0.256
Model:	OLS	Adj. R-squared:	0.254
Method:	Least Squares	F-statistic:	161.1
Date:	Sun, 19 Feb 2023	Prob (F-statistic):	0.00
Time:	10:04:15	Log-Likelihood:	-1.4299e+06
No. Observations:	122603	AIC:	2.860e+06
Df Residuals:	122341	BIC:	2.863e+06
Df Model:	261		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.27e+04	1156.364	19.629	0.000	2.04e+04	2.5e+04
C(ind02)[T.Animal production (112)]	-667.2345	1766.161	-0.378	0.706	-4128.882	2794.413
C(ind02)[T.Forestry except logging (1131, 1132)]	1.249e+04	3861.556	3.235	0.001	4922.899	2.01e+04

```
.
.
.
```

		C(ind02)[T.Armed Forces]	-1.397e-12	6.68e-13	-2.090	0.037	-2.71e-12	-8.71e-14
		female	-1.065e+04	182.433	-58.384	0.000	-1.1e+04	-1.03e+04
		age	386.8119	6.176	62.633	0.000	374.707	398.917
		has_college_educ	2.665e+04	245.818	108.407	0.000	2.62e+04	2.71e+04
Omnibus:	17539.109	Durbin-Watson:	1.844					
Prob(Omnibus):	0.000	Jarque-Bera (JB):	27650.331					
Skew:	1.004	Prob(JB):	0.00					
Kurtosis:	4.176	Cond. No.	1.05e+16					

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.24e-24. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

Voilà! What you've just estimated is no longer the relationship between gender and income across all Americans, but rather the relationship between gender and income *within each industry*.

To be clear, fixed effects aren't *mathematically* different from adding a normal control variable. One could say that adding `has_college_educ` means that we're now estimating the relationship between gender and income among college educated and among non-college educated. *Mechanically*, fixed effects are just additional indicator variables. But because we often use them for groups, thinking about the fact that, when added, one is effectively estimating variation *within* the groups specified by the fixed effects is a powerful idea.

Perhaps no place is this more clear than in full panel data, where you have data on the same entities over time. In a panel regression, the addition of entity fixed effects allow you to difference out any *constant* differences between entities, and focus only on changes within each entity over time. This even works for people! In a panel with individuals observed over time, adding individual fixed effects means you're effectively controlling for anything constant about each individual (things that don't change over time), and now you're just studying *changes over time* for each individual.

Clustering

When working with fixed effects, however, it's also often a good idea to cluster your standard errors by your fixed effect variable. Clustering is a method for taking into account some of the variation in your data isn't coming from the individual level (where you have lots of observations), but rather from the group level. Since you have fewer groups than observations, clustering corrects your standard errors to reflect the smaller effective sample size being used to estimate those fixed

effects (clustering *only* affects standard errors – it has no impact on coefficients themselves. This is just about adjustments to our confidence in our inferences).

Clustering is thankfully easy to do—just use the `get_robustcov_results` method from `statsmodels`, and use the `groups` keyword to pass the group assignments for each observation.

(R users: as we'll discuss below, I think the easiest way to do this is to use the plm package.)

TWO IMPORTANT IMPLEMENTATION NOTES:

1. First, if you're using formulas in statsmodels, the regression is automatically dropping observations that can't be estimated because of missing data, so you have to do the same before passing your group assignments to-`get_robustcov_results`—otherwise you'll get the error:

```
ValueError: The weights and list don't have the same length.
```

because the number of observations in the model doesn't match the number of observations in the group assignment vector you pass!

2. Whatever you pass to `groups` has to be a numeric array of group identifiers. If you don't, you'll get an error like:

```
TypeError: '<' not supported between instances of 'float' and 'str'
```

```
model = smf.ols(
    "annual_earnings ~ female + age + has_college_educ + C(ind02)", cps
).fit()

# Drop any entries with missing data from the model
fe_groups = cps.copy()
for i in ["annual_earnings", "female", "age", "ind02", "has_college_educ"]:
    fe_groups = fe_groups[pd.notnull(fe_groups[i])]

# Convert `ind02` categorical into group codes by
# pulling codes used in its categorical encoding.

# If you have a string instead of a categorical,
# just make it a categorical first with `pd.Categorical()`
group_codes = fe_groups.ind02.cat.codes
group_codes.head(5)
```

```
2     222
3     201
4     220
6     158
17    141
dtype: int16
```

```
model.get_robustcov_results(cov_type="cluster", groups=group_codes).summary()
```

OLS Regression Results

Dep. Variable:	annual_earnings	R-squared:	0.256
Model:	OLS	Adj. R-squared:	0.254
Method:	Least Squares	F-statistic:	1144.
Date:	Sun, 19 Feb 2023	Prob (F-statistic):	1.16e-148
Time:	10:04:18	Log-Likelihood:	-1.4299e+06
No. Observations:	122603	AIC:	2.860e+06
Df Residuals:	122341	BIC:	2.863e+06
Df Model:	261		
Covariance Type:	cluster		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.27e+04	707.825	32.067	0.000	2.13e+04	2.41e+04
C(ind02)[T.Animal production (112)]	-667.2345	21.509	-31.021	0.000	-709.591	-624.878
C(ind02)[T.Forestry except logging (1131, 1132)]	1.249e+04	189.349	65.971	0.000	1.21e+04	1.29e+04
C(ind02)[T.Logging (1133)]	6521.2131	117.869	55.326	0.000	6289.105	6753.321

.
.
.

affairs (928)]			
C(ind02)[T.Armed Forces]	-1.397e-12	1.81e-14	-77.232	0.000	-1.43e-12	-1.36e-12
female	-1.065e+04	583.273	-18.261	0.000	-1.18e+04	-9502.628
age	386.8119	16.979	22.782	0.000	353.378	420.246
has_college_educ	2.665e+04	1327.797	20.070	0.000	2.4e+04	2.93e+04

Omnibus:	17539.109	Durbin-Watson:	1.844
Prob(Omnibus):	0.000	Jarque-Bera (JB):	27650.331
Skew:	1.004	Prob(JB):	0.00
Kurtosis:	4.176	Cond. No.	1.05e+16

Notes:
[1] Standard Errors are robust to cluster correlation (cluster)
[2] The smallest eigenvalue is 2.24e-24. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

As you can see, while our point estimates haven't changed at all (the coefficient on `female`, for example, is still ∼-10,650), we have increased the size of our standard errors. The SE on `female`, for example, has gone from 182 without clustering to 583 with clustering.

Computationally Efficient Fixed Effects

OK, so everything we've describe up till here is a reasonable approach to fixed effects, but it has two limitations: our regression output looks *terrible*, and computing all those intercepts was slow.

This brings us to some of the specialized methods for calculating fixed effects. It turns out that if you aren't interested in the coefficient on each fixed effect, there are much more computationally efficient methods of calculating fixed effects. But to use them, we'll have to use a different library: linearmodels (installable using `conda install linearmodels` or `pip install linearmodels`).

(R users: see note at bottom on doing this in R)

In particular, we'll be using the `PanelOLS` function from `linearmodels`. As the name implies, `PanelOLS` is designed for linear regression (social scientists call linear regression Ordinary Least Squares, or OLS) with panel data, which is really any form of data organized along two dimensions. Normally a panel has data on many entities observed several times, so the first dimension is the `entity` dimension, and the second is the `time` dimension.

In this case, we don't really have a panel—just nested data—but because fixed effects are commonly used in panels, we'll use this tool.

The only catch is: you have to use `multiindexes` in `pandas`. I *know*, I hate them too. But the multi-index is required by the library for it to understand what variable constitutes the "group" for which you want to add fixed effects. Basically `PanelOLS` calls the first level of the multi-index the `entity` and the second level `time`. In this case, though, we'll just make the first level our counties, and the second level individual identifiers, then use `entity` fixed effects (and clustering).

```
cps.head()
```

	hhid	intmonth	hurespli	hrhtype	minsamp	hrlonglk	hrsample
2	000110339935453	January	1.0	Unmarried civilian female primary fam householder	MIS 4	MIS 2-4 Or MIS 6-8 (link To	0701
3	000110339935453	January	1.0	Unmarried civilian female primary fam householder	MIS 4	MIS 2-4 Or MIS 6-8 (link To	0701
4	000110359424339	January	1.0	Unmarried civilian female primary fam householder	MIS 4	MIS 2-4 Or MIS 6-8 (link To	0711
6	000110651278174	January	1.0	Civilian male primary individual	MIS 8	MIS 2-4 Or MIS 6-8 (link To	0601
17	007680515071194	January	1.0	Civilian male primary individual	MIS 8	MIS 2-4 Or MIS 6-8 (link To	0611

5 rows × 101 columns

```
# Move county groups into highest level of multi-index,
# with old index in second level.
# PanelOLS will then see the first level as the `entity`
# identifier.
cps_w_multiindex = cps.set_index(["ind02", cps.index])
cps_w_multiindex.head()
```

		hhid	intmonth	hurespli	hrhtype	minsamp	hr
ind02							
Residential care facilities, without nursing (6232, 6233, 6239)	**2**	000110339935453	January	1.0	Unmarried civilian female primary fam householder	MIS 4	M (6-
Business support services (5614)	**3**	000110339935453	January	1.0	Unmarried civilian female primary fam householder	MIS 4	M (6-
Hospitals (622)	**4**	000110359424339	January	1.0	Unmarried civilian female primary fam householder	MIS 4	M (6-
Truck transportation (484)	**6**	000110651278174	January	1.0	Civilian male primary individual	MIS 8	M (6-
******Department stores and discount stores (s45211)**	**17**	007680515071194	January	1.0	Civilian male primary individual	MIS 8	M (6-

5 rows × 100 columns

```
from linearmodels import PanelOLS

mod = PanelOLS.from_formula(
    "annual_earnings ~ 1 + female + age + has_college_educ + EntityEffects",
    data=cps_w_multiindex,
)
mod.fit(cov_type="clustered", cluster_entity=True)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[6], line 1
----> 1 from linearmodels import PanelOLS
      3 mod = PanelOLS.from_formula(
      4     "annual_earnings ~ 1 + female + age + has_college_educ + EntityEffects",
      5     data=cps_w_multiindex,
      6 )
      7 mod.fit(cov_type="clustered", cluster_entity=True)

ModuleNotFoundError: No module named 'linearmodels'
```

Fixed Effects and Hierarchical Models

It is often the case that the data you will work with as a data scientist will have a *nested* structure, meaning our individual observations can also be divided into groups. In a dataset on student test scores, for example, individual students can be grouped by classroom, or by school. In data on store revenues, it may be possible to group stores by the city in which they are located.

When this happens, we often want to take this group structure into account. For students, for example, we might want to take into account that all the students in a good school may do better than average (because the school is good), or that all the stores in a poor city may be likely to under-perform (not because of anything the store has done, but because the city is poor).

There are several strategies for making these adjustments. The one the is most popular among statisticians (and which is most familiar to MIDS students) are hierarchical models. Hierarchical models accomplish two things: (a) they allow for different intercepts (or different slopes) for all observations within a group, and (b) they adjust our standard errors to account for the possibility of group-level shocks (if the 300 kids in a single school are all over-performing, hierarchical models don't think of that as 300 observations when calculating standard errors for group effects (which would generally result in small standard errors since we have a sample size of 300 kids); it sees it as a single observation of a good school (with larger standard errors due to the fact we only have a sample size of one school).

Hierarchical models are popular in many settings, and for good reason – they are statistically *efficient* (meaning they generate estimates that tend to standard errors that are smaller than what you get from other models designed to accomplish the same things). But they also have limitations.

The chief limitation of hierarchical models is that they assume that the magnitudes of group difference (i.e. the group intercepts or differences in slope) are uncorrelated with the other explanatory variables in the model. Indeed, it the fact that they can leverage this assumption to make precise estimates that makes them statistically efficient *if that assumption is true*.

But there are many settings where this assumption breaks down. For example, suppose we're estimating the test scores of children, and we're controlling for whether children are low-income. If we estimate a hierarchical model to allow for different schools to have different average test scores, then those estimates of school-level differences will be biased if low-income students tend to go to under-performing school (because our explanatory variable – a child's income – is correlated with the school's average performance).

With that in mind, social scientists (who work with data where *almost everything* is extremely correlated) often prefer a different strategy: fixed effects with clustered standard errors.

However, it's worth noting that the question of whether to use Fixed Effects or Hierarchical Models is one of htose situations where we *can* statistically evaluate which strategy is the right answer! Because we know that the fixed effects estimates are always *right* (even if they have larger standard errors than we might want), we can run our models using both a higherarchical model strategy *and* a fixed effect strategy. If they answers are statistically similar, then we're probably OK using the hierarhcical models. If they're substantially different, then we know we can't trust the hierarchical model, and should use the Fixed Effects. The test for this difference is called a "Hausman Test".

Fixed Effects & Clustered Standard Errors

Fixed effects are just like hierarchical group estimates, except they don't require that the level (or slope) differences of groups be uncorrelated with other explanatory variables. This does make them less statistically efficient when these differences are uncorrelated with other explanatory variables (you tend to get larger standard errors than in hierarchical models), but it also makes them more robust (they give you unbiased estimates whether a correlation with explanatory variables exists or not).

But fixed effects don't address the second purpose of hierarchical models – correcting your standard errors. For that, we use clustered standard errors. The idea of clustered standard errors is to account for the fact that if everyone in a given group has a high (or low) error, we want our standard errors to reflect the fact we shouldn't treat that like lots of *individual* high (or low) errors

(which would generate small standard errors because of the implied large sample size), but rather a *single* high (or low) error for the group (with an accompanyingly large standard error since really, there's only one observation of a group error common to all observations). As a result, we *generally* get larger standard errors when we cluster.

(Fixed effects are also often used to allow for different group intercepts, not it is less common to use them to allow for differential group slopes. This actually can be done, but less frequently).

I recognize this all feels hand-wavy. If you're interested in the math, you can find a more detailed mathematical discussion, but I'll confess that this is an area I've found found that math to offer much intuition. Technically, what we're doing is allowing for off-the-diagonal terms in our variance-covariance matrix for members of each group to allow for correlated shocks. But if that explanation helps you understand in principle what we're doing, you're smarter than me. :)

So if you're using fixed effects for groups that you think might be subject to common "shocks" (things you aren't directly measuring that affect the outcomes of the observations – like how all students in a school may see test scores change because of an especially good principal, or how all stores in a city may have low sales because the city is unusually poor) – you should cluster all the observations for that group (we'll discuss how to cluster in Python in the next execises).

Last note: hierarchical models are equivalent to what social scientists often call "random effects models with clustered standard errors." So just think "hierarchical models" when you hear "random effects."

Things To Remember:

- Everything you learned about when and how you can use hierarchical models applies to fixed effects + clustered standard errors, except that fixed effects + clustered standard errors are even more robust.
- If you're using fixed effects for groups that might be subject to common shocks, you should probably also be using clustered standard errors.
- Random effects + clustered standard errors is just a different term for hierarchical models, and often you'll just hear them called "random effects models" or "mixed effects models".

What Is Matching?

This week we'll be exploring *matching*: an approach to analyzing observational data that is primarily designed to do the same types of analyses as regression, but with less sensitivity to the functional form assumptions implicit to regression.

As we move forward, I'll be drawing a lot (including most figures) from Ho, Imai, King and Stuart (2007) and two of Gary King's public lectures on youtube (theory, implementation). I've streamlined his argument and tried to adapt it for the concepual framworks, language, and level of technical depth we're targeting in this class, but if you're really into this topic, that source material may be extremely valuable to explore.

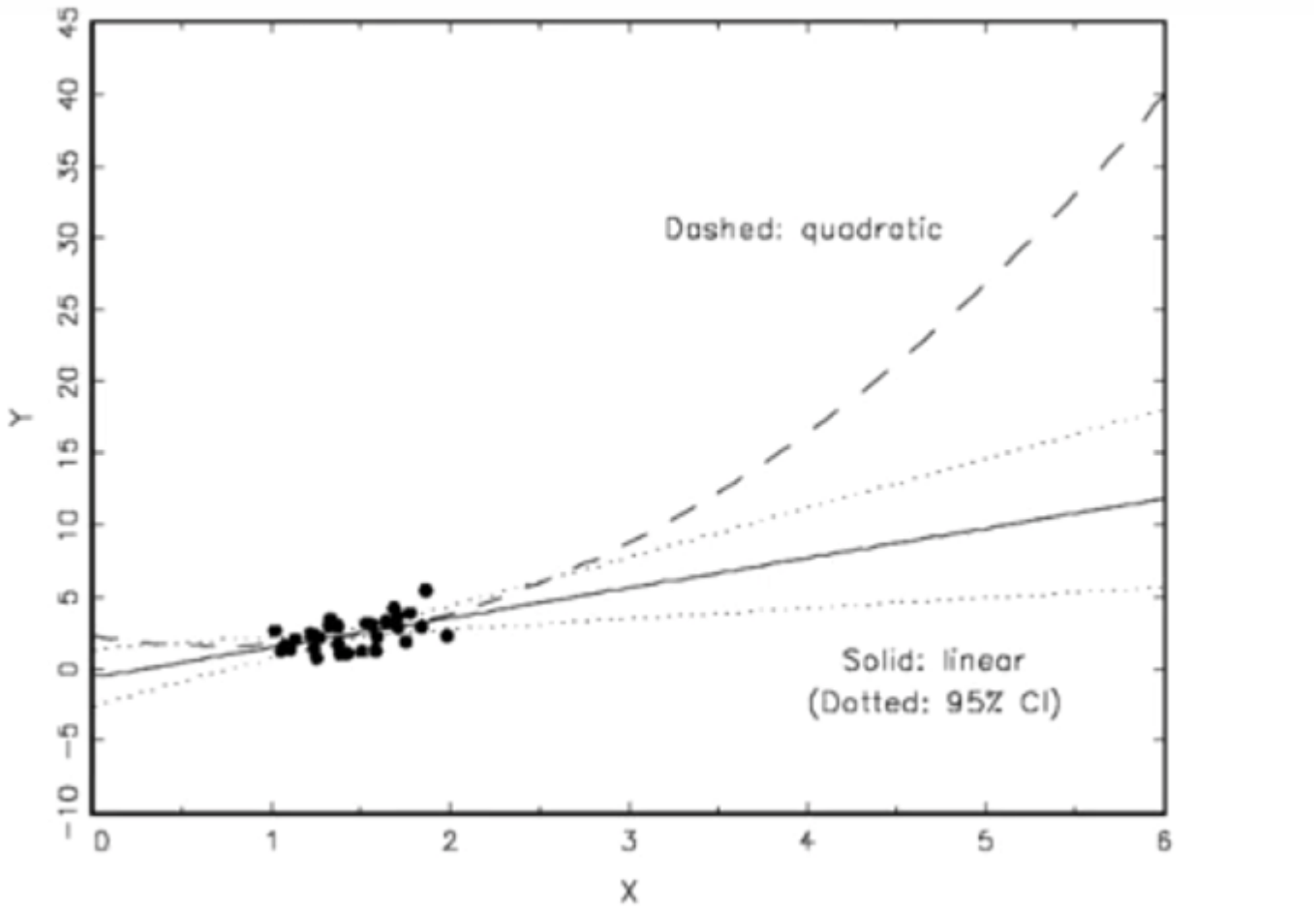
Model Dependency

To understand matching, it helps to start with the problem matching is designed to solved: model dependence.

The term model dependence refers to a situation in which the results of an analysis are highly sensitive to apparently small decisions made by the researcher in how they specify a regression – for example, whether they include X in their regression, or $X + X^2$, or whether they use X or $log(X)$ as an independent variable.

"But wait!" you may say: we have all sorts of test statistics for evaluating what fits our data best, so shouldn't we be able to just pick the specification with the best [insert test statistic of choice here]?

Yes... most of the time. But where you get into trouble is when you want to use a model fit on data that covers a certain range of values to make predictions about outcomes outside that range. For example, consider the following example:



Here we have a model where *either* a quadratic functional form or a linear functional form fits our data extremely well. *Maybe* there's a third decimal place distinction in some test statistic, but really, they're the same *in the area we have our actual data*.

But now suppose you're asked to make predictions about outcomes at , say, $x = 5$. Out there, the functional form you choose has a tremendous impact! So this is what's called "model dependence".

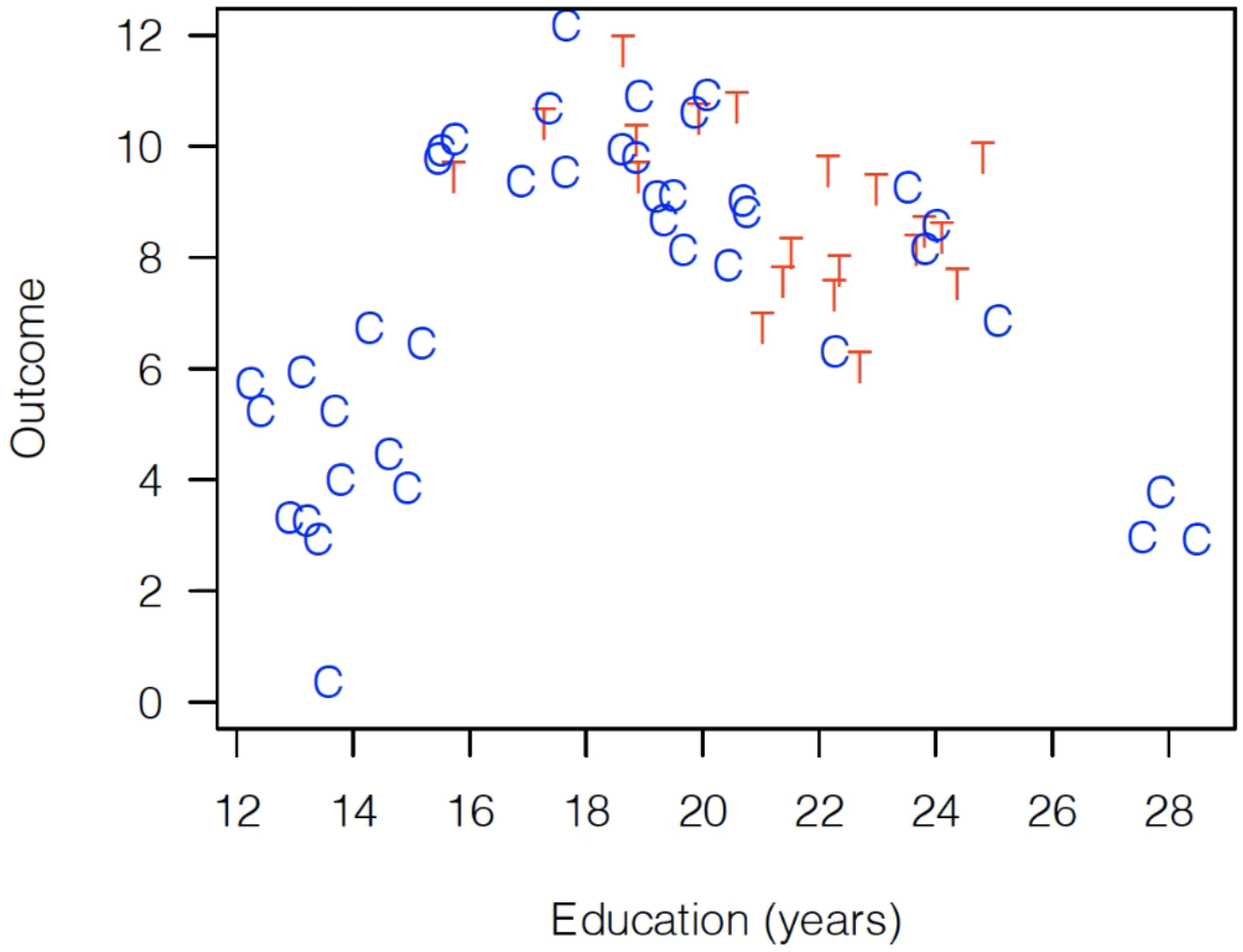
Note that "model dependence" is a condition that depends on the application. If you fit this model to predict Y at $x = 1.02315$, there's probably very little model dependence here. But if you want to predict a value at $x = 5$, there is. Something we'll come back to.

Model Dependency, Causal Inference, and Imbalance

OK, fine, but I *know* I'm never supposed to extrapolate that far from my data, so who cares?

Well... funny story: turns out we do this in causal inference all the time. That's because if our treatment group and our control groups look very different (something called "imbalanced"), then we are sometimes actually extrapolating a relationship we estimate *from the control group data* to estimate what the treatment group would look like in areas where have no actual data from the treatment group.

OK, let's illustrate that. Suppose we have the following observational data (i.e. this is data from the world, not a randomized experiment). It's made up, but let's assume it's from a consumer survey, where Treatment is whether the customer came to the store with a coupon, and "Outcome" is spending. Read T's are consumer we've been treated (got promotion), blue Cs are consumers who weren't treated (controls).

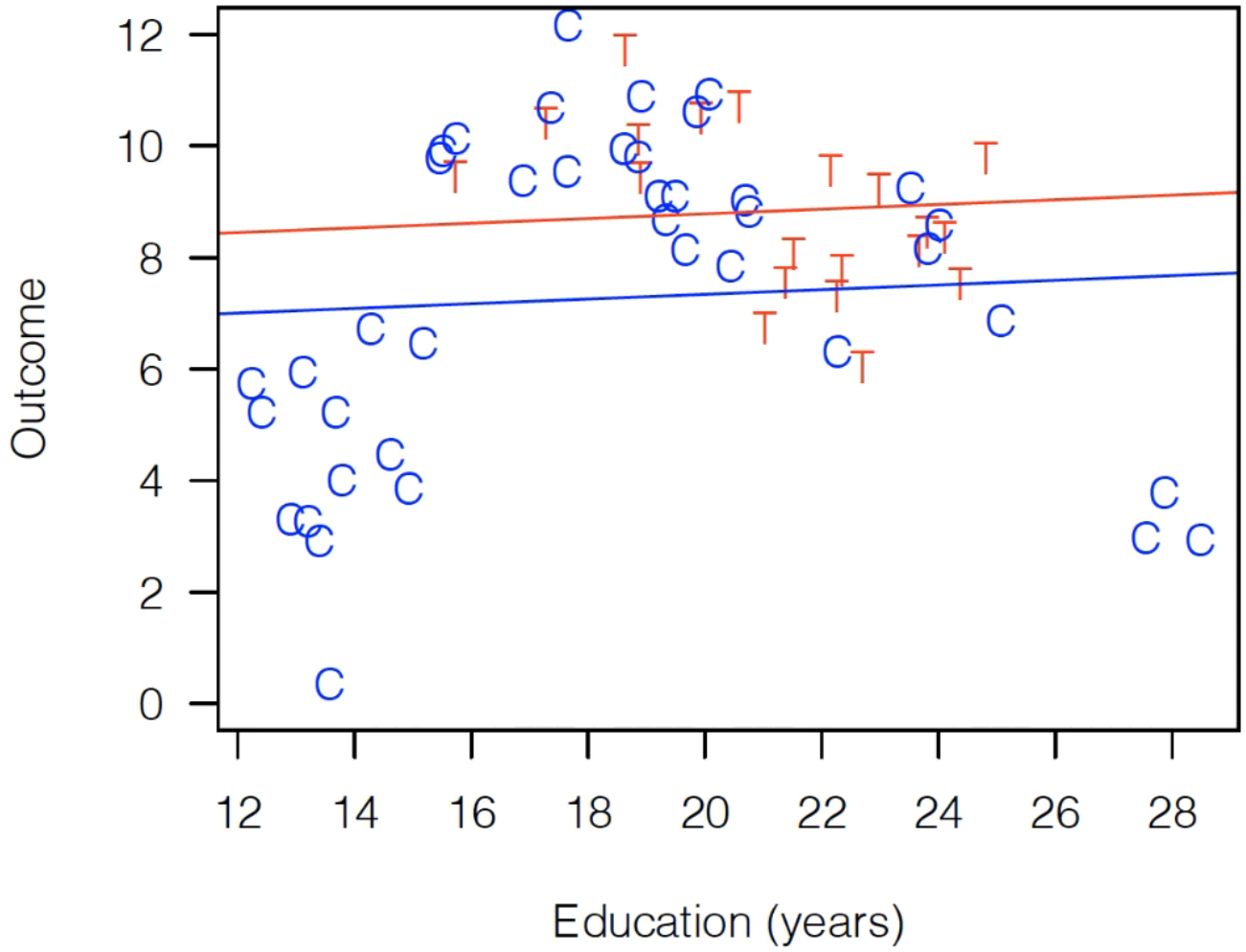


As you can see, this data is badly *imbalanced*, meaning that the range of values of Education are very different for our control population and our treatment population. For the control group, Education ranges ranges all the way from 12-28 years, while our treated group only has observations between 16 and 24.

Why does this matter? Suppose we now fit the following regression:

$$Y = \alpha + \beta_1 treat + \beta_2 educ + \epsilon$$

That fit would look something like this:

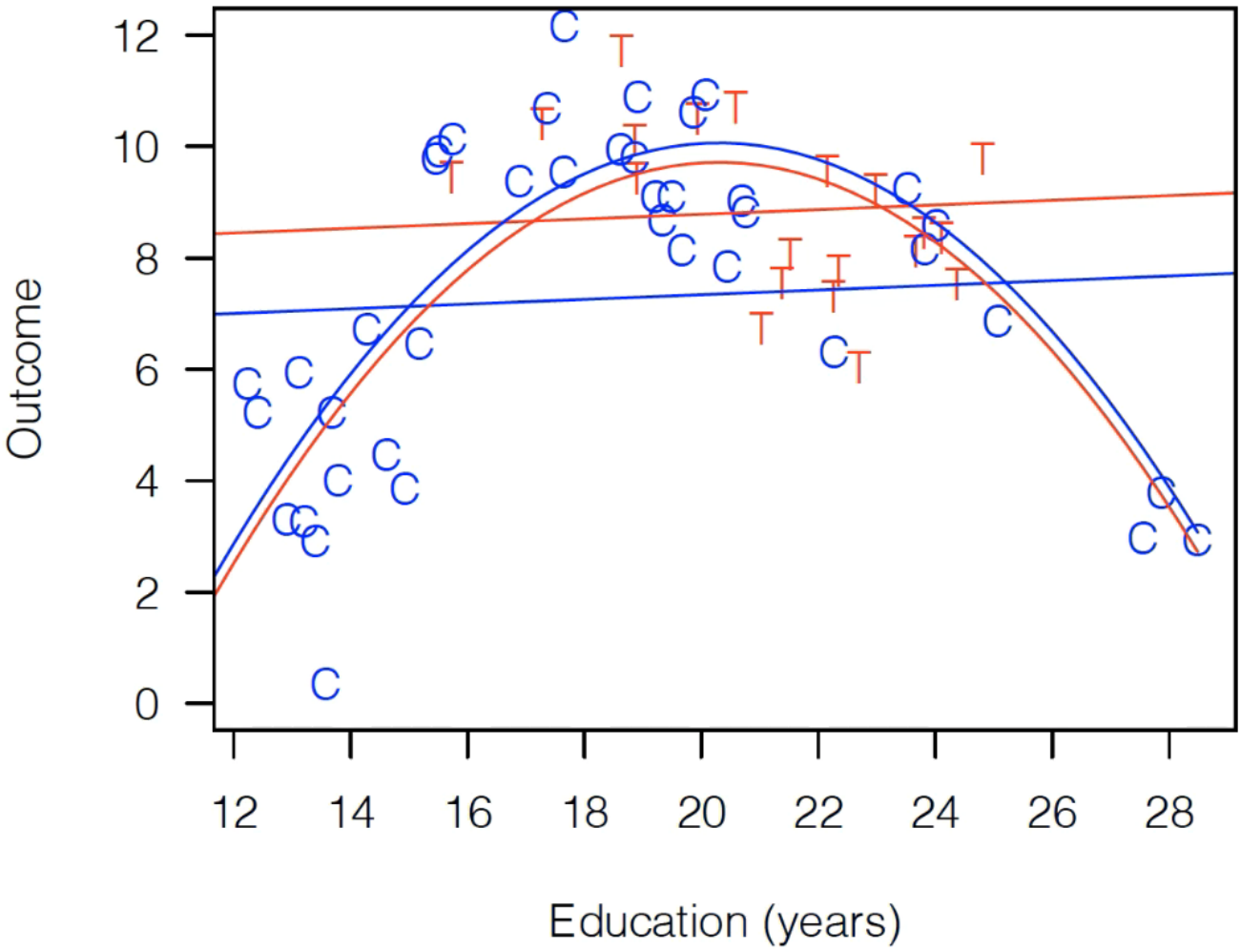


As you can see, Treatment is above Control, so we have a large positive treatment effect!

OK, but that clearly seems weird. It's not fitting those control observations, so instead let's try this:

$$Y = \alpha + \beta_1 treat + \beta_2 educ + \beta_3 educ^2 + \epsilon$$

That looks like this:



Well, now the treatment effect is *negative*! Red below blue!

So which is right? Well... the thing that should probably make us uncomfortable is that the way we drew that red parabola is that we estimated a relationship for our full data, but really we were just fitting our control data, then applying that fit to treatment. As you can see, no treatment observations are really contributing to us fitting a big inverted U, since all the T observations are in the middle of the distribution.

Now, if you want to take a strong theoretical position and say "I'm really confident education plays the same role for people who brought in coupons as those who did not", then you can stop here. But...

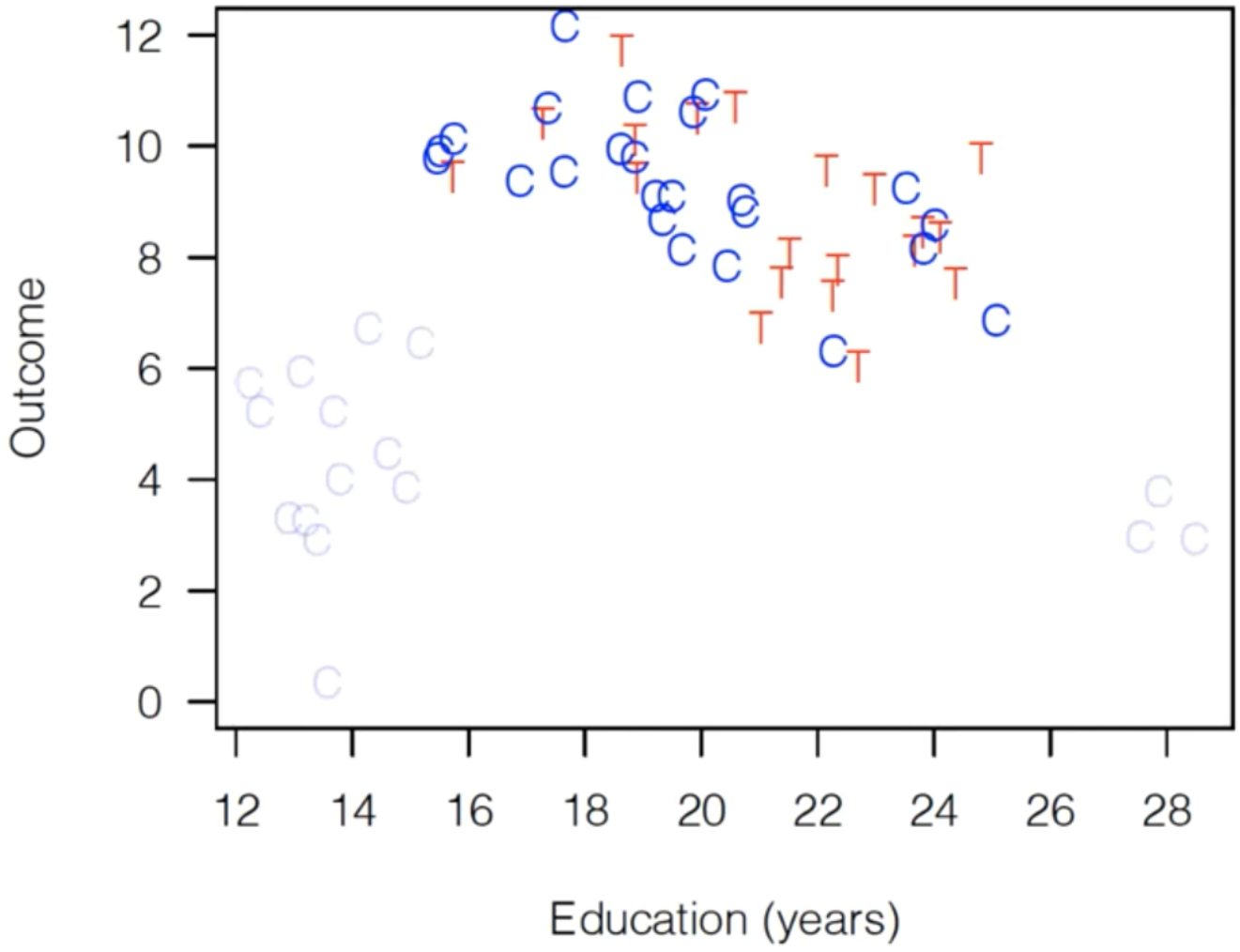
The idea of matching is that maybe that should make us nervous, and that maybe there's a way to avoid that

Matching: Better thought of as pruning

When you hear the term “matching”, your first thought is probably that the focus of the method is finding pairs of observations that are similar, and… you’re not *wrong*. But a better way to think of matching is as a method for *pruning* observations that don’t have any close matches.

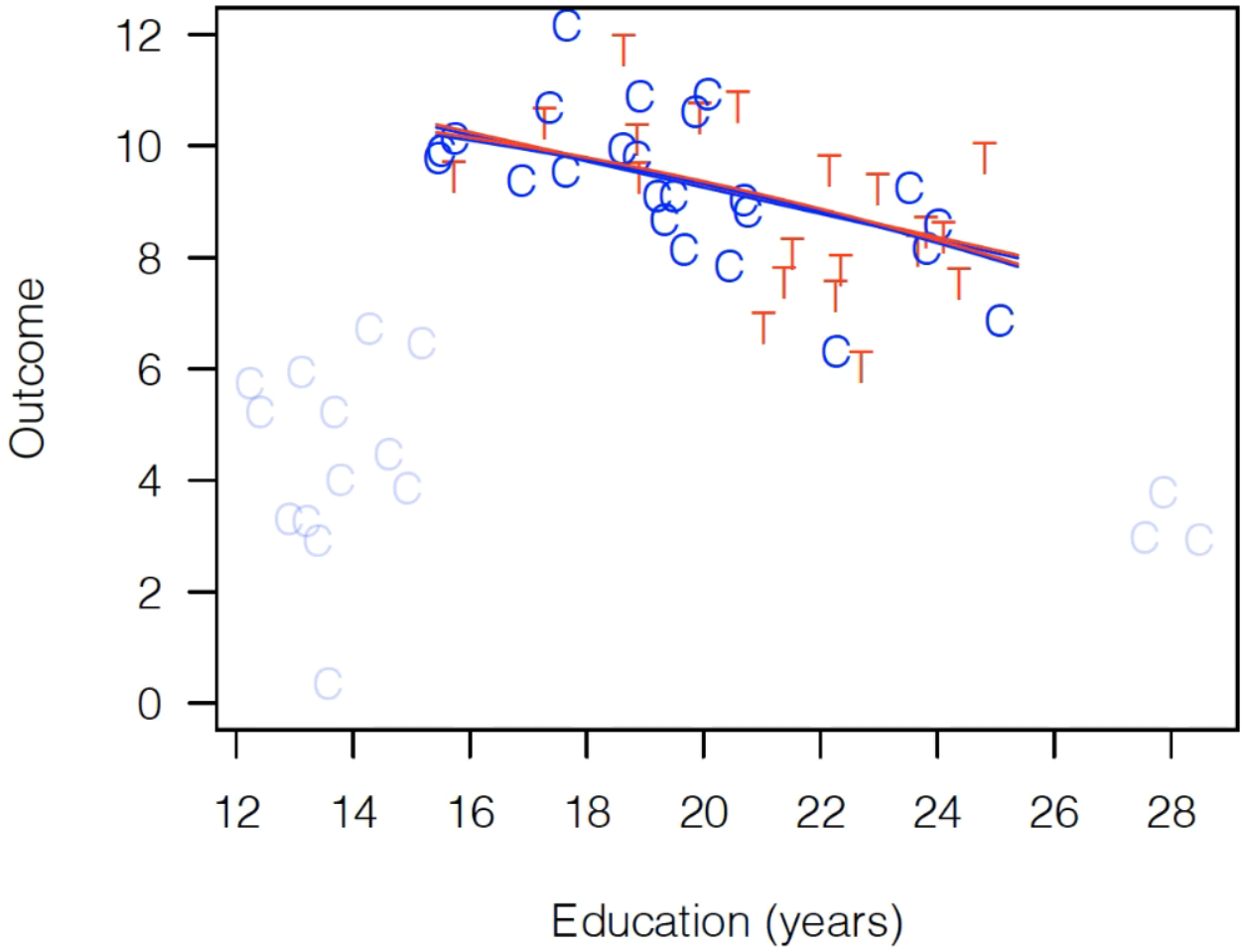
In other words, at its core, matching is about pruning back your dataset until you’re left with control and treatment observations that look relatively similar. Any control observations that are completely unlike any treatment observation (in terms of our explanatory variables) you drop, and any treatment observations that are completely unlike any controls you drop.

So if we ran matching here, we’d end up with the following dataset (dropped observations greyed out) (matching isn’t deterministic, there are choices, but a reasonably matching algorithm would likely do something like this):



As you can see, we've now dropped all the control observations that are completely unlike our treatment variables. And now the range of Education for control variables and treatment variables is about the same – we have a "balanced" dataset.

And now it doesn't matter if we do a simple linear fit or if we include a quadratic term – the results look exactly the same (i.e. no model dependence):



And if we limit our focus to these observations, we see that regardless of the functional form we put in our regression, we get no treatment effect.

The Cost of Matching

So what do we lose when we do matching? The main answer is that the group of people for whom our treatment effect is being estimated is changing dramatically. We aren't estimating an effect for the whole population we started with, we're estimating the effect for this small group where T and C overlap, which may or may not be useful in later applications.

In these kinds of studies, we often have far more control observations than treatment observations, and the control variables will often cover a much bigger range of values of your explanatory values (if not many people get treated, then it makes sense that the people who get treated may all look similar, and so may fall into a smaller range of, say, education than the general populous).

In these situtations – where during matching you get to keep all your treated observations because they all have close controls, but you drop some controls, then what you're estimating is the Average Treated on the Treated, which isn't half bad (it's often all we get with observational data).

But if you have Treated observations that don't look like *any* control observations and so you have to drop them, well now you're estimating an effect on... the sample of people in the range of explanatory variables for which there are both treated and untreated observations? And who really knows what that is.

Causal Inference Assumptions

Before closing our discussion of the high level pros and cons of matching, it's worth emphasizing one important point: both matching and good old fashion linear regression rest on the same basic assumption when it comes to causal inference: there are no baseline differences between the treatment and control groups *after conditioning on observable variables*.

Matching, in other words, is a way to address possible *modelling errors*, not to fundamentally change what assumptions we're making about the differences between treatment and control groups in terms of potential outcomes.

In Summary

- Matching is really best thought of a way of **pruning** your data to a sample where C and T observations look similar (impoves balance)
- Where the original data has observations of C and T that *don't* look similar (it's imbalanced), matching helps reduce the sensitivity of results to the functional form assumptions you make in your regression.
- But by changing the data on which you're making an estimate, you are also changing the actually population for which you're estimating your effect, which may be important to understand.

- To use matching for causal inference, you have to make the same basic assumption you use for regular linear regression: controlling for observable differences is enough to eliminate baseline differences in terms of potential outcomes for our treated and untreated observations.

How to Match

In this reading, I'll give a high level summary of how matching works before referring to a youtube lesson a the nitty gritty of a few specific implementations.

Pruning Your Data

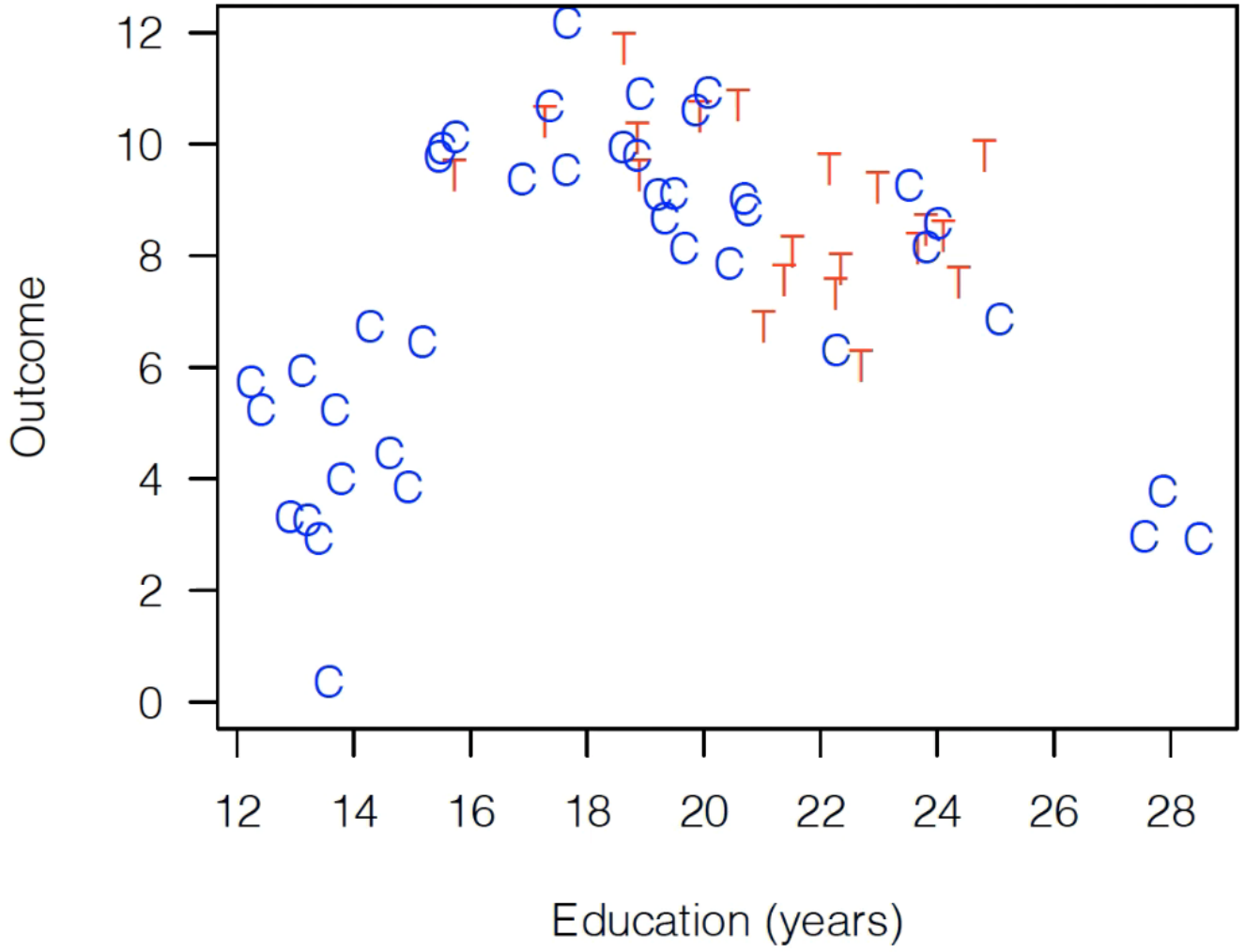
As noted in our last reading (this reading is a follow-on to The Why of Matching, so if you haven't read that start there), matching could be more appropriately called "pruning", as the goal is to winnow down your dataset until you have a set of observations for which your control and treatment variables look very similar in terms of observable characteristics. So how do we do that?

A simple matching algorithm would proceed like this:

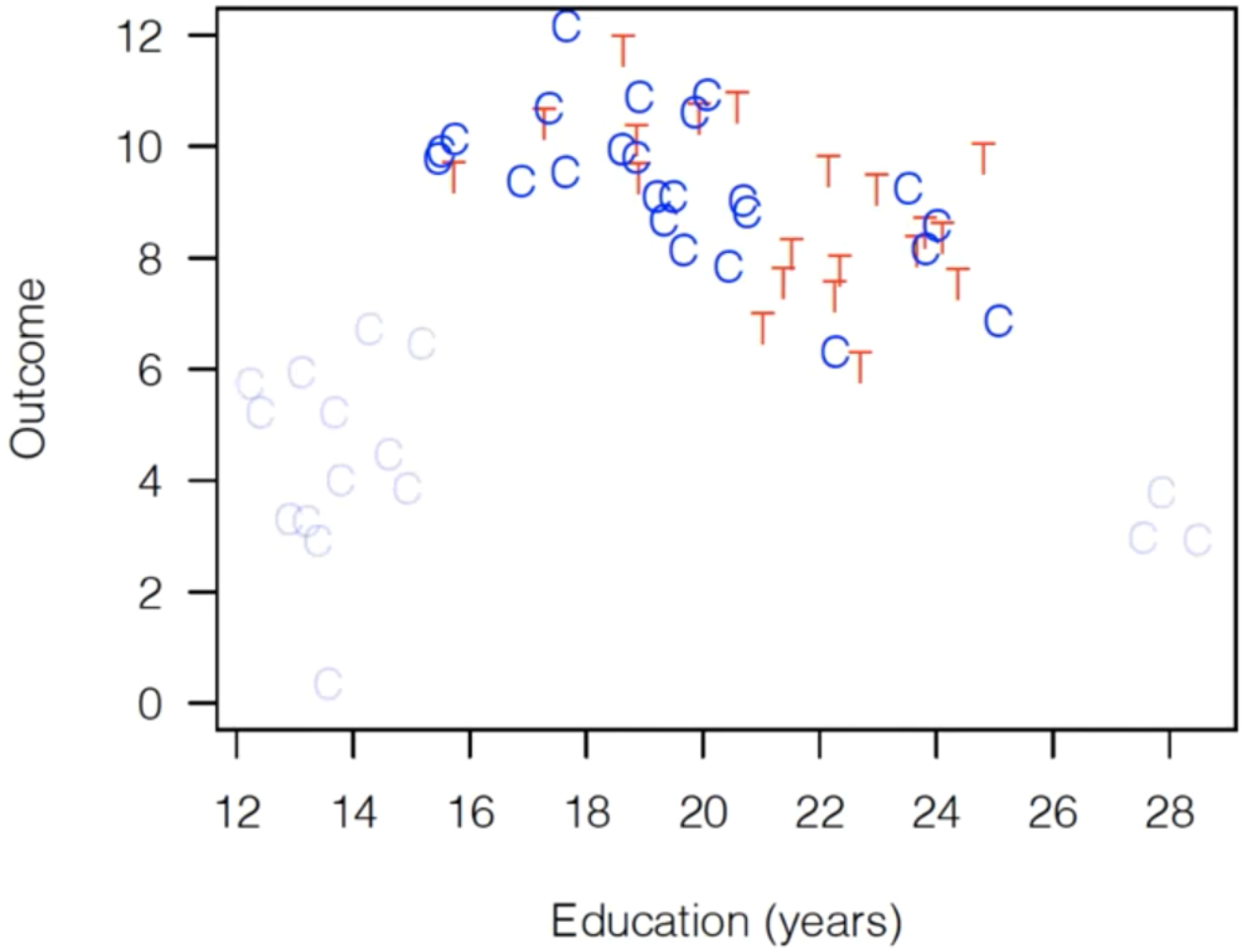
1. Loop over all your treated observations.
2. For each treated observation, look for the most similar untreated observation (not already in a pair) in terms of your control variables.
3. If that untreated observation is too dissimilar to the treated observation, throw away the treated observation. (As the user you have to pick a threshold for how dissimilar is ok.)
4. If not, call them a pair and keep both.
5. When you've finished looping over your treated observations, throw away any unpaired untreated observations.

When you're done, you'll have a collected of pairs of observations (one treated, one untreated), where both members of each pair are very similar in terms of their observable control variables. All other data has been thrown away.

To illustrate with the example from the last reading, if we started with this data:



A simple matching algorithm would probably prune it down to something like this:



Then, and here's the cool part, you take this dataset and analyze just the way you would otherwise! Just run your regression on this dataset!

Measuring Similarity

The biggest decision you have to make when doing matching is deciding how you want to measure whether two observations are "similar". The most simple, commonly used strategy is to measure the dissimilarity of two observations in a pair by:

- For each control variable, calculate the difference between the two observations in the pair (so if one has Education of 20 and one of 14, you'd get 6). Note the example above just has Education, but in reality you likely have dozens of these variables, so you do this for each variable.
- Normalize those differences by the standard deviation of the variable (so divide 6 by the standard deviation of Education)

- Square all those differences, add them up, and take the square root. That's your "similarity" score.

Basically, this is like taking the euclidean distance between the points in some really high dimensional space, except you also normalize distances by their standard deviation, a strategy called "Mahalanobis Distance Matching".

Of course it's not the only strategy – the video linked at the bottom of this reading will direct you to a talk on three very good strategies, as well as their strengths and weaknesses – but that's generally the idea.

When Can / Should I Use Matching?

Matching is best used in a somewhat odd situation: a place where you have some overlap in what your treated and untreated observations look like (called having "common support"), but where you also have some areas where they *don't* overlap (imbalance).

The first is necessary because when you prune your data, the goal is to keep only observations that look similar, so you need *some* area of overlap, or you won't have anything to match!

At the same time, however, if there's *no* imbalance, then you don't really need to do matching.

So when should you use it? When your distributions have **both** areas of overlap and areas of imbalance.

Checking Balance

There's a balance you have to strike when matching: the more strict you are about the maximium dissimilarity you're willing to include before you throw out a pair of observations, the more balanced your final dataset will be, but the smaller your dataset will be do.

Right? If you reject any pairs that aren't almost exactly identical, you'll end up with less data, but what's left will be more balanced.

So for your application, you have to decide on whether it's better to have the statistical power of more observations, or the better balance from fewer.

Analyze!

Now the best part of matching: now you just do do what you would have done normally.

In other words, you can think of this as a kind of "pre-processing step", and now you can carry forward by feeding this into a regression just the way you would with the original data.

Specific Models

OK, for the details of a few common models, please go watch this great video by Gary King – you can probably start 15 minuntes in, and should watch till at least 45 minutes, though what follows is also really interesting!

Gary King on Matching

How to Read (Academic Edition)

In this class, you will be asked to do significantly more reading than many of you — especially those of you from engineering backgrounds — are used to. Moreover, many of our reading may feel different from a lot of the reading that you've done in the past, and so I want to take a moment to discuss *how* you should approach reading in this — and indeed in *any* — reading intensive course.

Knowing that many students in this class speak English is a second language (and so may find reading more time-consuming), and many of you also don't have experience with more reading intensive courses, I have worked very hard to ensure that the readings I assign communicate key concepts as efficiently as possible. But as we will see, learning to answer causal questions well requires wrestling with the complexities that arise when apparently simple math meets the real world, and that unavoidably requires thoughtful exposition.

Read Actively

The first major piece of advice I can offer is that you should always be active when reading for this course. That may mean taking notes on a separate piece of paper as you go, or highlighting

passages that stand out and adding comments in the margins. But reading of the type you will encounter in this course should never be a *passive* process.

This is especially true any time you encounter mathematical notation. A key skill in answering causal questions is the ability to map concepts represented in mathematical notation onto facets of real world examples. With that in mind, any time you are reading something written in mathematical notation, it is good practice to think of a specific example and see if you can relate each term you encounter to the real world example.

Be Patient with Examples From Different Domains

Data science is an extremely diverse field. This course does its best to embrace that diversity though the use of examples from a wide range of substantive domains. This, at times, causes frustration among students as many examples will necessarily come from domains that don't feel relevant to your particular interests. Try and resist this urge — while this may *seem* like a problem, it's actually emblematic of a huge opportunity for intellectual arbitrage (the porting of insights that have been richly developed in one domain to a different domain where they are unfamiliar)!

This will be particularly relevant when we get to the study of causal inference (the study of how to answer Causal Questions), as there are lots of concepts that have been well-developed in the social sciences that people are only now starting to apply in industry, meaning many of the best texts and examples will be public policy or social science oriented.

And while the downside of that is that there aren't as many great books written about causal inference in industry as you may wish, the upside is that there are lots of opportunities to for young data scientists to innovate by applying these concepts in new ways!

So please bear with these examples, and practice trying to apply the concepts you read to an industry example that matters to you.

Do NOT Summarize with LLMs

I fully recognize that there is a strong temptation when faced with a long reading to stuff it into a Large Language Model and ask for a summary. **Don't.**

There are a few reasons for this. The first is that while an LLM can provide you with a broad summary of what you're reading, it will necessarily have to exclude all the nuance in the original reading. If you just want to figure out if a reading is generally relevant to your interests, that's fine; but you're here to learn a subject — one that can can be infuriatingly subtle — and cutting out all those nuances will limit what you can learn *and* prevent you from being able to test your understanding of concepts by wrestling to understand how each new sentence relates to what you've read previously. So just as the goal of the readings isn't to allow you to answer our reading reflection questions (those are just there to draw your attention to especially salient points), nor is the goal of the readings to understanding it at the level of a summary.

The second big reason is that the process of summarizing material yourself is critical to consolidating your learning. This insight comes, in part, from research on whether students taking notes on computers learn more effectively than students taking notes on paper. This research has found that students taking notes on computers can write much more quickly than students taking notes by hand, but that counter-intuitively this seems to result in *worse* learning outcomes (based on subsequent learning assessments). Why? It appears that students taking notes on computers are effectively able to *transcribe* everything happening in class, while students taking notes on paper have to think about the material in real time in order to summarize it enough that they can keep up taking notes.

Letting LLMs summarize material for you seems likely to cause a similar issue — by allowing an algorithm to organize the information in a more concise manner, it deprives *you* of the opportunity to engage with the material to create your own summary, a process that forces you to *actively* think about the connections between concepts. The importance of this type of *active learning* is one of the biggest bedrock findings of research on learning in recent years — students who have material given to them (e.g., through a passive lecture) often think they understand material, but it is the students who learn material actively — through class or group exercises, problem sets, or other activities — who perform better on learning assessments.

Finally, learning to focus on a reading for a prolonged period of time is an important skill, and one we practice less and less in the modern age. Sometimes letting our attention flitting from thing to thing is fine, but being able to focus for prolonged periods when required is an important skill to cultivate! (If you're interested, you can find a really interesting discussion of this idea here.)

Backwards Design

Backwards Design is a way of developing an efficient strategy for completing a new data science project, and in my view it is one of the most important skills of a professional data scientist.

If you don't have a lot of professional data science experience, it may not be obvious why this is an important skill, or even why I call it a "skill." That's because most data science students' experience with project development comes from classroom exercises or sites like kaggle. These types of projects are excellent opportunities for learning, but it is usually the case that – unbeknownst to the student – these projects have been carefully tailored to have clearly defined goals, and they come with data sets that have been cleaned and filter to provide only relevant variables. This is usually done for good reasons – the instructors design these exercises in a way that focuses student attention on the skills that they are trying to develop (like model selection or model interpretation). But as a result, students often come away with the impression that most of what data scientists do is work with statistical models.

In reality, however, often the most important thing that a data scientist does is (a) develop and articulate a concrete, feasible objective of a data science project, and (b) develop a strategy for achieving that objective efficiently. And Backwards Design is one of the best ways to go about accomplishing both of these goals.

Overview

As the name implies, the idea Backwards Design is to *start* by clearly defining where you want to end up at the end of the project, and then working backwards to figure out exactly what you need to do to get there. Backwards Design is actually a common project management strategy and a range of different domains, and so you may already be familiar with the strategy and broad terms. In this class, however, we will focus on a five-step strategy for doing Backwards Design in data science:

1. Define the problem you want to solve.
2. Define a *question* that you wish to answer to help you solve this problem.
3. Articulate exactly what an answer to your question would look like.
4. Determine the variables you would need in order to generate that answer.

5. Identify data sets with those variables, and develop a strategy for bringing them together.

1) Define Your Problem

This first step should be the most straightforward, and yet you will be surprised at how often it is never actually explicitly addressed. People get so excited about the idea of data science that they will often come to you (the data scientist) with the data set and say "do some data science with this!" So the first thing you should always do when starting a data science project is make sure that you can clearly articulate the objective of the project. In addition, you should always make sure that *your stakeholder* agrees with that articulation of the problem you are trying to address! There's nothing worse than spending weeks on a project and then discovering that it's not actually a value to your stakeholder.

Here are a few examples of defined problems:

- We don't know how to reduce mass incarceration.
- My business can't identify potential new customers.
- We don't know who is going to develop Alzheimers, so we can't test early interventions.

2) Define the Question You Wish to Answer

Although not everyone will agree with us, it is my view the data science is fundamentally the practice of using data to quantifiably answer questions about the world.

For example, when we ran our regressions of birth weight on various demographic variables and whether the mother smoked during pregnancy, those models were answering the question "is maternal smoking associated with lower birth weight (at a statistically significant level after controlling for other confounds)?"

If someone who runs a commerce website runs an A/B test where users visiting the site are randomly assigned to see two versions of a new landing page, and we then track their purchasing behavior, then when we analyze that data statistically what we're doing is answering the question "Which of these designs is more effective at getting customers to buy things?"

And finally we can think of supervised machine learning algorithms as answering two types of question: there's the broad question that you answer by building a model and evaluating it ("can we

identify cancer from patient x-rays, and if so, how well?"), and then the narrow question the is answered each time the model is run ("given this x-ray, how likely is this patient to have cancer?"). (There's a small digression on supervised machine learning and this "data science is about answering questions" conceptual framework at the end of this if you're interested).

So the next step in backwards design is to ask yourself: what question, if answered, would help you solve the problem that motivates you?

Defining a Good Question

A key feature of a good question is one that it is *concrete*, *tractable* and *answerable* by a data science project. Your question meets these criteria if it **directly implies how you should approach the data science project, and that approach seems feasible.** If your question is so vague that you don't immediately start thinking about the data you want to collect, it's not a good motivating question.

To illustrate, here are a set of *bad* questions to the three problems described above:

- What policies reduce mass incarceration?
- Can machine learning help me identify potential customers.
- What indicates Alzheimers?

By contrast, here's a set of concrete, tractable, and answerable questions:

- Does the availability of grand juries result in longer sentences? (Grand juries are a pool of citizens prosecutors can ask for guidance on sentencing. In theory they're supposed to hold prosecutors accountable, but in reality its often said that prosectors can shape the information grand juries get so they reach the recommendations that prosecutors wanted to begin with).
- What attributes are common to the customers who buy the most from my business?
- Are there lab results common in patients who later develop Alzheimers (diagnosed post-mortem) that we don't see in patients who don't go on to develop Alzheimers?

For the first set, we've asked questions, but they're so vague that it's not clear how you should approach answering the question. In the second set, by contrast, likely first steps are very clear. For example, the first good question clearly implies we need to find data on sentencing from places with and without grand juries. For the second question, its clear we need data on our current

customers, and data from the general population for comparison. And for the last question we clearly want lab data from patients with and without diagnosed Alzheimers!

Moreover, for these answerable questions, you can imagine what the answer to the question will look like: for the first, you could have a regression that regresses sentences on grand jury availability controlling for crime committed; and for the second, you could imagine a table that compares various demographic characteristics of customers to non-customers.

Why is Having a Good Question Important?

- It will save you from getting lost in your data, since it helps you focus you energy.
- Being able to articulate the question you wish to answer allows you to make sure that answering that question will actually help address your motivating problem (/make sure that your stakeholder agrees that answering your question will help them). There's *nothing* worse than getting excited, diving into your data, doing lots of work, and then getting a result that doesn't actually help address the problem that motivated you (but it happens *all the time*).

3) Write Down What An Answer Would Look Like

Seriously. Do it. Not abstractly: I mean draw the graph, figure, table, dataset with columns of predicted values and predictors, or set of model diagnostics you want to generate as a way of answering your question. Literally draw the graph, label the axes, etc.

Why?

- If you can't, then it turns out your question wasn't sufficiently concrete.
- You can then show this to your stakeholder to ensure they think this constitutes an answer that would help them solve their problem (and avoid later being told your work doesn't actually help them)
- It makes it even clearer what steps you need to do next to generate this result.

Falsifiability

OK. So now you're written down what an answer to your question looks like. But there's one other key feature of a good question that we didn't get into above: it should be *falsifiable*, which means

that (a) you should be able to articulate a hypothesis about the answer to your question *and* (b) know what a result to your question would look like.

So when writing down what your answer should look like, do the following:

1. State a hypothesis about what you think the answer to your question is likely to be.
2. Draw what an answer to your question would look like *if your hypothesis is true*.
3. Draw what an answer to your question would look like *if your hypothesis is false*.

For example, consider our question about grand juries and sentecing. My hypothesis might be that grand juries result in longer sentences because they insulate prosectors from accountability.

My result, as described above, could be a regression table where I regress sentences on whether a county has a grand jury along with controls for crime committed.

The answer if my hypothesis is true is that we'd have a positive coefficient on the presence of grand juries.

The answer if my hypothesis is false is that we'd have a zero or negative coefficient on the presence of grand juries.

Why do all this? Because it helps ensure that the way we're planning to answer our question will actually answer it by generating different results in different states of the world.

4) What Data Do You Need?

Congratulations! You've now completed the really hard part of a data science project: defining your goals. Now we turn to the easy stuff.

First, now you know your goal – the result described above – we turn to how we will actually answer our question. So ask yourself:

- What variables do I need to make the result I described above,
- What population do I need represented in that data

So let's think about our business trying to find new customers. Clearly, we need data on (a) customer spending, and (b) the demographics of those same customers.

We also need data on *both* people who spend a lot, and people who don't spend a lot so we can compare these two populations. We could do this either by getting data on all our customers and comparing big spenders from people who don't buy much (if there's a lot of variation in the data on level of spending), or we could compare current customers to the general public (most of whom aren't customers).

5) Where Can You Get That Data?

OK, now you know the variables you need measured and the populations for whom you need those variables. Now let's figure out:

- Where can I get that data, and
- If the data will come from different datasets, how will I combine them?

In the customer example above, for example, we can start by looking for the data the company already has on its current customers. What's in that data?

We can also look for similar demographic data for non-customers using a resource like the American Community Survey (the annual survey run by the US Census bureau).

If we use government data for our comparison group, we'll then have to make sure we get comparable samples, so we'll want to make sure we can match our observations on things like age, gender, and where people live, so we'll need to make sure we have those variables in both datasets.

Wrapping Up

Congratulations!

By the time you've done these 5 steps, you've managed to develop a concrete plan for exactly where you'll focus your time, and you've nearly guaranteed that the result you're working to generate will actually be useful in solving the problem that motivates you or your stakeholder. There's still a lot of data wrangling, model selection, etc. ahead, but at least you won't get lost in your data, or do lots of work that ends up no helping anyone!

Want a template for this? Great news! You can download one here!

A Digression on Supervised Machine Learning

As noted above, we can think of superived machine learning as a tool that answers two types of questions: the first is the broad question of "can we predict [outcome of interest] using [variables we have] and the training set we have access to?", and the second is the more narrow prediction question "for a given set of predictors, what value of [outcome of interest] would the model predict for a given observation"?

The first, I think, is pretty straightforward. But there's a nuance to the second question that's super important to understand: when we ask a supervised machine learning it's prediction for a given observation, what we're fundamentally asking your model is: **"how do you think the entity who labeled the data in your training data set would label this new observation?"**

Because that's all that supervised machine learning does: it develops models that are designed to replicate the behavior that gave rise to the data set due used for training your model. For example, if you train a supervised machine learning algorithm to label pictures with the name of the animal in the picture by feeding it a bunch of pictures that have been labeled by undergraduates at an American university, than what you are training that machine learning algorithm to do is answer the question "how would an American undergraduate label this photo" every time it sees an unlabeled photograph.

Obviously different supervised machine learning algorithms go about trying to answer this question in different ways, and some will be more successful than others depending on the context (which is why we spend so much time studying model selection in machine learning courses), but answering this question is *always* the goal to which they aspire.

This is a bit of a digression, but I think it's an important one: recognizing that this is all supervised machine learning algorithms do is important because it helps you, the data scientist, understand the limitations of supervised machine learning algorithms. For a surprisingly long time, people thought that machine learning algorithms were incapable of harboring racial or sexist prejudices. They are, after all, just built of math, and math can't be racist, can it? And so companies like amazon tried to build supervised machine learning algorithms to help them decide who to hire. The problem, though, is that they trained them using data on which people human employees had decided to hire in the past, and data from subjective employee evaluations that had been made by human supervisors. And because this gave rise to an algorithm that looked at people's resumes and asked itself "what would Amazon's (very human) hiring staff and supervisors have thought of this

person?," the algorithm of course inherited all the biases of those humans. And so, OOPS!, when Amazon suddenly realized that "their new recruiting engine didn't like women," they had to abandon the project.

OK, digression on bias in data science complete. For now. :)

Making Decisions Using Data

(This is something that I know I need to add to my classes but I haven't really managed integrated yet. I talk to my students about that fact that they should just pray at the altar of $p<0.05$, but instead weigh the relative costs and benefits of Type 1 and Type 2 errors in the context in which they are trying to make a decision. But where I really struggled is the fact that you can't directly map P values onto decision theory very easily because of all of the weirdnesses of frequentist P values—e.g. A p-value of 0.05 means that under the null, the odds of a Type 2 is 5%... but that means the ACTUAL odds of a Type 2 error if you have a p-value of 0.05% is pr(null is true) * 0.05. 🤦.)

Writing to Stakeholders

In the spirit of what is to follow, I will begin with the most important idea in this reading:

> The key to effective writing to stakeholders is to continually ask yourself — where ever you are in your document, whatever your thinking of writing, and whatever you're debating including — if my stakeholder stopped reading my report *right at this spot,* have I told them everything I want *them* to have learned.

Why Learning to Write to Stakeholders is Hard

As students (especially data science students), the way most of us were taught to write reports is to:

- start with an introduction that helps explain the broader context in which the report is being positioned,
- describes the data that we intend to use,

- describe that how we have wrangled and cleaned that data,
- describe how we plan to model that data,
- report the results,
- and discuss limitations and tack on a boilerplate conclusion.

This structure makes a lot of sense in the context of a class because the order of presentation mirrors the objectives of the assignment: demonstrate that you understand the substantive topics that are being taught, demonstrate that you are being thoughtful about the data that you are collecting and that you have internalized the emphasis your instructors have placed on the importance of data cleaning, and that you understand the principles of data modeling. Results come last because in the context of a class, your results don't actually matter. No professor is going to make a major business decision on the basis of a student report, and no government is going set policy on the basis of what you say.

This structure — which almost entirely front loads material that is not particularly interesting to the instructor — is also viable because it is basically the *job* of the reader (your instructor or teaching assistant) to read everything you wrote.

Putting Yourself in the Stakeholder Shoes

None of that is true in the real world. In fact, I would argue that being taught to write reports in this manner is about the worst possible preparation one could give students for learning to communicate to real-world stakeholders for two key reasons:

1. In the real world, the *only* thing that your stakeholder actually cares about are the conclusions you have reached about how to solve their problem (i.e. your results and how they relate to the stakeholder's problem).
2. There is nothing more scarce in any organization than a decision makers time, and so *the moment* it stops being obvious to the decision maker why the material their reading is directly relevant to solving their problem, there is a very high probability that they will let their attention shift to one of the other hundred critical issues vying for their attention.

Because of these two facts, when writing to a stakeholder you should always — *always* — be asking yourself two questions:

1. If my stakeholder stopped reading right now, have I already told them the things that I think it is most important they walk away knowing? and
2. At every transition — between sections, between topics, and even between paragraphs — is it explicitly clear to the reader that what follows is relevant to solving the stakeholder's problem so they have and affirmative reason to not get distracted?

What does this look like in practice? It depends a little bit on how much of your stakeholder's attention you think you can get in the best of circumstances (intelligence briefings were written very differently for Donald Trump than for Barack Obama), but the structure I am going to advocate for is roughly the following:

- **Executive Summary:** A full summary of absolutely everything you want your stakeholder to walk away knowing if they were to read nothing else in about two to four paragraphs. State the problem that you are setting out to address, state the question that you're going to answer in order to help solve that problem, and at a very high level state how you are going to answer that question, state the answer you have found, reiterate how that result helps solve the problem.
- **Context:** With the Executive Summary out of the way, you have hopefully piqued your stakeholder's interest just enough to double back and provide *a little* more context about the problem you are trying to address and the context and which it is situated. But don't get complacent — even here, it is important to very clearly and explicitly motivate why all of the information you are providing is relevant to solving the stakeholder's problem effectively.
    - This is where the answers you generated to your Exploratory Questions fit — they provide information to your stakeholder about why you have chosen to prioritize certain facets of the problem, and why those choices are correct.
- **Your Strategy:** Now you get to explain how you are going to answer the question that will help solve the stakeholder's problem in more detail. But don't get lost in the weeds — the stakeholder only needs to understand enough about your strategy (including your data) to be able to evaluate how much confidence they should have in our results and conclusions.
    - This is perhaps **the** hardest section for data science students to write, not because they can't think of what to put here, but because they want to include *way, way too much*. There is a very natural and very human tendency when you finish a project that you have put a tremendous amount of work into to talk about all of the work you did. And most of that work will involve wrestling with data, dealing with merging issues, tuning models, etc. But almost nothing you have done in this domain is something that your stakeholder needs to know about. You can write it all up — there may very well be some people on the

stakeholder's staff who will want to read it — but it belongs at the back of the report in an appendix, *not* in the body of the report.
    - The **only** exceptions to this rule are places where you exercised substantial *discretion* in how you handled the data or constructed your sample. If you exercised discretion in a substantial way *that may impact how one interprets the conclusions of the report* — for example, you had to choose which states to include in an analysis as control states — you can/should include a *brief* explanation for your reasoning along with a link to an appendix were you discuss those choices in detail.
- **Your Results:** Now you get to present your results in greater detail. Note that nothing here should come as a surprise to the reader — don't try and "hide the ball" to build suspense by not revealing your findings till now because while you may *think* what you've done is so interesting that the suspense will draw the reader in, in reality, it just means that when your stakeholder doesn't make it past the Executive Summary, they will not have learned **the thing** you spent so much time trying to learn.
    - Also, note that while this structure still has a results section towards the end, because you have kept your Context and Strategy sections very short, it should actually be coming up much sooner than your results would come up in a normal class report (in terms of the page number on which it appears).
- **Relating It Back To The Problem:** And now we bring things full circle — now that you presented your results, you have to make sure that your reader hasn't lost the thread by relating your results back to the problem that motivated your analysis to begin with.

Wow! What a Great Way of Thinking About This? Did You Invent This Yourself?

Obviously not. And now that you are primed to think about how writing can be organized in a manner that reflects the likelihood that most people who pick up a document won't actually read it all the way to the end, you will start to see how people front load the things they think matter most everywhere.

Journalism is the quintessential example of this style of writing. *No one* reads entire news articles, so they are always nearly organized with the most critical information up front, after which they double back to fill in additional details for anyone still reading. There are a number of different ways this can be accomplished — for example the inverted pyramid — start with "who? what? when?

where? how?", then add important details, then add context — is one of the first article formats journalists are introduced to.

But the structure I think you're likely to see most if you start looking for it is that in the first two or three paragraphs of a well-written news story, you will notice that there is a single paragraph that is designed to summarize everything that the journalist wants you to know about the story — the "nut graph".

Academic publications also usually follow this structure. They start with an abstract (essentially an Executive Summary of the Executive Summary), an "introduction" (which is, in effect, an Executive Summary), a literature review (to establish why what they're doing is novel and provide some broader context), a not too long discussion of their methodology, their results, and a short conclusion that ties the results back to the motivating question. All the sensitivity analyses, detailed discussion of the data, etc.? Those go into an "Online Only Appendix," a document that is often easily twice as long as the article itself.

Other Points To Bear In Mind

Putting The Pieces Together

In this course, we've spent a lot of time establishing *taxonomies* — ways of organizing and recognizing the many distinct things that we do as data scientists. The assignments we have done for this class have often been structured in a manner to emphasize these distinctions. But when writing to a stakeholder, you don't want to hold too rigidly to those distinctions. A good report to a stakeholder will not consist of stapling together all of the assignments we've done in this class one after the other. The goal of a good report is to construct a narrative that consistently and effectively communicates a key idea to the reader.

Many of you will be familiar with the idea of the "five-paragraph essay." The five-paragraph essay is a structure that is often used to introduce students to essay writing — you start with an introductory paragraph, you write three paragraphs, each of which starts with a topic sentence and then includes evidence in support of that topic sentence, and then you write a conclusion paragraph. It is a useful framework for introducing students to essay writing, but it is also a framework that is best left behind as soon as it is understood.

The emphasis the assignments we have done in this class puts on different types of questions is similar — the goal is to make sure you can recognize the different types of questions and the purposes to which we put them. But your stakeholder report should not include a section called *Exploratory Questions* — rather the answers you generated in answering your Exploratory Questions should be reflected in how you've refined your problem statement, and the results of some of those analyses will likely make an appearance as tools for motivating the focus of your final analysis.

Justify Your Methods With Concrete Examples

In discussing the methods you choose to use — such as a difference-in-difference design or matching — be sure to explain *why* you're using the method in concrete terms that are relevant to the context in question, and without using technical language your stakeholder may not understand.

If you're doing a difference-in-difference design (and not just a pre-post comparison), give examples of specific events that would cause problems for a simple pre-post analysis but that aren't a problem for the difference-in-difference design you are using. Consider the opioid project from IDS 720. In that project, we estimated the effect of state-level changes in opioid prescription regulations in states like Florida and Texas on opioid shipments and overdoses. For that project, it was important to give an example — like the US federal government creating a national limit on the amount of opioids manufacturers were allowed to ship to individual clinics around the same time — that would cause incorrect inferences to be drawn from a simple pre-post comparison, but not from our design of choice (a difference-in-difference).

If you don't include an example like this, it won't be clear to the stakeholder why you feel fancy techniques are necessary. And if you can't think of an example like this, then maybe you don't *need* the fancy technique at all!

Data Cleaning

As noted above, a discussion of data cleaning and data wrangling does not belong in the body of your report. Data cleaning is where you probably spent *most* of your actual time, so it's natural to want to demonstrate how much effort you put into a project by detailing all of the painstaking merging, string cleaning, harmonizing, etc. work you put into your project. All of that can go into an appendix, but including it in a report only makes it more likely that all of that effort will go to waste

because reading about it will bore your stakeholder into setting down your report before you want them to.

There's a famous saying in writing [that appears to have first appeared in a 1913 Cambridge Lecture by Arthur Quiller-Couch]:

> If you here require a practical rule of me, I will present you with this: 'Whenever you feel an impulse to perpetrate a piece of exceptionally fine writing, obey it — whole-heartedly — and delete it before sending your manuscript to press. **Murder your darlings.**'

Well, when it comes to data science, you don't have to murder those darling paragraphs about all the messy data you had to clean up, but you *should* shove them in an appendix.

Oh, and please *never* use verbatim variable names in reports — they're fine in appendices, but in the body of a report you should explain what you're measuring in substantive terms someone without the data documentation can understand.

Discretion

The exception to the "don't talk about your data manipulations" rule is that when you were forced to exercise substantial discretion in a way that may impact the internal or external validity of the analysis in a meaningful way (for example, selection of control states, or dropping certain time periods from the analysis you feel are too anomalous).

Even in these situations, however, it is usually best to explain the *logic* behind the decisions in the body of the report (like in choosing what entities belong in your control group). However, a full discussion of these kinds of choices belong in an appendix.

When you do exercise substantial discretion in an analysis (for example, deciding to include certain entities as controls while excluding others), **it's best practice to include some sensitivity analyses in your report** by examining how your results do (or hopefully do not) change if you exercise discretion in slightly different ways. A sensitivity analysis, as a reminder, is where you make different discretionary choices (like choosing what entities to use as controls, or what variables you include in your regression), then re-run your analysis to see if the results change. The goal is to show that while, yes, you did have to exercise discretion in your analysis and data

manipulations (we always do!), the choices you were forced to make aren't driving your results and conclusions.

Ideally, you can just reference that "results are similar when [example of doing something a little differently]" in the body of the report and include the sensitivity analysis in an appendix.

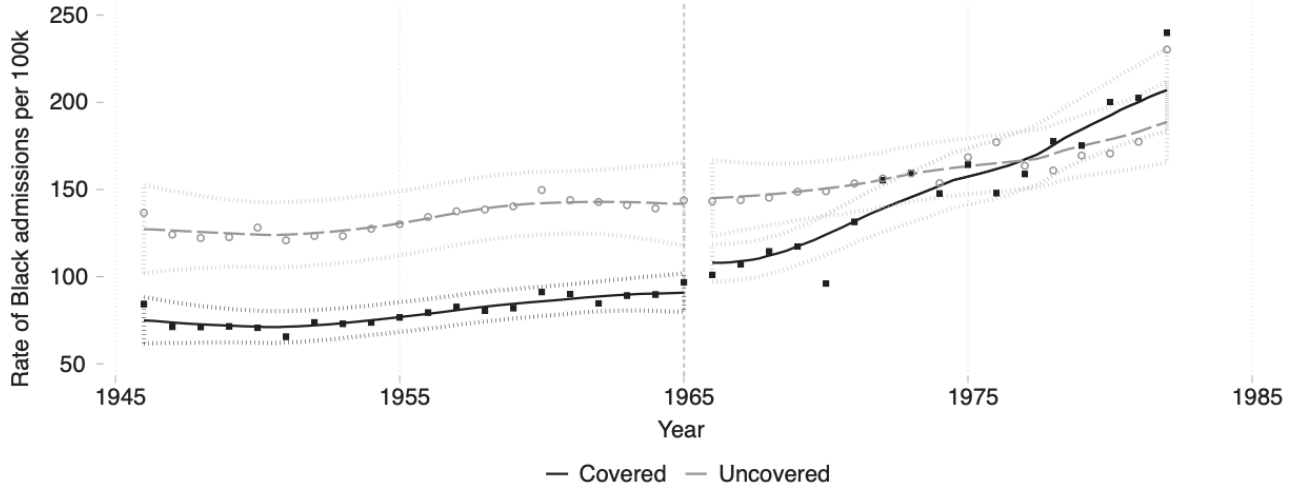
If different discretionary choices *do* lead to substantively different results, then you need to (a) try to characterize how fragile your results are (how much do they change in response to how small of a tweak in your choices), and/or (b) defend why the choices you made were correct, and the choices that give different results are not.

Figures

Not everyone may agree with this approach, but my philosophy with plots is that they should be more or less freestanding — if a reader were to only look at the figures in a report, the labels on the axes, the titles, and any notes associated with the figure, they should be able to get a pretty good understanding what's going on without having to go read the text in the report.

For example, this plot (which, yes, is from a paper on which I am a co-author, but all credit for the figure quality goes to my coauthor):

FIGURE 2. Section 5 Coverage and the Black Prison Admissions Rate



Note: The sample of states includes Alabama, Georgia, Louisiana, Mississippi, North Carolina, South Carolina, and Virginia (covered) and Arizona, Delaware, Florida, Kentucky, Maryland, Missouri, New Mexico, Oklahoma, Tennessee, Texas, and West Virginia (not covered). The scatter presents the averages of the raw data for states in each coverage category. The lines are local polynomial fits (bandwidth = 2) and 95% confidence intervals. Note that due to software limitations, standard errors in these plots do not reflect uncertainty due to missing data imputation. See Table 1 for analogous estimates with corrected standard errors.

Limitations

There is a tendency for students to use their "Limitations" sections to, well... just try and cover their butts by throwing out anything they can think of about the paper that is imperfect. That's ok in the classroom, but it's not useful in the real world.

The point of a Limitations section isn't to demonstrate your ability to identify any imperfections in the study; the point of a limitation section is to give your stakeholder a sense of how much confidence they should have in the results presented in the report *from your professional perspective.* Just because you had to make an assumption does not mean that the assumption constitutes a "limitation" of the study unless you have reason to think that the assumption is unlikely to be true (or is sufficiently untrue as to impact the results). Please only include things in your limitation section that you think really are substantive limitations!

Interpretable Models

For most applications, perform just as well as fancy pants models. https://arxiv.org/abs/1811.10154
Allow for non-specialists to see what goes into a model to debate ethics (after all, we data scientists have specialized knowledge when it comes to statistical methods, but not ethics)